Proofs, Types, and Lambda Calculus

Andrew Polonsky

December 4, 2010

Contents

Introduction

The present monograph reports our work during the PhD project *Automating Coherent Logic* (ACL) at University of Bergen, Norway.

The thesis consists of two parts covering proof theory and lambda calculus. Although the focus in each case is somewhat different, both investigations stress the syntactic approach to the study of proofs and terms.

The two subjects are closely related through type theory, via the result known as *Curry–Howard isomorphism.* This is a statement of correspondence between proofs of logical formulas on the one hand, and lambda terms and their types on the other.

The two parts also differ in the style and character of the writing. This structure of presentation was driven by the desire to help the readers of diverse motivations access the information most relevant to their interests as expediently as possible.

Part I, written with the view toward practical applications, describes in detail the development of an efficient translation algorithm from first-order logic to coherent logic. It can be read as a report on the ACL project itself, as a technical manual for the translator that we developed, or as a general, gentle introduction to coherent logic. We hope it may be helpful to future developments in the subject.

Part II presents some results in the pure lambda calculus. Our main contribution is the refutation of the range property for $\mathcal{H}$ — a conjecture of Barendregt concerning images of λ-terms modulo identification of meaningless terms. This part contains the primary mathematical content of the thesis. Accordingly, its style is formal and its goal is to present the proofs as concisely as possible.

The unifying theme of this thesis is the relationship between logic and computation, as mediated by type theory. Thus the core formalism studied in each parts is based on a Turing-complete programming language, giving it universal computational capabilities. Yet it is the logical content of the programs which makes it interesting to study them.

In Part I, the logical content of programs is just the usual Curry–Howard isomorphism between proofs and types. Because the proof theory of coherent logic is constructive, any proof produced by a coherent prover can be readily transformed to a lambda term inhabiting the corresponding type. With the translation from first-order logic that we develop, this property of coherent logic allows one to build a fully automated first-order prover whose proof objects are acceptable in the various interactive proof environments used for formalizing mathematics.

In Part II, our counterexample to the range property requires deciding Σ_1-sentences at every stage of the construction. The logical power of such sentences is equivalent to the halting problem, so these decisions cannot be done computably. We circumvent this difficulty by using equality of the λ-theory $\mathcal{H}$, which is Π_2-complete. As a result, the construction is rendered sufficiently effective to be realizable in a lambda term.

The translation algorithm developed in Part I has been fully implemented using the LISP language. We will be happy to provide the source code to the reader upon request.

For comments, questions, or anything else, please contact the author at andrew.polonsky@gmail.com.

Finally, I would like to express my gratitude to the following people.

Marc Bezem, for being the best PhD advisor I've ever had.

Stefan Berghofer and John Fisher, for very productive and rewarding collaboration on the project.

Dag Hovland, for being a great officemate during my PhD years. Also, I thank all my friends and acquaintances at the informatics institute at the university in Bergen for all their help and stimulation.

I would like to thank my family, which brought me to the point at which I could even write this thesis — especially Oksana, Sergiy, Maxim, and Rosemary. I am forever grateful to Amy Dunbar, John Phillips, and Joel Marks for making my university education possible. Joel inspired me to pursue science more than any other person.

I would also like to thank the amazing faculty at the department of mathematics at the University of Connecticut. I am especially grateful to my academic parents Stuart Sidney, Keith Conrad, Rich Bass, Tara Holm, Ron Blei, and Wolodymyr Madych.

Finally, I thank all the people that made my years in Bergen as fun as they were: our “D12 family” at Fantoft, my Russian friends Boria, Gleb, and Max, my fellow logicians Paul Simon, Truls, and Piotr, and everyone else who made Bergen such an exciting part of my life.

I thank Irine Røsnes for infinite inspiration.

December 4, 2010
Amsterdam

Andrew Polonsky

Part I

Coherent Logic

Chapter 1

Aspects of Coherent Logic

Coherent logic, which will henceforth be usually abbreviated as "CL", can be viewed from many angles. Although formally it is only a fragment of full first-order logic (FOL), CL theories can be used to represent arbitrary FOL formulas, and the correspondence extends to the level of proofs as well. Yet it is the remarkable stock of features native to CL itself which makes this correspondence useful. For example, proof-theoretic analysis of CL sheds new light on Skolemization — a concept dating back to the earliest results of mathematical logic — with useful applications in both theory and practice.

Coherent logic is the subset of FOL consisting of formulas of the form

$$\begin{gathered} A_1 \wedge \cdots \wedge A_m \to B_1 \vee \cdots \vee B_n, \\ \text{with } \; B_j = \exists \vec{y}. C_{j,1} \wedge \cdots \wedge C_{j,l_j} \end{gathered} \tag{1.1}$$

Here A_i and $C_{j,k}$ are atomic formulas, and the entire expression is implicitly universally closed (so while we don't write the quantifiers, the free variables of the clause are always quantified over on the outside). One can view the above definition as an extension of resolution logic; while resolution requires that all of A_i and B_j are atoms, CL allows B_j to contain existentially quantified conjunctions of atoms. This simple relaxation has a number of significant consequences.

Notice that every logical connective occurs in the general format above — this is what allows CL to represent full first-order logic. Most importantly, the presence of existential quantifiers *makes Skolemization unnecessary* in translation from FOL to coherent clauses. This motivates the main practical application of CL — as an automated prover logic from which proof objects could be recovered without the infamous problems caused by introduction of Skolem functions (Bezem et al. [2000]).

The question is whether the above property can be exploited to create an efficient first-order prover with explicit proof objects. The translation from general first-order formulas into the coherent fragment would be the central piece of any such system, and this is the primary focus of our investigation here. We develop a translator which is flexible, practically useful, and keeps the task of proof reconstruction easy.

To give the reader a feeling for what CL has to offer, we briefly discuss some of the unique features of this curious logic.

1.1 A positive dimension of first-order logic

CL consists of clauses which contain only positive operators on both sides of the arrow. This motivates the forward proof search which proceeds by extending the proof state with atoms on the right of a clause whenever one has satisfied all of the atoms on the left. A fundamental result of CL is that this proof procedure is complete — that is, any fact logically entailed by a CL theory will be found by the forward search.

One consequence of the completeness theorem is that the proof theory of CL becomes very simple, having just a single rule of deduction (the extension of the proof state with new facts). CL proofs can even be given an intuitive and visually appealing presentation as trees of facts, from which the structure of rule applications can be read off immediately. The simplicity of proofs means that proof objects are always readily available, in stark contrast to the proofs by resolution.

Another consequence of the completeness of positive deduction is that CL proofs are constructive. That is, any consequence of a CL theory can be derived from it using only intuitionistic logic.

1.2 A powerful intuitionistic fragment

In logic one is often interested in classes of formulas for which classical and intuitionistic provability coincide. One such class is that of Harrop formulas; another is the image of Glivenko–Gödel double-negation translation. Both of these classes are quite restrictive and very syntactical in nature. At the same time, an implication with a finite number of coherent antecedents can be interpreted as a consequence relation inside a topos, as we will discuss below.

The interest in constructive proofs often comes from working in a proof assistant whose underlying logic is constructive. Completeness of CL shows that if, within such a system, one is able to derive a coherent theory, then one can get any formula implied by this theory with little effort. In fact, this procedure allows for a straightforward automatization, which, together with the translation of general FOL theories into CL theories, gives an all-purpose first-order tactic for interactive theorem provers.

However, the translation itself is not constructive (obviously it cannot be, since not all proofs are intuitionistic). Proving FOL problems via coherent logic therefore gives us an unusual factorization of an arbirary first-order proof into a non-constructive part (the CL translation), and a constructive part (the CL proof). From this vantage point, CL can be seen as a method of filtering constructive content from general proofs.

1.3 A notation for sequent proofs

Coherent logic has a single rule of inference. Nevertheless, using this rule one can represent reasoning in full FOL. A possible view of CL, then, is that it combines the four rules of the complete tableau calculus into one, giving a uniform treatment of all logical phenomena in one fell swoop.

While the relationship between coherent logic and analytic tableau is the most intimate one, it can be extended to nearly the whole of sequent calculus. The left-sided rules (basically, the tableau rules) are represented immediately, because the primary derivation rule of CL is essentially a generic elimination rule.

Our final translation algorith will also suggest how introduction rules can be represented. As a result, the inference process can be made even more positive, in the sense of not being necessarily refutation-based. However, there's one gap: we cannot faithfully represent the right sequent rule for implication. The discharge mechanism of CL does not suffice to internally represent discharging of assumptions in the implication introduction. A similar problem occurs for the universal quantifier, but there it can be circumvented by a trick which is dual to Skolemization. Still, it is possible that a satisfactory treatment of introduction rules can be given based on a semantic forcing relation for CL, as defined in (Coquand [2003]).

1.4 An automated prover for interactive proof assistants

A central application of coherent logic is in proof assistants used for formalizing mathematics. Formalizing a non-trivial result in such provers usually requires a large amount of effort, most of which is spent on technical detail too trivial to be included in published proofs. It is perhaps the fundamental problem in formalization technology today — that explaining trivial things to an interactive prover is a non-trivial task.

The situation could be much improved if the full power of automated theorem proving was available to a user of one of these systems. Unfortunately, the best automated theorem provers today are based on the resolution method, and the notion of proofs in this framework is too weak for prover output to be acceptable as a valid proof in interactive systems like Coq or Isabelle, most of which are based on type theory. (Although, resolution provers can be useful as relevance filters; Isabelle's *sledgehammer* tactic uses them to assist proof search of an internal tableau prover.) In particular, the clause normal form (CNF) transformation converts the input formula into a form which is no longer logically equivalent with the original. As a result, it is not possible to recover a lambda term whose type is the original problem from the proof trace of a resolution prover.

The fatal problem comes from Skolemization, an essential step during CNF transformation. When a formula such as $\forall x(P(x) \rightarrow \exists y.P(y))$ is converted into $\forall x.(P(x) \rightarrow P(a))$, it loses the property of being a tautology. And even if one succeeds at refuting the Skolemized formula, one needs to assume choice axioms in order to lift this proof to a proof of the original formula. Such axioms are inherently non-constructive, and may even be inconsistent with the larger proof context one is working in. (Bezem et al. [2000], de Nivelle [2003])

In contrast, CL does not need Skolemization. The presence of existential quantifiers renders it unnecessary. Proof objects can be recovered from traces of CL proofs with virtually no effort. Finally, even if, for efficiency reasons, one chooses to introduce Skolem functions into the input, *they can be eliminated from the proof at no cost in complexity.* This indicates that the common view that eliminating Skolem functions must necessarily yield exponentially longer proofs is not entirely accurate.

1.5 A first-order projection of geometric logic

CL has its origins in topos theory. The logic of toposes is commonly called *geometric logic*, because the internal logic of any topos is intimately connected with the geometry of its underlying sheaf.

Equivalently, geometric logic can be described as the language consisting of clauses of the form

$$A_1 \wedge \cdots \wedge A_n \to \bigvee_{\alpha} B_{\alpha,1} \wedge \cdots \wedge B_{\alpha,m_\alpha}$$

Note that now the disjunction is infinitary, and may be taken over an arbitrary set of indices α. One can motivate this definition by the following stipulation. Suppose $A_i, B_{\alpha,j}$ are elements of a subbasis of a topological space. Then the left side of the implication denotes some basic set, while the right side denotes some open set. Hence, a geometric formula is the statement of inclusion of a basic set into an open set, while a geometric theory expresses inclusion relations between general open sets. That is, it defines a topology. The "points" of this formal space are precisely the models of the theory.

Geometric logic is inherently constructive, but what gives it foundational significance is that it can characterize all of constructive logic. This can be seen by considering the nature of the semantics of intuitionistic logic. In classical logic, the collection of propositions ordered by logical entailment has the structure of a Boolean algebra — the Lindenbaum–Tarski algebra of propositional logic. Going over to first-order logic, we are thus lead to interpret predicates as arbitrary subsets of the domain, i.e. by elements of the domain's powerset algebra, which is a complete Boolean algebra. In intuitionistic logic however, the Lindenbaum–Tarski algebra induced by the syntax is a Heyting algebra — whose completion is a frame. Hence predicates should be interpreted not by arbitrary subsets, but by a collection of subsets which form a topology. The operations of frames (finite meets and arbitrary joins) are fully represented by the format of geometric formulas. Indeed, geometric logic is exactly the language required to give a description of an arbitrary frame. For a more detailed discussion, consult Mulvey [2003].

An important program currently underway aims to use geometric logic to give constructive interpretations of central theorems of classical mathematics. For references, see Spitters, Coquand, Mulvey.

1.6 A Turing-complete programming language

Since CL is constructive, it automatically inherits the Curry–Howard isomorphism between proofs and programs from type theory. That is, any CL proof can be readily seen as a lambda term inhabiting the type of the formula proved. Explicitly, the correspondence has the following form. If $\mathcal{T}$ is a coherent theory, $\Gamma = \{A_1, \ldots, A_n\}$ is a set of facts, and *Goal* is an atomic formula, then $\Gamma \vdash_{\mathcal{T}} \mathit{Goal}$ in coherent logic if and only if the context of assumptions from $\mathcal{T}$ yields an inhabitant of type $A_1 \to \cdots \to A_n \to \mathit{Goal}$.

Fisher and Bezem [2007] showed that one can go further — that one can view CL itself as a model of computation. Specifically, they introduced operational semantics for coherent "programs". This semantics takes the form of *Skolem machines*, idealized devices implementing basic forward search. Their computational power is equivalent to that of Turing machines — a fact which is ultimately responsible both for the undecidability and the great expressive power of coherent logic (equal to full FOL).

That CL has intrinsic computational content canonically related to its proof theory is yet another illustration of its conceptual elegance.

Before we proceed, a word of caution. In Part I of this thesis, footnotes will almost exclusively be used to revive the definition of a term or concept introduced earlier. The reader is hereby advised to stay clear of footnotes when they appear in a sentence whose meaning is clear.[1]

[1] In this manner we attempt to restore the original function of the footnote as an expository device, which draws its usefulness from knowing that the reader will not follow it unless she is in want of additional comments or clarifications. This enterprise proceeds by repeatedly reminding the reader that the utility of reading a footnote could well be void, a fact presently demonstrated by reflection.

Chapter 2

The Forward Search and the Basic Translation from FOL

In this chapter we discuss the basic proof search of CL. After giving a definition of coherent provability, we present the first version of translation of general FOL formulas into the coherent fragment. Together, this yields a semi-complete decision procedure for FOL. Refutation-completeness is motivated by close correspondence with the tableau method**?**. The proof of the latter will serve to prepare the reader for the optimized versions of translation to appear later, since the correspondence with tableau calculi forms the basis for proving completeness of those translations as well. We give examples as we work through the chapter.

2.1 Ground forward reasoning

The definition of provability for coherent logic is particularly simple in the case of ground reasoning. Here one restricts attention to closed formulas and views coherent clauses as schema for deriving new facts by instantiating free variables by ground terms.

Notation. In what follows, $\mathsf{FV}(\varphi)$ will denote the set of variables free in a formula φ. We say that φ is *ground* if $\mathsf{FV}(\varphi) = \varnothing$; if φ is ground and atomic, then φ is a *fact*. Facts will usually be denoted by upper-case latin letters $(A, B, C...)$, while greek letters $(\varphi, \psi, \chi...)$ will stand for general FOL formulas. The letters σ and τ denote substitutions — finite maps from variables to terms. In this chapter, we'll only be looking at ground substitutions, which map variables to closed terms. We write φ^σ for the result of substituting each free occurrence of a variable x in φ with $\sigma(x)$.

Definition 1. Let Γ be a set of facts, $\mathcal{T}$ a coherent theory, φ a fact. The relation $\Gamma \vdash_\mathcal{T} \varphi$ is defined by induction:

Base case $\Gamma \vdash_\mathcal{T} \varphi$ if $\varphi \in \Gamma$.

Induction Let $L = (\bigwedge A_i \to \bigvee B_j)$ be a clause in $\mathcal{T}$,
and $\sigma : \mathsf{FV}(L) \to \mathsf{dom}(\Gamma)$ a substitution with $\{A_i^\sigma\} \subseteq \Gamma$.
If for each $B_j = \exists \vec{y}.C_1 \wedge \cdots \wedge C_{k_j}$, $\Gamma \cup \{\overline{C_i^\sigma}\} \vdash_\mathcal{T} \varphi$, then $\Gamma \vdash_\mathcal{T} \varphi$.

The above definition has two additional pieces of notation which we introduce now. For a set of formulas Γ, the notation $\mathsf{dom}(\Gamma)$ refers to the set of ground terms over constant and function symbols in Γ, which we choose to call *the domain of* Γ. In the literature, $\mathsf{dom}(\Gamma)$ is often called the *Herbrand universe* of Γ. It is defined inductively by:

- $c \in \mathsf{dom}(\Gamma)$ if c is a constant symbol occurring in $\mathcal{T}$ or in Γ. If no such symbols exist, it is allowed to take for c the particular symbol o.[1]
- $f(t_1, \ldots, t_n) \in \mathsf{dom}(\Gamma)$ if $t_1, \ldots, t_n \in \mathsf{dom}(\Gamma)$ and f is a function symbol occurring in $\mathcal{T}$ or in Γ.

The ground forward search we defined above involves dynamically extending the domain with new witnesses. If φ is a formula, then by $\overline{\varphi}$ we denote the formula obtained by replacing every free variable x in φ by a *fresh* constant c_x. (In this context, "fresh" means that c_x should not occur in either Γ or $\mathcal{T}$. So in the definition above, $\overline{C_i^\sigma}$ denotes the set of formulas $\{C_i\}$ in which the universally quantified variables $\vec{x}$ have been replaced by $\sigma(\vec{x})$, and the existentially quantified variables $\vec{y}$ have been replaced by new constant symbols. This set is then added to Γ for the inductive inference.

Example 2. Let $\mathcal{T}$ be the following theory.

$$\mathcal{T} = \left\{ \begin{array}{rcll} E & \to & \exists y.(Q(y) \wedge D(y)) & (1) \\ D(x) & \to & \mathit{Goal} \vee (P(x) \wedge E) & (2) \\ P(x) \wedge Q(x) & \to & \mathit{Goal} & (3) \end{array} \right.$$

Then $\{D(a)\} \vdash_\mathcal{T} \mathit{Goal}$. One possible derivation is the following:

[1]This is done to prevent pathologies relating to empty domains.

$$\dfrac{D(a), Goal \vdash_{\mathcal{T}} Goal \qquad \dfrac{\dfrac{D(a), \cdots, D(c_y), Goal \vdash_{\mathcal{T}} Goal \qquad \dfrac{D(a), \cdots, P(c_y), \, Goal \vdash_{\mathcal{T}} Goal}{D(a), \cdots, D(c_y), P(c_y) \vdash_{\mathcal{T}} Goal}\,3}{D(a), P(a), E, Q(c_y), D(c_y) \vdash_{\mathcal{T}} Goal}\,2}{D(a), P(a), E \vdash_{\mathcal{T}} Goal}\,1}{D(a) \vdash_{\mathcal{T}} Goal}\,2$$

The numbers to the right of the inference lines indicate which clause from $\mathcal{T}$ has been used in the corresponding inductive inference. Notice the different substitutions that were used in the two applications of clause (2).

We have given the full derivation above to illustrate the effects of Definition 1. For the purposes of concise presentation, this sequent-style format is much too bulky. We shall now introduce a leaner notation.

Since the formula on the right side of "$\vdash$" never changes while the sets on the left increase upwards from the root, the structure of the derivation in Example 2 is completely determined by which formulas are added into the context — the set of facts known at a particular branch — by each application of a rule[2] in $\mathcal{T}$. Hence the tree above can be represented in the following manner:

$$\begin{aligned}
&\{D(a)\} \vdash_2^{a/x} \{Goal\} \\
&\qquad \vdash_2^{a/x} \{P(a), E\} \vdash_1 \{Q(c_y), D(c_y)\} \vdash_2^{c_y/x} \{Goal\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdash_2^{c_y/x} \{P(c_y)\} \vdash_3^{c_y/x} \{Goal\}
\end{aligned}$$

Here the turnstile symbol is overloaded to mean that the formulas immediately to the right are obtained from all of the preceding facts by an application of the clause in the subscript under the substitution in the superscript.

This notation, while being easier to read, also stresses a different aspect of Definition 1. On the literal reading, the definition seems to describe an inverse (or *negative*) inference process: the inductive step states which hypotheses are needed in order to derive the conclusion rather than listing the consequences of the facts that we have. Hence the structure of the sequent-like tree above, with the facts data increasing as we go backwards from the last inference. The shorter notation suggests an alternative view of coherent provability: the clauses are rules stating which facts we may conclude from what we have now, with several possibilities in the presence of disjunctions.

[2]The word "rule" ambiguously refers either to a clause in $\mathcal{T}$ or to a closed instance of a clause in $\mathcal{T}$.

This *positive* view more directly reflects the proof search performed by the current implementation algorithms. (Although CL provers based on the inverse method are also possible.) At the end of this chapter, we give a brief survey of these.

2.2 The role played by negation

In general, either of the two sides of a coherent clause may be empty. This is reflected by the cases $m = 0$ and $n = 0$ in the general form

$$A_1 \wedge \cdots \wedge A_m \rightarrow B_1 \vee \cdots \vee B_n \tag{2.1}$$

of a coherent clause. As a notational matter, we will write $\top$ on the left side when $m = 0$, and we write $\bot$ on the right when $n = 0$. Let us now observe the effect of having a clause of the form $\bigwedge A_i \rightarrow \bot$ in the context of Definition 1.

Supposing that we find ourselves in the inductive case with a substitution σ under which the A_is become members of Γ, we see that the condition on the B_js is immediately satisfied (vacuously), and hence that $\Gamma \vdash \varphi$.

So whenever we can satisfy the left side of a clause whose right side is empty, then we can conclude the formula φ, regardless of what it is. This is the *ex falso* rule, stating that anything follows from absurdity, and being an important element of CL provability.

Dually, when the left side of (2.1) is empty, the hypotheses of the clause are always satisfied, and the conclusion can be applied under any substitution. That is, the facts on the right side can always be added into the context Γ (possibly introducing new branches). This allows us to add axioms (initial facts) to the theory $\mathcal{T}$: we simply add the clause $\top \rightarrow A$ for each fact A that we want to be available as an axiom in every $\mathcal{T}$-derivation. Hence the clauses of a coherent theory serve both as the specification of initial data and as the rules for deriving new facts from old.

The clause $\top \rightarrow \bot$ is a contradictory clause: it exposes the fact that the theory $\mathcal{T}$ is a priori inconsistent. Similarly to its role in resolution, where a derivation of the empty clause signals the end of the refutation process, this clause sometimes appears when a function simplifying the translation for a CL prover can already detect that the input was unsatisfiable. Invocation of the prover is thus rendered unnecessary.

Going back to $\perp$, we note that subsumption of the ex falso rule by Definition 1 gives CL an adequate mechanism of negation, just as this rule implements negation in Gentzen's Natural Deduction. As an axiom, the fact $\neg A$ can be represented by the clause $A \to \perp$. But in order to reason about negated facts, it is necessary to introduce for every such A a new symbol $\overline{A}$ that will stand for $\neg A$. The coherent theory should then be augmented with the "bottoming out" clauses that have the form

$$A(\vec{x}) \wedge \overline{A}(\vec{x}) \to \perp \tag{2.2}$$

This rule says that whenever we concluded both $A(\vec{t})$ and its negation, we may finish the proof of the current branch. Using tableau terminology, we say that the branch is *closed*. Our notation suggests that (2.2) can end the proof by a "derivation" of $\perp$. We will adopt this interpretation, and will often talk about deriving $\perp$, leaving the actual meaning implicit.

The canonical translation of FOL formulas into CL formulas which we will describe in the next section involves introduction of auxiliary predicates for every subformula of the input. The question arises of how to treat negated subformulas: whether we should keep all negations in place and have both positive and negative polarities for introduced predicates, or whether to "push down" all of the negations by repeated applications of DeMorgan's laws. In the latter case, we will only need to deal with negations at the atomic level, making the translation process considerably less complicated.

The same dilemma appears in the theory of tableau systems. Following Smullyan [1995], many authors consider tableaux consisting of the so-called *signed formulas*, where each formula occurring in the tableau comes with a "sign" stating whether its polarity is positive or negative. As a result, every logical operator has two corresponding tableau rules — one for each sign of the formula**?**. Others prefer to first transform the formula into its *negation normal form* (NNF) using DeMorgan's laws and work with only positive polarities of the formulas. Each connective then has its own unique tableau rule.

One drawback of using negation normal forms is that some of the DeMorgan's laws are not constructive. The NNF translation also rewrites non-basic connectives such as implication in terms of conjunction, disjunction, and negation — this can also be an issue. Most notably, the transformation makes use of the double negation elimination rule $\neg\neg P \to P$, which is equivalent to the law of excluded middle. The translation is thus inherently non-constructive.

This does not pose a big problem for us because we are interested in full first-order logic, and there can be no constructive translation which is complete for classical tautologies (in the sense of making them provable by a coherent logic prover). However, it would indeed be desirable to have a constructive variant that is complete for intuitionistic tautologies. The existence of such a translation is an interesting open problem.

We will make use of the NNF transformation in all of our translations. Although the later versions reintroduce polarities in a manner which is related to signed formulas from tableaux, the *occurrences* of non-atomic subformulas will always be made positive with respect to the whole formula. So the first step of our translation from first-order logic to coherent logic is to take the negation normal form of the input formula.

Definition 3. Let P be a FOL formula. The *Negation Normal Form of* P, written $\mathsf{NNF}(P)$, is the (unique) normal form of P under the following rewrite rules:

$$\begin{array}{ccc}
(\varphi \to \psi) & \longrightarrow & (\neg\varphi \vee \psi) \\
(\varphi \leftrightarrow \psi) & \longrightarrow & (\varphi \to \psi) \wedge (\psi \to \varphi) \\
\neg\neg\varphi & \longrightarrow & \varphi \\
\neg \bigwedge \varphi_i & \longrightarrow & \bigvee \neg\varphi_i \\
\neg \bigvee \varphi_i & \longrightarrow & \bigwedge \neg\varphi_i \\
\neg\exists\vec{x}\varphi & \longrightarrow & \forall\vec{x}\neg\varphi \\
\neg\forall\vec{x}\varphi & \longrightarrow & \exists\vec{x}\neg\varphi
\end{array}$$

The formula $\mathsf{NNF}(P)$ is equivalent to P, but only has negations in front of atomic subformulas. It looks like a tree of $\wedge$ and $\vee$, occasional $\forall$ and $\exists$, and leaves which are *literals* — possibly negated atomic formulas.

2.3 The first translation

The basic idea for a translation from FOL to CL was proposed by Bezem and Coquand [2005]. It consists of introducing a new atomic predicate for every subformula of the input, and adding a clause stating the precise relationship between the subformula and its immediate children. Because the format of CL contains every logical connective, we are able to generate such clauses for every subformula. Essentially this amounts to a simulation of the tableau method within the CL framework. The CL theory produced by the translator encodes a "map" of an analytic tableau for the given formula, and a refutation of the formula by a tableau system corresponds to a derivation of $\bot$ from the theory by a CL prover. This illustrates for the first time the close connection between ground tableau and coherent provability.

We now define the canonical translation from FOL to CL.

Definition 4. Let Φ be a first-order formula in negation normal form.

- For an atomic predicate A occurring in Φ, let
  - T_A, F_A be fresh predicate symbols of the same arity as A,
  - C_A be the coherent clause $T_A(\vec{x}) \wedge F_A(\vec{x}) \to \bot$.
- For a literal[3] $L \subseteq \Phi$, define the atomic formula

$$L^t = \begin{cases} T_A(\vec{t}\,) & \text{if } L = A(\vec{t}\,) \\ F_A(\vec{t}\,) & \text{if } L = \neg A(\vec{t}\,) \end{cases}$$

- For a compound (non-literal) subformula $\varphi \subseteq \Phi$, let
  - T_φ be a fresh predicate of arity $|\mathsf{FV}(\varphi)|$,
  - φ^t be the formula $T_\varphi(\vec{x})$, where $\vec{x} = \mathsf{FV}(\varphi)$,
  - C_φ be the coherent clause defined according to the top connective in φ (with implicit universal quantification over all free variables that occur in it):

φ	C_φ
$\varphi_1 \wedge \cdots \wedge \varphi_n$	$\varphi^t \to \varphi_1^t \wedge \cdots \wedge \varphi_n^t$
$\varphi_1 \vee \cdots \vee \varphi_n$	$\varphi^t \to \varphi_1^t \vee \cdots \vee \varphi_n^t$
$\exists \vec{y} \psi$	$\varphi^t \to \exists \vec{y} \psi^t$
$\forall \vec{y} \psi$	$\varphi^t \to \psi^t$

[3]Recall that L is a literal if it is either an atomic formula or a negation of such.

- The *canonical coherent theory of* Φ is the set

$$\mathcal{T}_\Phi = \{C_A | A \in \Phi\} \cup \{C_\varphi | \varphi \subseteq \Phi\} \cup \{\top \to T_\Phi\}$$

Definition 5. Let Γ be a first-order theory. The *canonical translation* of Γ is

$$\mathcal{T}_\Gamma = \bigcup_{\varphi \in \Gamma} \mathcal{T}_{\mathsf{NNF}(\varphi)}$$

Example 6. Suppose we wish to translate into coherent logic the negation of the formula

$$\varphi = (\forall xy. Ax \vee By) \to (\forall x. Ax) \vee (\forall y. By)$$

The first step is to compute the negation normal form of $\neg\varphi$:

$$\mathsf{NNF}(\neg\varphi) = (\forall xy. Ax \vee By) \wedge (\exists x. \neg Ax \wedge \exists y. \neg By)$$

Now the new predicates T_ψ for every $\psi \subseteq \mathsf{NNF}(\neg\varphi)$ are introduced: $T_{(\forall xy.Ax \vee By) \wedge (\exists x.\neg Ax \wedge \exists y.\neg By)}$, $T_{\forall xy.Ax \vee By}$, $T_{\exists x.\neg Ax \wedge \exists y.\neg By}$, $T_{Ax \vee By}(x, y)$, $T_{\exists x.\neg Ax}$, $T_{\exists y.\neg By}$, $T_A(x)$, $F_A(x)$, $T_B(x)$, $F_B(x)$.

Finally, one constructs the clauses as per Definition 4:

$$\mathcal{T}_{\neg\varphi} = \begin{cases} \top & \to \quad T_{(\forall xy.Ax \vee By) \wedge (\exists x.\neg Ax \wedge \exists y.\neg By)} \\ T_{(\forall xy.Ax \vee By) \wedge (\exists x.\neg Ax \wedge \exists y.\neg By)} & \to \quad T_{\forall xy.Ax \vee By} \wedge T_{\exists x.\neg Ax \wedge \exists y.\neg By} \\ T_{\forall xy.Ax \vee By} & \to \quad T_{Ax \vee By}(x, y) \\ T_{Ax \vee By}(x, y) & \to \quad T_A(x) \vee T_B(y) \\ T_{\exists x.\neg Ax \wedge \exists y.\neg By} & \to \quad T_{\exists x.\neg Ax} \wedge T_{\exists y.\neg By} \\ T_{\exists x.\neg Ax} & \to \quad \exists x. F_A(x) \\ T_{\exists y.\neg By} & \to \quad \exists y. F_B(y) \\ T_A(v) \wedge F_A(v) & \to \quad \bot \\ T_B(v) \wedge F_B(v) & \to \quad \bot \end{cases}$$

The reader may have noticed that φ is a first-order tautology. Thus $\neg\varphi$ is unsatisfiable, and we should hope that its translation would reflect this property. Indeed, using the rules of coherent logic (Definition 1), it is possible to derive $\bot$ from the theory $\mathcal{T}_{\neg\varphi}$, without any extra axioms. One possible derivation is shown in Figure 2.1.

In this figure, dom is the domain restriction predicate asserted for every new term added to the proof search, and sk_0 is the Skolem constant introduced when firing the δ rule at step 5. Recall that although $\bot$ is not an atom, the ex falso mechanism allows us to treat it as such — hence the predicate symbol false. Similarly, true denotes the empty state $\top$.

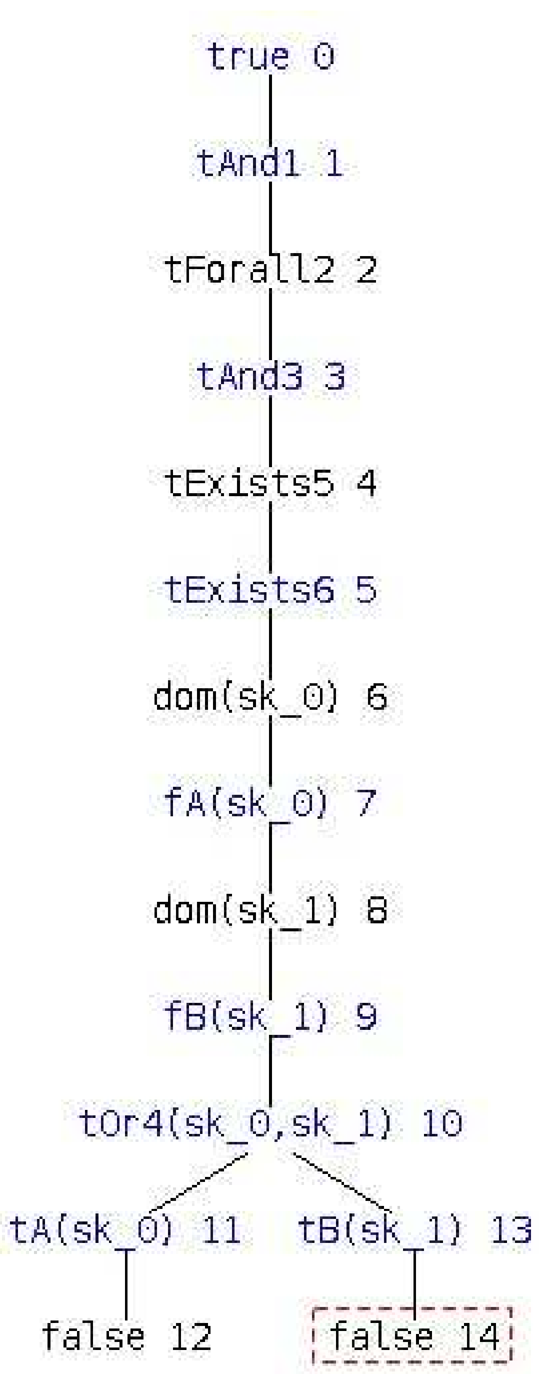


Figure 2.1: Example 6

Does the canonical translation always preserve satisfiability of input formulas? Of course it does! That was our principal motivation for investigating CL in the first place — to make it possible to prove a FOL formula φ by deriving $\bot$ from the translation of $\neg\varphi$. In order to give a formal proof of this fact we must make an excursion into elementary proof theory of classical first-order logic and explore more closely the relationship between proofs in FOL and proofs in CL.

2.4 Sequent calculus and analytic tableaux

This section gives a minimal introduction to Gentzen's sequent calculus and the tableau method. This is the system of choice for investigations into the proof theory of first-order logic (Buss [1998]). It is a fundamental property of this calculus, given by the celebrated Cut-Elimination Theorem of Gentzen [1935], which inspired the proof procedure bearing the name of Analytic Tableau.

Here we give an executive summary of these concepts; a reader seeking more information should consult Samuel Buss's *Introduction to Proof Theory* cited above.

Definition 7. A *sequent* has the form $\Gamma \vdash \Delta$, where Γ and Δ are sets of first-order formulas.

Definition 8. The *Sequent Calculus* LK is given by the following set of rules.

$$\frac{\varphi, \psi, \Gamma \vdash \Delta}{\varphi \wedge \psi, \Gamma \vdash \Delta} \wedge\mathrm{L} \qquad \frac{\Gamma \vdash \Delta, \varphi \quad \Gamma \vdash \Delta, \psi}{\Gamma \vdash \Delta, \varphi \wedge \psi} \wedge\mathrm{R}$$

$$\frac{\varphi, \Gamma \vdash \Delta \quad \psi, \Gamma \vdash \Delta}{\varphi \vee \psi, \Gamma \vdash \Delta} \vee\mathrm{L} \qquad \frac{\Gamma \vdash \Delta, \varphi, \psi}{\Gamma \vdash \Delta, \varphi \vee \psi} \vee\mathrm{R}$$

$$\frac{\varphi(t), \Gamma \vdash \Delta}{\forall x.\varphi, \Gamma \vdash \Delta} \forall\mathrm{L} \qquad \frac{\Gamma \vdash \Delta, \varphi(b)}{\Gamma \vdash \Delta, \forall x.\varphi} \forall\mathrm{R}$$

$$\frac{\varphi(b), \Gamma \vdash \Delta}{\exists x.\varphi, \Gamma \vdash \Delta} \exists\mathrm{L} \qquad \frac{\Gamma \vdash \Delta, \varphi(t)}{\Gamma \vdash \Delta, \exists x.\varphi} \exists\mathrm{R}$$

$$\frac{\Gamma \vdash \Delta, \varphi}{\neg\varphi, \Gamma \vdash \Delta} \neg\mathrm{L} \qquad \frac{\varphi, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg\varphi} \neg\mathrm{R}$$

$$\frac{\Gamma \vdash \Delta}{\varphi, \Gamma \vdash \Delta} \mathrm{WL} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \varphi} \mathrm{WR}$$

$$\frac{\Gamma \vdash \Delta, \varphi \quad \varphi, \Gamma \vdash \Delta}{\Gamma \vdash \Delta} \mathrm{Cut} \qquad \frac{}{\varphi \vdash \varphi} \mathrm{Axiom}$$

The term t appearing in the rules $\forall$L and $\exists$R represents an arbitrary first-order term. In contrast, the rules $\forall$R and $\exists$L are subject to a provision named *the eigenvariable condition.* This is a requirement that the constant b appearing in the hypotheses of these rules must be *fresh* — not occurring in either Γ or Δ.

As before, we write $\top$ in place of Γ when it is empty and $\bot$ in place of Δ whenever Δ is empty.

The *Cut-Elimination Theorem* states that any derivation in LK can be transformed into a derivation of the same sequent which does not make use of the Cut rule. This result is the cornerstone of proof theory. It is also the theoretical basis for the tableau method of automated deduction. The latter shall presently be illustrated.

Suppose that φ is a first-order tautology. Then $\mathsf{NNF}(\neg\varphi)$ is unsatisfiable, and thus $\mathsf{NNF}(\neg\varphi) \vdash \bot$ is a valid sequent. By the completeness theorem, there exists a derivation having this sequent as the root. Moreover by the cut-elimination theorem, there exists a derivation of $\mathsf{NNF}(\neg\varphi) \vdash \bot$ which does not make use of the Cut rule.

Let π be such a derivation. What are the possibilities for the last rule applied in π? In fact, the only possibilities are the left-side rules, because the right side of the root sequent is empty, and all of the rules $\boxdot$R introduce a formula on the right side (for $\boxdot \in \{\wedge, \vee, \forall, \exists, \neg, \mathrm{W}\}$), while Cut is not a rule that occurs in π.

Furthermore, with the exception of the rule $\neg$L, all of the left-sided rules have the property that *the right side of the hypothesis is the same as the right side of the conclusion.* That is, the set Δ remains empty until we come to an application of the $\neg$L rule.

But because the formula in the root is in negation normal form, $\neg$L could only be applied with an atom as the principal formula — the new formula to appear on the left side of the hypothesis. Consequently, *the only rules* which could appear in π, in addition to $\boxdot$L, are WR and Axiom.

It therefore follows that the proof of $\mathsf{NNF}(\neg\varphi) \vdash \bot$ (which *does* exist if φ is a tautology) has an exceedingly simple shape: it is nothing but a tree of applications of the four *logical* left introduction rules ($\boxdot$L with $\boxdot \in \{\wedge, \vee, \forall, \exists\}$), occasional shifting of negated atoms to the right side, weakenings, and axioms at the leaves. The suggestion of automating the search for such a tree now suggests itself automatically.

In the tableau formalism, we focus on the four left logical rules and abstract away from negation and weakening: these are subsumed by Ax* — a minimal strengthening of the Axiom rule that allows the branch to be closed whenever $A(\vec{t})$ and $\neg A(\vec{t})$ both appear on the left side of the sequent. Using Ax* in place of Axiom, the leaves now look like $\Gamma \vdash \bot$, with $A(\vec{t}), \neg A(\vec{t}) \in \Gamma$.

It therefore becomes unnecessary to talk about sequents, since Δ is always empty. It also becomes more convenient to look at the trees upside-down: with root at the top, and leaves on the bottom. At every node, we have a set Γ consisting of formulas which have been derived so far. The possible descendents of Γ are given by the four logical rules.

The logical rules get new names in the tableau world: the rules $\wedge$L, $\vee$L, $\forall$L, and $\exists$L are now respectively known by the names of α, β, γ, and δ. They are also expanded from being binary to having any finite number of successors. For example, a chain of $\exists$L rules can be represented by a single δ rule, which strips off all outermost existential quantifiers, replacing the quantified variables with fresh constants. When we want to emphasize the difference, we write $\boxdot$L* for the chained version of the rule $\boxdot$L.

By LK* we shall denote the sequent calculus obtained by using only the rule Ax* and the four logical rules $\boxdot$L*. *Applying* a rule to a formula $\varphi \in \Gamma$

means that Γ is expanded by the formulas occurring in the hypothesis of the corresponding LK* rule (having φ as the principal formula). The following definition should now be self-explanatory.

Definition 9. Suppose that Φ is a formula in negation normal form. A *tableau* for Φ is a finite tree T in which every node is labelled either by a FOL formula or a set of formulas, and which has the following properties:

- The label of the root is Φ.
- If $\sigma \in T$ is a leaf, then there exists a predicate P and closed terms $\vec{t}$ such that the atomic formulas $P(\vec{t})$ and $\neg P(\vec{t})$ both occur as labels on the path σ. Leaves sometimes have an extra label $\bot$.
- If σ is not a leaf, then there is a formula φ on the path to σ such that one of the following holds:

  (α) φ is a conjunction $\varphi_1 \wedge \cdots \wedge \varphi_n$ and the unique successor of σ is labelled by the set of conjuncts $\{\varphi_i\}$. In this case, we say that *the rule applied in T at σ is the α rule.*

  (β) φ is a disjunction, and the children of σ are labelled by each of the disjuncts. Then we say that the β rule has been applied at σ.

  (γ) φ is a universal formula $\forall x.\psi(x)$ and the only child of σ is labelled by a *closed* instance $\psi(t)$ of its matrix. This is the γ rule.

  (δ) φ is an existential $\exists x.\psi(x)$ and the only child of σ is labelled by a *fresh* instance $\psi(a)$ of its matrix. Freshness means that a does not occur on the branch from the root to σ (thus a also does not occur in the formula Φ). This is the δ rule.

The observations leading to Definition 9 above are usually summarized by reference to a result known as *The Subformula Property.* This proposition states that for cut-free proofs, every formula occurring in a derivation is a subformula of one of the formulas in the end-sequent (the root of the derivation).

The subformula property is what put the word *analytic* into the proper name *Analytic Tableau.* It reflects the intuition of the tableau breaking down, or analysing, the formula according to its syntactic structure. The alternative name *Semantic Tableau* stresses the close connection between tableau rules and the meaning (semantics) of logical connectives. That both names are used interchangeably indicates yet again that Gentzen's sequent calculus is a natural, complete, syntactic theory of first-order logic.

Finally, we would like to ask the reader to take Definition 9 more as an informal description rather than a rigorous definition. For us, a tableau will be just a derivation of the sequent $\varphi \to \bot$ in LK* — the variant of LK consisting of the rules $\boxdot$L* and the rule Ax* — with φ a NNF. The trees of Definition 9 are merely a shorthand notation for such derivations.

Example 10. Let φ be the first-order tautology $(\exists x.Px \wedge Qx) \vee (\forall y.Py \to \neg Qy)$. To find a proof of φ using the tableau method, we must compute the negation normal form of $\neg\varphi$:

$$\mathsf{NNF}(\neg\varphi) = (\forall x.\neg Px \vee \neg Qx) \wedge (\exists y.Py \wedge Qy)$$

Now we repeatedly apply the rules α, β, γ, and δ, starting with the initial set $\{\mathsf{NNF}(\neg\varphi)\}$, until we can close every branch. This process is depicted below, where every inference is labelled with the tableau rule that was applied at the corresponding node.

$$
\alpha\,\dfrac{(\forall x.\neg Px \vee \neg Qx) \wedge (\exists y.Py \wedge Qy)}{\begin{array}{c}\forall x.\neg Px \vee \neg Qx\\ \delta\,\dfrac{\exists y.Py\wedge Qy}{\gamma\,\dfrac{Pa\wedge Qa}{\beta\,\dfrac{\neg Pa\vee\neg Qa}{\alpha\,\dfrac{\neg Pa}{\dfrac{\begin{array}{c}Pa\\Qa\end{array}}{\bot}} \quad \alpha\,\dfrac{\neg Qa}{\dfrac{\begin{array}{c}Pa\\Qa\end{array}}{\bot}}}}}\end{array}}
$$

This tableau directly corresponds to the LK* derivation[4] of the sequent $\mathsf{NNF}(\neg\varphi) \vdash \bot$ displayed in Figure 2.2, where the principal formula in the conclusion of each inference is underlined.

[4]Recall that LK* derivations are just LK derivations in which non-axiom rules are n-ary and the rule Ax* abbreviates one application of $\neg$L, a sequence of zero or more applications of WL, and one Axiom.

$$\begin{prooftree}
\AxiomC{}
\RightLabel{Ax*}
\UnaryInfC{$\left\{\begin{array}{l}(\forall x.\neg Px \lor \neg Qx) \land (\exists y.Py \land Qy),\\ \forall x.\neg Px \lor \neg Qx, \exists y.Py \land Qy,\\ Pa \land Qa, \neg Pa \lor \neg Qa,\\ \underline{\neg Pa}, \underline{Pa}, Qa\end{array}\right\} \vdash \bot$}
\RightLabel{$\land$L}
\UnaryInfC{$\left\{\begin{array}{l}(\forall x.\neg Px \lor \neg Qx) \land (\exists y.Py \land Qy),\\ \forall x.\neg Px \lor \neg Qx, \exists y.Py \land Qy,\\ \underline{Pa \land Qa}, \neg Pa \lor \neg Qa, \neg Pa\end{array}\right\} \vdash \bot$}
\AxiomC{}
\RightLabel{Ax*}
\UnaryInfC{$\left\{\begin{array}{l}(\forall x.\neg Px \lor \neg Qx) \land (\exists y.Py \land Qy),\\ \forall x.\neg Px \lor \neg Qx, \exists y.Py \land Qy,\\ Pa \land Qa, \neg Pa \lor \neg Qa,\\ \underline{\neg Qa}, Pa, \underline{Qa}\end{array}\right\} \vdash \bot$}
\RightLabel{$\land$L}
\UnaryInfC{$\left\{\begin{array}{l}(\forall x.\neg Px \lor \neg Qx) \land (\exists y.Py \land Qy),\\ \forall x.\neg Px \lor \neg Qx, \exists y.Py \land Qy,\\ \underline{Pa \land Qa}, \neg Pa \lor \neg Qa, \neg Qa\end{array}\right\} \vdash \bot$}
\RightLabel{$\lor$L}
\BinaryInfC{$\left\{\begin{array}{l}(\forall x.\neg Px \lor \neg Qx) \land (\exists y.Py \land Qy),\\ \forall x.\neg Px \lor \neg Qx, \exists y.Py \land Qy,\\ Pa \land Qa, \underline{\neg Pa \lor \neg Qa}\end{array}\right\} \vdash \bot$}
\RightLabel{$\forall$L}
\UnaryInfC{$\left\{\begin{array}{l}(\forall x.\neg Px \lor \neg Qx) \land (\exists y.Py \land Qy),\\ \underline{\forall x.\neg Px \lor \neg Qx}, \exists y.Py \land Qy,\\ Pa \land Qa\end{array}\right\} \vdash \bot$}
\RightLabel{$\exists$L}
\UnaryInfC{$\left\{\begin{array}{l}(\forall x.\neg Px \lor \neg Qx) \land (\exists y.Py \land Qy),\\ \forall x.\neg Px \lor \neg Qx, \underline{\exists y.Py \land Qy}\end{array}\right\} \vdash \bot$}
\RightLabel{$\land$L}
\UnaryInfC{$\left\{\ \underline{(\forall x.\neg Px \lor \neg Qx) \land (\exists y.Py \land Qy)}\ \right\} \vdash \bot$}
\end{prooftree}$$

Figure 2.2: An LK* proof of Example 10

2.5 Completeness of CL as a FOL proof procedure

We are now ready to justify the use of coherent logic to prove first-order tautologies. Recall Example 6:

$$\varphi \;=\; (\forall xy.Ax \lor By) \rightarrow (\forall x.Ax) \lor (\forall y.By)$$

Since this is a tautology, its negation must have a refutation. We have

$$\mathsf{NNF}(\neg\varphi) \;=\; (\forall xy.Ax \lor By) \land (\exists x.\neg Ax \land \exists y.\neg By),$$

and the complete tableau for the refutation of $\mathsf{NNF}(\neg\varphi)$ is

$$\begin{prooftree}
\AxiomC{$(\forall xy.Ax \lor By) \land (\exists x.\neg Ax \land \exists y.\neg By)$}
\RightLabel{α}
\UnaryInfC{$(\forall xy.Ax \lor By), (\exists x.\neg Ax \land \exists y.\neg By)$}
\RightLabel{α}
\UnaryInfC{$\exists x.\neg Ax, \exists y.\neg By$}
\RightLabel{δ}
\UnaryInfC{$\neg Aa$}
\RightLabel{δ}
\UnaryInfC{$\neg Bb$}
\RightLabel{γ}
\UnaryInfC{$Aa \lor Bb$}
\AxiomC{}
\noLine
\UnaryInfC{}
\end{prooftree}$$

$$\dfrac{Aa \lor Bb}{\dfrac{Aa}{\bot} \qquad \dfrac{Bb}{\bot}}\ \beta$$

Now compare the tree above with the tree in Figure 2.1 that was produced from the canonical translation of $\neg\varphi$. That the trees look so similar is not an accident: each node in the tableau labelled by the subformula $\psi(\vec{t})$ corresponds to a node in the coherent derivation with label $T_\psi(\vec{t})$. More importantly, the correspondence is not just between the *nodes* of the trees, but also the *rules* with which the nodes are derived. In particular, the principal formula(s) that were used in an application of one of the four tableau rules always corresponds with an instance of the fact(s) appearing on the left-hand side of the coherent clause that was used to derive the translated node.

The following proposition is a precise formulation of the statement: "To every closed tableau of a FOL tautology φ there corresponds a unique coherent derivation of $\bot$ from the canonical coherent theory of $\mathsf{NNF}(\varphi)$."

Theorem 11. *Let φ be a formula in negation normal form. The set of LK*-derivations of the sequent $\varphi \vdash \bot$ is in bijection with the set of (coherent) $\mathcal{T}_\varphi$-derivations of $\bot$ from $\varnothing$.*

Proof. Let $\mathcal{T}$ be the canonical translation of φ. Let π be a derivation in LK* of $\varphi \vdash \bot$. For $\Delta \to \bot$ a sequent in π, let Δ^t be the set $\{T_\psi(\vec{t})|\psi(\vec{t}) \in \Delta\}$. By induction on subderivations $\xi \subseteq \pi$, we construct a coherent derivation of $\Delta^t \vdash_{\mathcal{T}} \bot$, for every Δ in π.

- ξ is an application of the rule Ax*. Then there is a predicate P and closed terms $\vec{t}$ such that $P(\vec{t}), \neg P(\vec{t}) \in \Delta$. By Definition 4, $\mathcal{T}$ contains a clause $T_P(\vec{x}) \wedge F_P(\vec{x}) \to \bot$, while Δ^t contains the facts $T_P(\vec{t})$ and $F_P(\vec{t})$. Hence by the induction clause of CL-provability we have that $\Delta^t \vdash_{\mathcal{T}} \bot$, using $(\vec{t}/\vec{x})$ as the substitution.

- ξ ends in an application of the α rule. Then $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_n \in \Delta$, and the immediate subderivation of ξ is a derivation of the sequent $\Delta, \varphi_1, \cdots, \varphi_n \to \bot$. By the inductive hypothesis, we have that $\Delta^t, \varphi_1^t, \cdots, \varphi_n^t \vdash_{\mathcal{T}} \bot$. But $\varphi^t \to \varphi_1^t \wedge \cdots \wedge \varphi_n^t$ is a clause in $\mathcal{T}$ by construction. Hence $\mathcal{T}$ yields $\bot$ from Δ^t by coherent induction.

- The remaining clauses are treated similarly.

It therefore follows that $\{\varphi^t\} \vdash_{\mathcal{T}} \bot$ (since this corresponds to the last sequent in π.) But $\mathcal{T}$ contains the clause $\top \to \varphi^t$, and we have by one more application of CL induction that $\varnothing \vdash_{\mathcal{T}} \bot$. This completes the construction of the coherent derivation corresponding to π. The inverse transformation is symmetrical. □

Corollary 12. *Let φ be a first-order tautology. Let $\mathcal{T} = \mathcal{T}_{\mathsf{NNF}(\neg\varphi)}$. Then*

$$\vdash_{\mathcal{T}} \bot$$

Theorem 11 presents coherent provability as an integration of the four rules of the analytic tableau calculus into one — the rule of Definition 1. Exploiting this fact is the essence of the canonical translation.

The simplicity of this translation makes it very easy to implement it on a computer. The fact that composing it with a coherent prover yields a general first-order procedure which is equivalent to analytic tableau makes this idea especially appealing.

Example 13. Recall the formula $\varphi = (\exists x.Px \wedge Qx) \vee (\forall y.Py \to \neg Qy)$. In Example 10, we computed a proof of φ using the tableau method. Now we will show how the proof can be found automatically by feeding the formula to a CL prover through the translator. Below is the result of applying a simple implementation of the basic translation to

$$\mathsf{NNF}(\neg\varphi) = (\forall x.\neg Px \vee \neg Qx) \wedge (\exists y.Py \wedge Qy)$$

```
true => tAnd_47.

tQ(V1), fQ(V1) => false.
tP(V1), fP(V1) => false.

tAnd_47 =>              tForall_42, tExists_46.
tExists_46 =>           dom(Y), tAnd_44(Y).
tAnd_44(Y) =>           tP(Y), tQ(Y).
dom(X), tForall_42 =>   tOr_40(X).
tOr_40(X) =>            fP(X); fQ(X).
```

Feeding this theory to the GeologUI prover produces the tree shown in Figure 2.3. It is instructive to compare it with the tableau proof given at the end of the previous section.

As we see, the translation–prover tandem gives us a method of proving first-order tautologies. (Of course, it is only a semi-algorithm, because FOL provability is semidecidable.) The efficacy of the method depends on two things: the translator and the prover. The translator should be flexible and

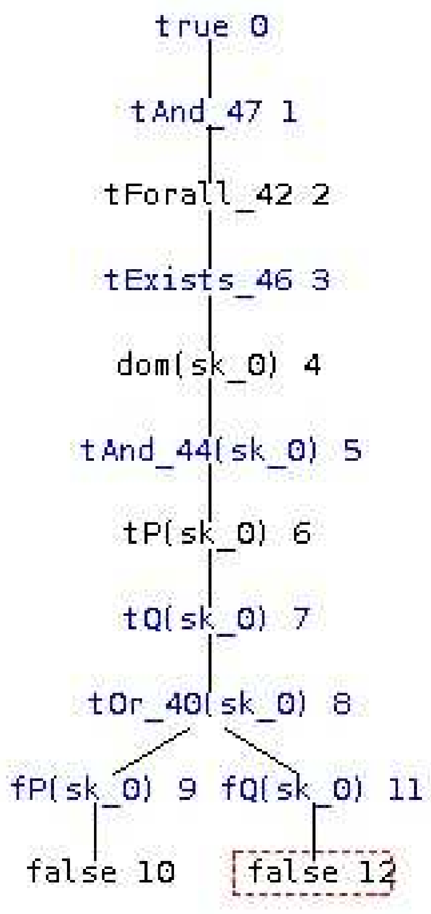


Figure 2.3: Automated proof of Example 10

efficient, and the prover powerful, in order for their combination to be useful. Developing a high-performance, efficient translation from FOL to CL is the primary subject of the Part I of this thesis. The present chapter concludes with a list of provers which are available for coherent logic. They have greatly assisted us during translator development.

2.6 The provers

The following provers can all loosely be said to implement Definition 1. All of them follow the depth-first strategy, possibly with minor extensions.

Prover	Author	Language	Distinguishing feature
CL	Marc Bezem	Prolog	Simple
GeologUI	John Fisher	Java	Visual
(Co)Colog	John Fisher	Java	Concurrent
Coherent	Stefan Berghofer	SML	Isabelle integration
Geo	Hans de Nivelle	C	High performance
Euclid	P. Janičić, S. Kordić	Prolog	First CL prover

We now give a more detailed description of each system.

1. **CL** — This is a short implementation by Marc Bezem whose main objective was to serve as a prototype for later implementations. Surprisingly however, its performance is quite good, considering that it is more or less a direct Prolog implementation of the forward search described above. It was successfully applied for large proof development, as reported by Bezem and Hendriks [2008]. The prover follows depth-first strategy, and uses Prolog native indexing routines. There is a backend that generates proof objects for Coq and Isabelle from the proof trace recorded by the prover. One recently added feature is a queueing mechanism which prevents the same existential rule from firing again until other applicable existential rules fire first.

2. **GeologUI** — This nice implementation by John Fisher features a truly beautiful graphical user interface to the prover. The user can select theories by dropping files into the window, execute the proof one step at a time, backtrack, set various parameters and heuristics, or run the prover for a fixed number of steps and have the result visually presented as a tree of facts. The core algorithm is identical with that of Bezem's prototype, but also implements the *earliest-first* fairness restriction, making the prover refutation-complete. This prover was very useful in the development of our translator, and we will often use its capabilities to display proofs. For example, the proof shown in Figure 2.3 was generated by **GeologUI**.

3. **(co)Colog** — CoColog is a command-line version of GeologUI in which

the core of the algorithm has been parallelized. The experiments with concurrency are ongoing, but the proof search has exhibited speed-up factors up to 2.5 on certain problems when executed on a quad-core machine Fisher [2009].

4. **Coherent** — This prover was written by Stefan Berghofer in the Standard ML language. The prover is implemented in the environment of the Isabelle system, an interactive prover based on Higher-Order Logic (HOL). The code uses datatypes native to Isabelle, and generates proof objects immediately available to the system. Such extensive integration between the prover and its environment can be said to accomplish to a significant extent the primary aim of the ACL project — to have a native first-order prover for interactive proof development. (The remaining part is the translation.)

5. **Geo** — Hans de Nivelle has written a high performance prover based on coherent logic which competes in CASC — the CADE ATP Systems Competition. It uses an algorithm which is different from those above, and relies most heavily on a particular kind of lemma learning introduced by De Nivelle and Meng [2006]. A peculiar feature of the preprocessing stage of the prover is the so-called antiskolemization procedure, that replaces all function symbols, including constants, by existential formulas. Still, **Geo** is a depth-first ground forward prover.

6. **Euclid** — This prover was implemented by the Serbian school (Janicic and Kordic [1995]) before coherent logic was defined in (Bezem and Coquand [2005]). It was designed to solve problems in elementary geometry. Later, many problems from geometry would be used as test examples for other coherent provers. Another idea implemented in **Euclid** which anticipates later work is the classification of input clauses according to whether they have branches, existential quantifiers, or unrestrained free variables.

Chapter 3

Improving the Translation

The purpose of this chapter is to serve as the bridge between the naïve translation introduced in the previous one and the full general translation developed in the next one. We reflect on the basic algorithm and suggest a few improvements. Some of these will motivate the core design of the advanced algorithm.

3.1 Immediate optimizations

Much of our work has been directed at improving the translation from FOL to CL in order to make it useful for proving FOL formulas using coherent provers. Soon we will see that this requires fundamental changes to the translation algorithm. However, we can already discuss some trivial improvements which have significant impact on performance.

1. The first is strengthening the NNF transformations to not only push negations down, but also collect the logical connectives together if their arguments have the same operator, like so:

Input	Output
$A \wedge ((B \wedge C) \wedge D)$	$A \wedge B \wedge C \wedge D$
$A \vee ((B \wedge C) \vee C)$	$A \vee (B \wedge C) \vee C$
$\forall x(\forall y(\forall z.\varphi))$	$\forall xyz.\varphi$
$\exists x_1 \cdots \exists x_n \varphi$	$\exists \vec{x}.\varphi$

   These rules are completely cosmetic and might be automatically subsumed by a particular representation of the formulas. For example, in the Isabelle prover, the two forms of quantified formulas denote the same objects — with "Input" being the inner representation and "Output" being the form displayed to the user — while the formula $A \wedge B \wedge C$ is just a notation for $(A \wedge B) \wedge C$, for which the needed transformation is obtained by adding to the rules in Definition 3 the rule of $\wedge$-associativity oriented to the left. (The same is done for $\vee$.)

However, these rules do have influence on the efficiency of proof search if one applies Definition 4 verbatim, because if $\forall x.\forall y.\varphi$ were to give rise to two clauses instead of one, the proof search would contain more facts, depleting computing resources quicker, and it would require more operations of matching/unification.

2. The second optimization consists of packing the clauses returned by the translator as much as possible into the coherent format.

   Generally, it is desirable to keep the number of clauses to a minimum, because every additional clause means more matching operations. For example, if the translation produces a theory containing the clauses

$$\begin{array}{rcl} T_{Pxa \vee (\exists y.Pxy \wedge Qy)}(x) & \rightarrow & T_P(x,a) \vee T_{\exists y.Pxy \wedge Qy}(x) \\ T_{\exists y.Pxy \wedge Qy}(x) & \rightarrow & \exists y.T_{Pxy \wedge Qy}(x,y) \\ T_{Pxy \wedge Qy}(x,y) & \rightarrow & T_P(x,y) \wedge T_Q(y) \end{array}$$

   then it is desirable to "reduce" these clauses to

$$T_{Pxa \vee (\exists y.Pxy \wedge Qy)}(x) \rightarrow T_P(x,a) \vee \exists y.(T_P(x,y) \wedge T_Q(y))$$

   which is a valid CL clause. This decreases the number of times a fact must be looked up in the current state, allowing the prover to construct the model quicker.

   The improvement in performance is particularly striking with regard to the packing of conjunctions into existential clauses. Whether the last two clauses in the above example are combined into one could in some cases make a difference in whether the depth-first search will terminate. This effect can be alleviated by adding the "backward" clause for the conjunction $T_P(x,y) \wedge T_Q(y) \rightarrow T_{Pxy \wedge Qy}(x,y)$, but this solution comes with a penalty: the number of facts gets much bigger, consequently the proofs get much longer.

3. After the translation is finished, the following rewrite rules should be applied to the resulting theory; it is a form of normalization, which performs partial proof before the actual prover is invoked. (Removing redundant clauses and subformulas.)

$$\begin{array}{rcll}
\top \wedge \varphi & \longrightarrow & \varphi & \\
\bot \wedge \varphi & \longrightarrow & \bot & \\
\top \vee \varphi & \longrightarrow & \top & \\
\bot \vee \varphi & \longrightarrow & \varphi & \\
\exists x.\top & \longrightarrow & \top & \\
\exists x.\bot & \longrightarrow & \bot & \\
\forall x.\top & \longrightarrow & \top & \\
\forall x.\bot & \longrightarrow & \bot & \\
P(\vec{t}) & \longrightarrow & \top & \qquad\qquad (\top \to P(\vec{x})) \in \mathcal{T} \\
P(\vec{t}) & \longrightarrow & \bot & \qquad\qquad (P(\vec{x}) \to \bot) \in \mathcal{T} \\
\varphi_1 \wedge \cdots \wedge \varphi_n & \longrightarrow & \bot & \qquad\qquad \exists i,j.\ \varphi_i = \neg\varphi_j \\
\varphi_1 \vee \cdots \vee \varphi_n & \longrightarrow & \top & \qquad\qquad \exists i,j.\ \varphi_i = \neg\varphi_j
\end{array}$$

In addition to the rules above, the translator should also discard useless clauses like $\top \to \top$ or $P(\vec{t}) \to P(\vec{t})$.

4. Finally, before invoking the prover, it is important to order the clauses of a coherent theory in the following manner:

   - Bottoming-out clauses (with empty right side) come first.
   - Then come clauses without disjunctions, existentials, or universals (free variable clauses).
   - Now disjunctions without existentials and universals.
   - Now free variables, still without existentials.
   - Then existentials *without disjunctions*.
   - Everything else.

It is possible that the simplifications above give rise to the empty clause

$$\top \to \bot$$

The process of refutation is thereby already completed — the theorem was proved by the translator, so to speak. We usually say in this case that “the translated theory has been simplified to nothing.”

Example 14. The *drinker's paradox* is the formula

$$\varphi = \exists x.(Px \to \forall y.Py)$$

In order to obtain the proof of this formula using coherent logic, we compute

$$\mathsf{NNF}(\neg\varphi) = \forall x.(Px \land \exists y.\neg Py)$$

The canonical translation of this formula is

$$\mathcal{T}_{\mathsf{NNF}(\neg\varphi)} = \begin{cases} (1) & \top & \to & T_{\forall x.(Px \land \exists y.\neg Py)} \\ (2) & T_{\forall x.(Px \land \exists y.\neg Py)} & \to & T_{Px \land \exists y.\neg Py}(x) \\ (3) & T_{Px \land \exists y.\neg Py}(x) & \to & T_P(x) \land T_{\exists y.\neg Py} \\ (4) & T_{\exists y.\neg Py} & \to & \exists y.F_P(y) \\ (5) & T_P(x) \land F_P(y) & \to & \bot \end{cases}$$

Denoting by $\rightsquigarrow$ one or more applications of the simplification rules above, we get the sequence of conversions in Figure 3.1.

As can be seen from the figure, the drinker's paradox simplifies to nothing.

Generally, it is best to leave the inference process to the prover, which is designed to carry it out as efficiently as possible. However, the simplifications suggested above significantly reduce the complexity of the translated theories. The results they produce are both more elegant and more efficient. So performing these transformations is a good investment of the prover time.

From now on, the examples we give will implicitly apply all of the improvements above to the result of Definition 4.

3.2 Shortcomings of the naïve translation

While the basic translation presented in the previous chapter is all nice and good for theoretical work, it is unfortunately of minimal practical utility. Applying the translation to simple formulas often results in coherent theories which cannot be refuted by a prover.

What is worse, applying the translation to theories which are *already* in coherent form (and easily refuted) can too produce unfeasible translations by making the theories far more complex. This is clear evidence that the translation is falling far short of having an "optimal" behavior one would wish for.

$$
\begin{aligned}
\mathcal{T}_{\mathsf{NNF}(\neg\varphi)} &\leadsto \left\{\begin{array}{lrcl}
(1^a) & \top & \to & \top \\
(2^a) & \top & \to & T_{Px\wedge\exists y.\neg Py}(x) \\
(3) & T_{Px\wedge\exists y.\neg Py}(x) & \to & T_P(x)\wedge T_{\exists y.\neg Py} \\
(4) & T_{\exists y.\neg Py} & \to & \exists y.F_P(y) \\
(5) & T_P(x)\wedge F_P(y) & \to & \bot
\end{array}\right. \\
&\leadsto \left\{\begin{array}{lrcl}
(2^b) & \top & \to & \top \\
(3^b) & \top & \to & T_P(x)\wedge T_{\exists y.\neg Py} \\
(4) & T_{\exists y.\neg Py} & \to & \exists y.F_P(y) \\
(5) & T_P(x)\wedge F_P(y) & \to & \bot
\end{array}\right. \\
&\leadsto \left\{\begin{array}{lrcl}
(3^c) & \top & \to & \top \\
(4^c) & \top & \to & \exists y.F_P(y) \\
(5^c) & \top\wedge F_P(y) & \to & \bot
\end{array}\right. \\
&\leadsto \left\{\begin{array}{lrcl}
(4^d) & \top & \to & \exists y.F_P(y) \\
(5^d) & F_P(y) & \to & \bot
\end{array}\right. \\
&\leadsto \left\{\begin{array}{lrcl}
(4^e) & \top & \to & \exists y.\bot \\
(5^e) & \bot & \to & \bot
\end{array}\right. \\
&\leadsto \left\{\begin{array}{lrcl}
(4^f) & \top & \to & \bot
\end{array}\right.
\end{aligned}
$$

Figure 3.1: Simplification of the Drinker's Paradox

Example 15. Consider the proposition that the diamond property (DP) is preserved under reflexive closure of a rewriting system. It can be formalized in coherent logic by the following theory DPE:

```
% start and finish
true => dom(a), dom(b), dom(c).
true => re(a,b), re(a,c).
re(b,X), re(c,X) => goal.

% equality axioms
dom(X) => e(X,X).
e(X,Y) => e(Y,X).
e(X,Y), re(Y,Z) => re(X,Z).

% basic facts on re
```

```
e(X,Y) => re(X,Y).
r(X,Y) => re(X,Y).
re(X,Y) => e(X,Y); r(X,Y).

% DP
r(X,Y), r(X,Z) => dom(U), r(Y,U), r(Z,U).
```

Here the predicates `r`, `re`, and `e` respectively represent the rewrite relation, its reflexive closure, and equality. It is easy to see that $\mathsf{DPE} \vdash$ `goal`: running a naïve implementation of the ground forward search yields a proof of `goal` in 35 steps, as shown in Figure 3.2 produced by GeologUI.

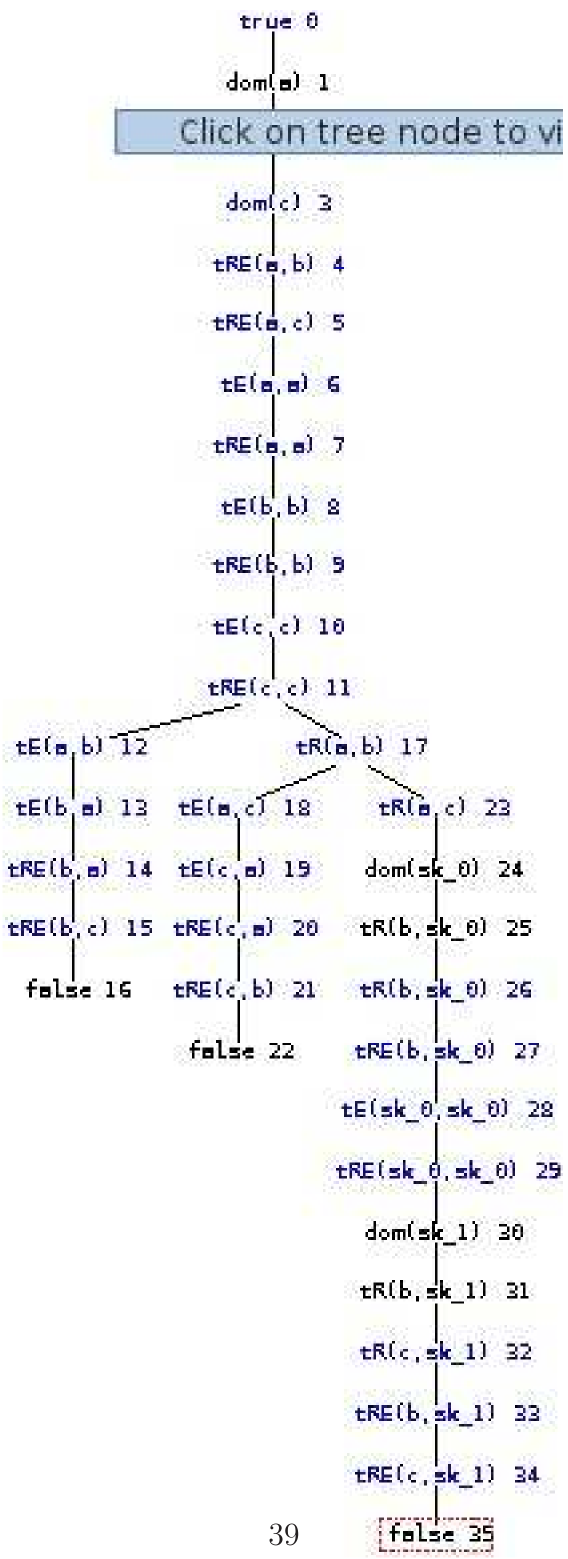


Figure 3.2: DPE

However, applying the canonical translation to the conjunction of the formulas in the theory above yields a set of clauses that is far more complicated:

```
tR(V1,V2),fR(V1,V2)   => false.
tE(V1,V2),fE(V1,V2)   => false.
tRE(V1,V2),fRE(V1,V2) => false.
tGOAL,fGOAL           => false.
true           => dom(a), dom(b), dom(c).
true           => tAnd_23.
tAnd_23                  => fGOAL, tRE(a,b), tRE(a,c), tAll_7,
   tAll_8, tAll_10, tAll_12, tAll_14, tAll_16, tAll_18, tAll_22.
dom(X), tAll_7           => tOr_6(X).
tOr_6(X)                 => fRE(b,X); fRE(c,X); tGOAL.
dom(X), tAll_8           => tE(X,X).
dom(X), dom(Y), tAll_10  => tOr_9(Y,X).
tOr_9(Y,X)               => fE(X,Y); tE(Y,X).
dom(X), dom(Y), dom(Z), tAll_12 => tOr_11(Y,X,Z).
tOr_11(Y,X,Z)            => fE(X,Y); fRE(Y,Z); tRE(X,Z).
dom(X), dom(Y), tAll_14  => tOr_13(X,Y).
tOr_13(X,Y)              => fE(X,Y); tRE(X,Y).
dom(X), dom(Y), tAll_16  => tOr_15(X,Y).
tOr_15(X,Y)              => fR(X,Y); tRE(X,Y).
dom(X), dom(Y), tAll_18  => tOr_17(X,Y).
tOr_17(X,Y)              => fRE(X,Y); tE(X,Y); tR(X,Y).
dom(X), dom(Y), dom(Z), tAll_22 => tOr_21(X,Y,Z).
tOr_21(X,Y,Z)            => fR(X,Y); fR(X,Z); tEx_20(Y,Z).
tEx_20(Y,Z)              => dom(U), tAnd_19(Y,Z,U).
tAnd_19(Y,Z,U)           => tR(Y,U), tR(Z,U).
```

Let's analyze the above example further. The output displayed is from the most direct implementation of Definition 4. When we also make use of the simplifications suggested in the previous section, we end up with a considerably leaner theory:

```
tR(V1,V2),fR(V1,V2)   => false.
tE(V1,V2),fE(V1,V2)   => false.
tRE(V1,V2),fRE(V1,V2) => false.
true          => dom(a), dom(b), dom(c).
true                     => tRE(a,b), tRE(a,c).
```

```
dom(X)                        => fRE(b,X); fRE(c,X).
dom(X)                        => tE(X,X).
dom(X), dom(Y)                => fE(X,Y); tE(Y,X).
dom(X), dom(Y), dom(Z)        => fE(X,Y); fRE(Y,Z); tRE(X,Z).
dom(X), dom(Y)                => fE(X,Y); tRE(X,Y).
dom(X), dom(Y)                => fR(X,Y); tRE(X,Y).
dom(X), dom(Y)                => fRE(X,Y); tE(X,Y); tR(X,Y).
dom(X), dom(Y), dom(Z)        => fR(X,Y); fR(X,Z);
                                   dom(U), tR(Y,U), tR(Z,U).
```

Still, when we invoke **CL** to refute the above set of clauses, the prover *chokes itself* — it generates so many facts and witnesses that the proof progress slows down to the point where there is no longer any hope it could finish the search within a reasonable amount of time.

Running the Prolog profiler on the prover while it is trying to refute the theory *without* simplifications shows that most of its time is spent in the `tOr_11` predicate. Inspecting the theory, we find that it occurs in the clauses

```
dom(X), dom(Y), dom(Z), tAll_12 => tOr_11(Y,X,Z).
tOr_11(Y,X,Z)                 => fE(X,Y); fRE(Y,Z); tRE(X,Z).
```

In the simplified theory, these clauses are reduced to

```
dom(X), dom(Y), dom(Z)        => fE(X,Y); fRE(Y,Z); tRE(X,Z).
```

Could this clause be responsible for the terrible behavior of the prover?

Let's inspect it closer. On the right side, we have a triple disjunction, and it contains three free variables. However, the left side has no atoms or predicates introduced by the translator. Hence the only hypotheses on the left side are the domain restriction predicates, leaving the free variables essentially unguarded. Thus the clause above is able to fire for any ground substitution of the variables $(\mathtt{X}, \mathtt{Y}, \mathtt{Z})$. (Notice that this is not a result of the simplification, because the fact `tAll_12` that appears to be "guarding" the clauses in the earlier version has no variables either and is actually asserted right after the "top" rule is fired.)

So *for every sequence of three elements* of the domain available at the current branch, the clause gives rise to three new branches. In turn, any of these which does not immediately yield a refutation will eventually arrive at the clause for the *next* sequence, and so on. We therefore have a tower of exponentials, of height $3^{3^{3\cdots}}$ No wonder the prover chokes on the problem.

Looking at the original theory, we see that the clause above corresponds to the input subformula

```
e(X,Y),re(Y,Z) => re(X,Z).
```

which has neither disjunctions nor unrestrained free variables. It is putting all of its predicates on the right side which introduces the new branches. Simultaneously, removing predicates guarding free variables from the left exacerbates the situation exponentially. The verdict is clear: movement of atoms into positive position has disastrous consequences for proof search.

One could blame NNF transformation for the replacement of implication with disjunction, but that would be a fig's leaf, because a good translator should try to find the best equivalent theory anyway. In particular, if the input contained disjunction, it should be translated into a theory which uses implication instead.

Nor should we try to use the particular syntactic structure of the input to help decide its possible translations. In fact, the optimal behavior of the translator should be due to its inner design logic rather than ad hoc checks such as "is this formula already coherent?"

At the same time, it is clear that moving those atoms from left to right was a bad idea. We should not have done it. So let's move them back.

In fact, this is exactly what we will do. But before we discuss the procedure in detail, let's pause and ask ourselves: what makes a good translator?

3.3 Criteria for a good translation

It is hardly possible to state precisely what a universally "good" translation will do. Therefore, we propose the following list of not-so-precise — but more useful — translator benchmarks. They should be read as intuitive aims rather than formal specifications.

1. Optimality — this property is simultaneously most decidedly desirable and least plausibly definable. (Of course, we could say that a translator is optimal if it generates theories running in the least number of steps on some benchmark prover. But then such a translator should output $\top \to \top$ if the theory is true and $\top \to \bot$ otherwise. Alas, first-order logic is undecidable.) One possible notion is that a translation is optimal if, among the possible translations it can return, it returns the one that would be most favored by a human.

2. Utilize the CL format to the maximum extent possible. The translation should try to fit the general FOL formula into coherent clauses as far as this is possible. Of course, this again cannot be given a precise sense, but two examples are: packing and existential anchoring.

3. Easy proof objects. The translation should not make use of any operations which are difficult to justify on the level of proof objects (such as Skolemization). Also, the proof objects should not exhibit superlinear blow-up as a result of reversing the translation steps.

4. The translated theory should not deviate too much from the logical structure of the input. In particular, if the prover finds a counter-model, the user should be able to comprehend it as a counter-example to his input formula. Why we satisfy this: because the introduced predicates in all cases correspond to a subformula of the input, or its NNF

5. Flexibility. The translator should be able to create many theories with different structure. The theories should create theories with very different prover behavior. Should have many different effects on proof search.

6. Idempotent. Applying translator to a translated theory should not make it more complicated or introduce any non-cosmetic changes.

7. Highly customizable, it should be easy to change behavior by a small change in a parameter. We want a translator which can be much better used to navigate the (hopefully) large space of possible translations.

8. Does not make coherent theories worse.

There is tension between some of the items above, especially 4 and 5. The solution is to rely on as few principles as possible, and maintain the close connection between the syntax of the FOL formula and the resulting coherent theory.

3.4 Improved translation

In this section we fully develop the mechanics of moving atoms back to the left. Although this idea is very simple, it has a very good effect on the translated theories, enough to earn it the status of the first major improvement to the translator. It also serves as the stepping stone to the general translation that is the subject of the next chapter.

Let's recall the setup from section 3.2. Suppose we are given a formula

$$\forall xy.P(x,y) \to Q(x,y) \tag{3.1}$$

where P is an atomic predicate. Its canonical translation is

$$\top \to F_P(x,y) \vee T_Q(x,y) \tag{3.2}$$

Although (3.1) is already a coherent formula and has no disjunctions, its translation (3.2) splits the proof search whenever it is applied. What's worse, clause (3.2) has what we call *unguarded doms*[1] — free variables which only occur on the right side of the clause.

Both problems would be fixed by allowing P to be moved to the left.

The naïve strategy could run as follows. Given a FOL formula φ, compute its canonical translation. For every clause which has a disjunct consisting of a single literal T_A (respectively F_A), remove this disjunct and add a conjunct on the left side of the clause with the opposite polarity F_A (respectively T_A).

However, it is not possible to blindly move *all* atomic predicates to the left. In the absense of compensatory clauses, this might cost us completeness. For an example, consider the tautology

$$(\exists xyz.(Pxyz \vee Q) \wedge (\neg Pxyz \vee Q)) \to Q \tag{3.3}$$

The canonical translation of its negation is

$$\begin{array}{ll}
\top & \to T_{\exists xyz.(Pxyz \vee Q)\wedge(\neg Pxyz \vee Q)} \wedge F_Q \\
T_{\exists xyz.(Pxyz \vee Q)\wedge(\neg Pxyz \vee Q)} & \to \exists xyz.T_{Pxyz\vee Q}(x,y,z) \wedge T_{\neg Pxyz \vee Q}(x,y,z) \\
T_{Pxyz\vee Q}(x,y,z) & \to T_P(x,y,z) \vee T_Q \\
T_{\neg Pxyz\vee Q}(x,y,z) & \to F_P(x,y,z) \vee T_Q \\
T_P(x,y,z) \wedge F_P(x,y,z) & \to \bot \\
T_Q \wedge F_Q & \to \bot
\end{array}$$

Running this theory with Geolog yields contradiction in 10 steps, as shown in Figure 3.3. However, if the disjunctive clauses are replaced by

[1] dom(x) is the range restriction predicate in Bezem's Prolog prover — it specifies which elements belong to the domain in the current search state.

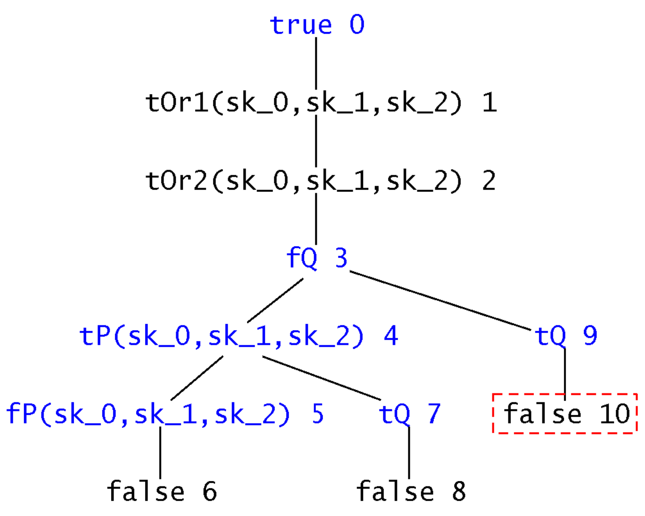


Figure 3.3: Translation and refutation of (3.3)

$$\begin{aligned} T_{Pxyz \vee Q}(x,y,z) \wedge F_P(x,y,z) \wedge F_Q &\to \bot \\ T_{\neg Pxyz \vee Q}(x,y,z) \wedge T_P(x,y,z) \wedge F_Q &\to \bot \end{aligned}$$

then we will not be able to derive $\bot$ with ground forward search. Neither of these two clauses can fire, because the theory has no clauses which could add facts with atoms T_P or F_P to the context. The countermodel produced by the proof search would interpret both $F_P(\vec{a})$ and $T_P(\vec{a})$ as false, which is not the intention of the translator.

The completeness can be restored by adding the clause

$$\top \to T_P(x,y,z) \vee F_P(x,y,z)$$

which is dual to the "bottoming out" clause appearing in the usual translation. Unfortunately, such clauses are extremely dangerous, because they have no atoms on the left side, all of the free variables are unguarded, and their number depends on the arity of the predicate. The presence of such a clause thus immediately gives rise to the situation of equation (3.2), with exponential explosion of branching. As our current goal is to avoid such situations, we cannot accept a solution requiring these clauses. We are forced to conclude that when atoms are moved to the left side, any atoms appearing in the opposite polarity must stay in place, since they may be required to make the new (contrapositive) clause fire.

To summarize, *for any predicate symbol occurring in a formula, there is at most one polarity with which literals containing the predicate may be moved to the left in the formula's translation.*

The improved translation algorithm runs as follows. Given a formula φ, it picks a polarity for every predicate symbol occurring in φ, computes the canonical translation of φ, and moves to the left all literal disjuncts of the appropriate polarity. We arrive at the following definition.

Definition 16. The *improved translation* gives for each formula φ in NNF a set of coherent theories $\{\mathcal{T}_\mathfrak{p}\}$ indexed by maps $\mathfrak{p} : \mathcal{A} \to \{T, F\}$, where $\mathcal{A}$ is the set of atomic predicates occurring in φ. For each such $\mathfrak{p}$, we define the theory $\mathcal{T}_\mathfrak{p}$ as follows.

Let $\mathcal{T}_0 = \mathcal{T}_\varphi$ be the basic translation of φ. For a literal $L \subseteq \mathsf{NNF}(\varphi)$, recall that L^t is the formula $T_A(\vec{t})$ when $L = A(\vec{t})$, and $F_A(\vec{t})$ when $L = \neg A(\vec{t})$. Accordingly, define

$$L^f = \begin{cases} F_A(\vec{t}) & L^t = T_A(\vec{t}) \\ T_A(\vec{t}) & L^t = F_A(\vec{t}) \end{cases}$$

and write $L \in \bar{\mathfrak{p}}$ if $L^f = \mathfrak{p}(A)_A(\vec{t})$. $\mathcal{T}_\mathfrak{p}$ is the set

$$\left\{ \bigwedge C_i \wedge \bigwedge L_k^f \to \bigvee\nolimits_{j \neq j_k} D_j \,\middle|\, (\bigwedge C_i \to \bigvee D_j) \in \mathcal{T}_0, L_k^t = D_{j_k}, L_k \in \bar{\mathfrak{p}} \right\}$$

The translation thus defined is easily automated. In the following examples, we use an implementation written in Common Lisp.

Example 17. Let's find the translation of the example we used in the discussion above.

$$(\exists xyz.(Pxyz \vee Q) \wedge (\neg Pxyz \vee Q)) \to Q$$

Our implementation of the canonical translation algorithm yields the theory $\mathcal{T}_0$ displayed below. (Compare with the listing two pages back.)

```
true => tOr1(X,Y,Z), tOr2(X,Y,Z), fQ.
tOr1(X,Y,Z) => tP(X,Y,Z); tQ.
tOr2(X,Y,Z) => fP(X,Y,Z); tQ.
tP(X,Y,Z), fP(X,Y,Z) => false.
tQ, fQ => false.
```

The set of possible assignments of polarities to predicate symbols is

$$\{(P,T),(Q,T)\},\{(P,T),(Q,F)\},\{(P,F),(Q,T)\},\{(P,F),(Q,F)\}$$

Corresponding to each of these possibilities, the improved algorithm generates the following theories.

```
true => tOr1(X,Y,Z), tOr2(X,Y,Z), fQ.
tOr1(X,Y,Z) => tP(X,Y,Z); tQ.
tOr2(X,Y,Z), tP(X,Y,Z) => tQ.
tP(X,Y,Z), fP(X,Y,Z) => false.
tQ, fQ => false.

true => tOr1(X,Y,Z), tOr2(X,Y,Z), fQ.
tOr1(X,Y,Z), fP(X,Y,Z) => tQ.
tOr2(X,Y,Z) => fP(X,Y,Z); tQ.
tP(X,Y,Z), fP(X,Y,Z) => false.
tQ, fQ => false.

true => tOr1(X,Y,Z), tOr2(X,Y,Z), fQ.
tOr1(X,Y,Z), fQ => tP(X,Y,Z).
tOr2(X,Y,Z), fQ, tP(X,Y,Z) => false.
tP(X,Y,Z), fP(X,Y,Z) => false.
tQ, fQ => false.

true => tOr1(X,Y,Z), tOr2(X,Y,Z), fQ.
tOr1(X,Y,Z), fQ, fP(X,Y,Z) => false.
tOr2(X,Y,Z), fQ => fP(X,Y,Z).
tP(X,Y,Z), fP(X,Y,Z) => false.
tQ, fQ => false.
```

The corresonding refutations are displayed in Figure 3.4.

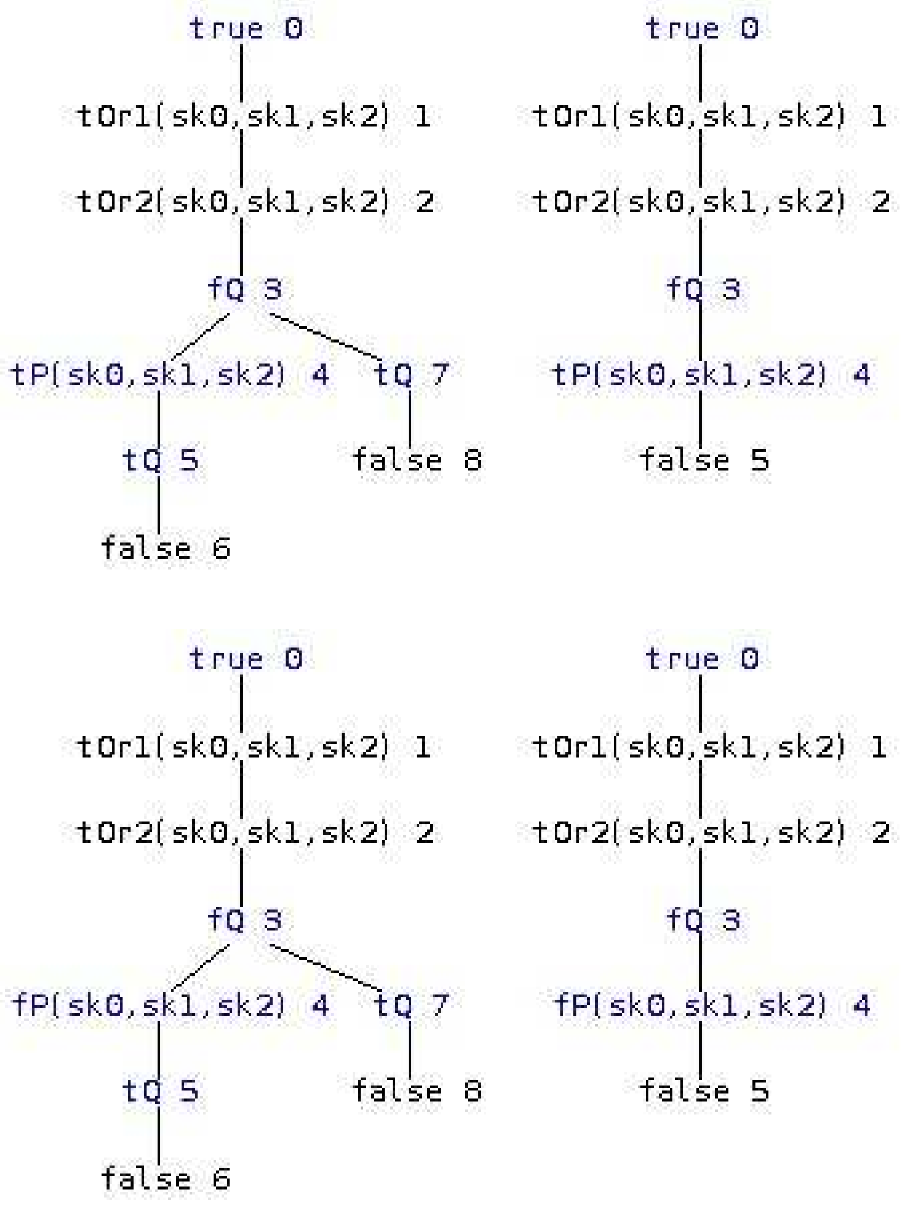


Figure 3.4: Four improved translations of Example 17

Note that none of the improved theories exactly equals the canonical translation — in the case where $\mathfrak{p}$ always picks the positive polarities, all of the negative literals are moved to the left.

It is immediate that the number of coherent theories produced by Definition 16 equals two raised to the number of atomic predicates in the formula. A practical algorithm therefore also has the job of choosing one of these.

A common theme in automated theorem proving is minimizing the number of branches, because this is almost always beneficial to proof search. Since each movement of a literal from right to left decreases the number of branches (disjuncts) in a theory, a natural solution to the problem above is to pick the polarity assignment $\mathfrak{p}$ that allows a maximal number of atoms to be moved. This is the choice our algorithm makes when asked for a single theory. (In the example above, the algorithm would pick $\mathfrak{p} = \{(P, T), (Q, F)\}$.)

The resulting translation thus satisfies the (first) criterion from the previous section "by construction". What about the others?

In order to measure how the translator handles theories which are already coherent, we go back to the problem example of section 3.2.

Example 18. Applying the improved translation to DPE yields the theory

```
tR(V1,V2), fR(V1,V2) => false.
tE(V1,V2), fE(V1,V2) => false.
tRE(V1,V2), fRE(V1,V2) => false.

true => dom(a), dom(b), dom(c).

tRE(b,A), tRE(c,A)  => false.
true                => tRE(a,b).
true                => tRE(a,c).
tR(A,B)             => tRE(A,B).
tE(A,B)             => tRE(A,B).
tE(A,B), tRE(B,C)   => tRE(A,C).
tE(A,B)             => tE(B,A).
dom(A), true        => tE(A,A).
tRE(A,B)            => tE(A,B); tR(A,B).
tR(A,B), tR(A,C)    => dom(D), tR(B,D), tR(C,D).
```

which is the same as the input, except that the reflexivity axiom has been moved lower and the predicate `goal` has been replaced with `false`. In fact, these predicates serve the same function.

So it appears that when a FOL theory is already coherent, its improved translation does not change much. Indeed, it is easy to see this, and more — that the translation is idempotent — by applying Definition 16 to a generic coherent clause

$$\forall \vec{x}.A_1 \wedge \cdots \wedge A_m \to B_1 \vee \cdots \vee B_n \tag{3.4}$$

with $B_j = \exists \vec{y}.C_1^j \wedge \cdots \wedge C_{l_j}^j$. This yields the theory

$$\begin{aligned}
\top &\to T_{\mathsf{NNF}(\forall \vec{x}.A_1 \wedge \cdots \wedge A_m \to B_1 \vee \cdots \vee B_n)} \\
T_{\mathsf{NNF}(\forall \vec{x}.A_1 \wedge \cdots \wedge A_m \to B_1 \vee \cdots \vee B_n)} &\to T_{\neg A_1 \vee \cdots \vee \neg A_m \vee B_1 \vee \cdots \vee B_n}(\vec{x}) \\
T_{\neg A_1 \vee \cdots \vee B_n}(\vec{x}) &\to F_{A_1}(\vec{x}) \vee \cdots \vee T_{B_n}(\vec{x}) \\
T_{B_1}(\vec{x}) &\to \exists \vec{y}.T_{C_1^1 \wedge \cdots \wedge C_{l_1}^1}(\vec{x},\vec{y}) \\
T_{C_1^1 \wedge \cdots \wedge C_{l_1}^1}(\vec{x},\vec{y}) &\to T_{C_1^1}(\vec{x},\vec{y}) \wedge \cdots \wedge T_{C_{l_1}^1}(\vec{x},\vec{y}) \\
&\vdots \\
T_{B_n}(\vec{x}) &\to \exists \vec{y}.T_{C_1^n \wedge \cdots \wedge C_{l_n}^n}(\vec{x},\vec{y}) \\
T_{C_1^n \wedge \cdots \wedge C_{l_n}^n}(\vec{x},\vec{y}) &\to T_{C_1^n}(\vec{x},\vec{y}) \wedge \cdots \wedge T_{C_{l_n}^n}(\vec{x},\vec{y})
\end{aligned}$$

The simplifier will remove the first two clauses. For each i, the pairs

$$\begin{aligned}
T_{B_i}(\vec{x}) &\to \exists \vec{y}.T_{C_1^i \wedge \cdots \wedge C_{l_i}^i}(\vec{x},\vec{y}) \\
T_{C_1^i \wedge \cdots \wedge C_{l_i}^i}(\vec{x},\vec{y}) &\to T_{C_1^i}(\vec{x},\vec{y}) \wedge \cdots \wedge T_{C_{l_i}^i}(\vec{x},\vec{y})
\end{aligned}$$

will be combined into

$$T_{B_i}(\vec{x}) \to \exists \vec{y}.T_{C_1^i}(\vec{x},\vec{y}) \wedge \cdots \wedge T_{C_{l_i}^i}(\vec{x},\vec{y})$$

Then all of these clauses will be packed back into the third clause, so that the only thing which remains of the set above is the clause

$$\begin{aligned}
\top \to F_{A_1}(\vec{x}) \vee \cdots \vee F_{A_m}(\vec{x}) \quad & (3.5) \\
\vee\ \exists \vec{y}.T_{C_1^1}(\vec{x},\vec{y}) \wedge \cdots \wedge T_{C_{l_1}^1}(\vec{x},\vec{y}) \quad & (3.6) \\
\vdots \quad & (3.7) \\
\vee\ \exists \vec{y}.T_{C_1^n}(\vec{x},\vec{y}) \wedge \cdots \wedge T_{C_{l_n}^n}(\vec{x},\vec{y}) \quad & (3.8)
\end{aligned}$$

Now F_{A_i} are all atoms; they will be moved to the left of the clause unless the rest of the input advises against it (by having more disjunctions of their complements T_{A_i}), which is not possible if (3.4) is part of a theory produced by the improved translation. Then (3.5) will be transformed into

$$\begin{aligned} T_{A_1}(\vec{x}) \vee \cdots \vee T_{A_m}(\vec{x}) \to \quad & \exists \vec{y}. T_{C_1^1}(\vec{x},\vec{y}) \wedge \cdots \wedge T_{C_{l_1}^1}(\vec{x},\vec{y}) \\ & \vee \exists \vec{y}. T_{C_1^2}(\vec{x},\vec{y}) \wedge \cdots \wedge T_{C_{l_2}^2}(\vec{x},\vec{y}) \\ & \qquad \vdots \\ & \vee \exists \vec{y}. T_{C_1^n}(\vec{x},\vec{y}) \wedge \cdots \wedge T_{C_{l_n}^n}(\vec{x},\vec{y}) \end{aligned}$$

which is obtained from (3.4) by renaming the atoms A_i, C_k^j into T_{A_i}, $T_{C_k^j}$.

If (3.4) is just a clause from some coherent theory, then some of the atoms may get their polarity reversed. The resulting theory will have fewer disjunctions. The clauses might also get reordered based on the heuristics from section 3.1. But the essential structure of the input theory will remain unchanged. Hence criteria (6) and (8) are both satisfied.

The previous discussion demonstrates that the behavior of the improved translation is optimal for Π_2-formulas: they are simply filled into the coherent format according to their logical structure. The theories thereby remain very near to the original formulas in logical meaning, and the proof objects of translation can be recovered effortlessly. Considering the simplicity of this translation, the result is quite satisfactory.

But it opens the door to the next question: what about formulas of higher complexity? Unfortunately, the improved translation does not scale to formulas with high syntax trees, precisely because it only concerns itself with atoms. In order to extend the improvement to more complex inputs it is necessary to be able to reverse the polarity of arbitrary subformulas, and not just the literals. Yet it is not immediate what this should even mean.

Another drawback is that the improved translation greedily picks the polarities which minimize the number of branches. This might be a good first-guess heuristic, but an advanced translator should be aware of other elements of logical structure. For example, if the cost of removing one branch is that three additional unguarded variables are introduced, is it worth to remove it? Ultimately we want to allow the user to influence how the translator decides these questions. That is, we want to develop a flexible system for fine-tuning translator behavior.

3.5 Hypertableau

We finish this chapter by sketching the completeness of the improved translation we defined. That is, if φ is a first-order tautology and $\mathfrak{p}$ is an assignment of polarities to atomic predicates, then $\bot$ can be derived from the theory $\mathcal{T}_{\mathfrak{p}}$ generated by the improved translation of $\neg\varphi$. The formal proof of this will be given in section 4.3, where we prove the completeness of the more general method of translation. Here we will present an informal argument.

Again, the completeness of the translation is most easily established by relating it to analytic tableau. It turns out that the idea of moving atoms to the left is the CL-equivalent of a well-known in the tableau community optimization technique bearing the name of *hypertableau*. The latter is the general observation that any application of a β rule can be delayed until every disjunct that is an atomic formula occurs on the current branch in the negative polarity. Its justification is very simple.

Suppose that π is a derivation in LK*, and let ι be an inference step that uses the β rule:

$$\frac{\Gamma, \varphi_1 \vdash \bot \qquad \cdot \quad \cdot \quad \cdot \qquad \Gamma, \varphi_n \vdash \bot}{\Gamma, \varphi_1 \vee \cdots \vee \varphi_n \vdash \bot}$$

Let ξ be the subderivation of π which has ι as the last inference. Then ι can be permuted with other inferences in ξ — without affecting the end-sequent — until its context Γ has the property that $\neg\varphi_i \in \Gamma$ for every atomic φ_i.

For consider the branch ξ_i ending in $\Gamma, \varphi_i \vdash \bot$, with φ_i an atom. The only rule of LK* which may use φ_i is the axiom rule Ax*. If such an inference indeed occurs in ξ_i then $\neg\varphi_i$ must be derivable from Γ alone — φ_i is not used by any rules before Ax*. This derivation of $\neg\varphi_i$ from Γ can be placed in ξ before ι. The branching of the proof is thus pushed further back toward the leaves.

Notice that no new steps are introduced into the proof. At the same time, some of the facts which were used to derive $\neg\varphi_i$ might also be used in closing the other branches. Thus there is a great potential for speed-up. Indeed, the provers that implement hypertableau (meaning that they fire β rules only when all atomic branches can be closed immediately) perform far better than vanilla ground tableau provers. Accordingly, CL provers perform better when the translation provides support for this optimization, which is exactly what the improved translation does.

When the clause

$$T_{P\vec{x} \vee (\neg Q\vec{x}) \vee \varphi}(\vec{x}) \to T_P(\vec{x}) \vee F_Q(\vec{x}) \vee T_\varphi(\vec{x})$$

is changed into

$$T_{P\vec{x} \vee (\neg Q\vec{x}) \vee \varphi}(\vec{x}) \wedge T_Q(\vec{x}) \to T_P(\vec{x}) \vee T_\varphi(\vec{x})$$

the use of the clause is blocked until the prover has derived an appropriate instance of $T_Q(\vec{x})$. But whereas a hypertableau prover is only able to move the literals into positive polarity, the improved translation can also generate the clause

$$T_{P\vec{x} \vee (\neg Q\vec{x}) \vee \varphi}(\vec{x}) \wedge F_P(\vec{x}) \wedge T_Q(\vec{x}) \to T_\varphi(\vec{x})$$

if it discovers that most of the theory's P-disjuncts are positive.

The hypertableau is thus obtained from the improved translation by always taking $\mathfrak{p} = \lambda p : \mathcal{A}.T$. This choice is consistent, but is completely independent of the formula's logical structure. Luckily, having the completeness hold for this single $\mathfrak{p}$, as provided for by Baumgartner et al. [1996], is sufficient to conclude it for all.

For suppose that $\mathcal{T}_\mathfrak{p}$ is one of the theories generated by the improved translation of φ. Let ψ be obtained from φ by replacing every predicate assigned negative polarity by the negation of a new predicate:

$$\psi \equiv \varphi[(P \mapsto \neg\overline{P}) \mid \mathfrak{p}(P) = F]$$

Then ψ is equisatisfiable with φ: a model $(\mathfrak{M}, [\![]\!])$ of either one extends to a model of the other by asserting $\vec{a} \in [\![P]\!] \iff \vec{a} \notin [\![\overline{P}]\!]$. Let $\mathcal{T}'_T$ be the improved translation of ψ which assigns all predicates the polarity T. Then $\mathcal{T}_\mathfrak{p}$ and $\mathcal{T}'_T$ are identical theories up to renaming of predicate symbols (F_P are replaced by $T_{\overline{P}}$). Consequently, derivations of $\perp$ from $\mathcal{T}_\mathfrak{p}$ stand in exact correspondence with derivations of $\perp$ from $\mathcal{T}'_T$; so whenever φ is the negation of a tautology, $\perp$ is derivable from $\mathcal{T}_\mathfrak{p}$.

To summarize the relationship between hypertableau and Definition 16, both implement the same optimization mechanism, but while hypertableau makes an arbitrary choice, the improved translation tries to make an informed decision.

Still, in order to obtain a general translation from FOL to CL which has superior performance for complex formulas, it is necessary to go beyond simple shuffling of atomic disjuncts. We need to be able to switch the flow of proof search at arbitrary levels of syntax, under arbitrary logical operators. In the section that follows, this need is fulfilled.

Chapter 4

General Translation

This chapter will cover all aspects of our final translation algorithm. The central idea consists of trying to extend in a natural way the mechanism of reversing the polarity to arbitrary subformulas.

This approach has been independently explored by John Fisher [2008] in his "fore-aft translation".

4.1 Bidirectional reasoning

The translation of the previous chapter could move more than just atoms.

For example, suppose that a translated theory contains the following set of clauses, corresponding to some subformula $\varphi = \forall x.(\forall y.Pxy) \vee (\exists z.Qxz)$.

$$\begin{aligned} T_\varphi &\to T_{\forall y.Pxy}(x) \vee \exists z.T_Q(x,z) \\ T_{\forall y.Pxy}(x) &\to T_P(x,y) \end{aligned}$$

This has one disjunction and two unguarded doms[1],one over the disjunction. The improved translation could produce the superior theory

$$\begin{aligned} T_\varphi &\to T_{\forall y.Pxy}(x) \vee \exists z.T_Q(x,z) \\ T_{\forall y.Pxy}(x) \wedge F_P(x,y) &\to \bot \end{aligned}$$

But this still has x as an unguarded dom in the first clause, and it's the best the improved translation can do. Yet we could avoid the free variable x if we

[1]Recall that by an "unguarded dom" we just mean a free variable that does not occur on the left side of the clause.

flipped the polarity of P not only at the atomic level of $T_P(x, y)$ but *before* the universal quantification takes place, like so

$$\begin{aligned} T_\varphi \wedge F_{\forall y.Pxy}(x) &\to \exists z.T_Q(x, z) \\ F_P(x, y) &\to F_{\forall y.Pxy}(x) \end{aligned}$$

In this case, we have a set of clauses which is trivially equivalent to the earlier one, but without the pesky free variable and without disjunction!

Of course, this solution was easy to find because Pxy is atomic. What would happen if instead the formula φ had the form $\varphi = \forall x.(\forall y.\psi) \vee (\exists z.Qxz)$, where ψ is an arbitrary formula in the variables x and y? In this case the two clauses above would no longer constitute a leaf of the translation, but would be put together with the clauses generated by translating ψ, the first of which would have the form

$$T_\psi(x, y) \to \cdots$$

depending on the top connective in ψ. But now the usefulness of the optimization above is put into question, because the second clause must — since ψ is still in positive polarity — have the form

$$\top \to F_{\forall y.\psi}(x) \vee T_\psi(x, y)$$

and this clause has a new disjunction with two free variables and no constraints on the left side whatsoever! So it's better to stick with the original version unless — the polarity of ψ could be reversed as well, so that the clause above would become

$$F_\psi(x, y) \to F_{\forall y.\psi}(y) \tag{4.1}$$

Accordingly, the theory generated by ψ should then begin with a clause of the form "$\cdots \to F_\psi(x, y)$".

And so the process could be continued as long as we are able to flip the polarities of each successive formula. This brings us to the following problem: For each logical connective, how do we dualize the translation clauses to represent subformulas standing in contrapositive relation?

We have just observed that this is very easy for universal quantifiers. Now let's investigate the case of disjunctions by continuing the example above with ψ of the form $Pxy \vee Qyx$.

Normally we would have the clauses

$$T_{\forall y.\psi}(y) \to T_\psi(x,y)$$
$$T_\psi(x,y) \to T_P(x,y) \vee T_Q(y,x)$$

But if the first clause was changed into (4.1), then the second must be adjusted accordingly by moving ψ to the right side:

$$\top \to T_P(x,y) \vee T_Q(y,x) \vee F_\psi(x,y)$$

And this will make everything worse unless we can prevent the extra disjunction and new unguarded doms by moving some disjunct(s) to the left, as our improved translation could do:

$$F_P(x,y) \to T_Q(y,x) \vee F_\psi(x,y)$$

This is indeed a much better outcome, but it depends on being able to move P to the left side. If the same formula occurs somewhere else in the input with P negated, then only one of them could be treated in this manner. (We might be able to flip Q instead, but Q might also be used elsewhere.)

To summarize, flipping the polarity of a disjunctive formula is only useful if we can also flip enough of the disjuncts to cover all of the free variables occurring in the formula. Since generally we cannot expect all of the disjuncts to be atomic, nor can we be sure that the atoms we wish to move can be moved without disturbing other parts of the translation, we again are in want of being able to reverse the polarity of arbitrary subformulas.

Before we treat the remaining cases of conjunctions and existentials, we can already observe the pattern which is emerging. It is most resembling of the tableau rules for *signed formulas*, which extend Definition 9 by clauses for negative occurrences. The complete set of rules is summarized in the following table:

Operator	Rule			Type
$\wedge$	$T[\varphi_1 \wedge \cdots \wedge \varphi_n]$	$\to$	$T[\varphi_1]\ ,\ \cdots\ ,\ T[\varphi_n]$	α
	$F[\varphi_1 \wedge \cdots \wedge \varphi_n]$	$\to$	$F[\varphi_1] \mid \cdots \mid F[\varphi_n]$	β
$\vee$	$T[\varphi_1 \vee \cdots \vee \varphi_n]$	$\to$	$T[\varphi_1] \mid \cdots \mid T[\varphi_n]$	β
	$F[\varphi_1 \vee \cdots \vee \varphi_n]$	$\to$	$F[\varphi_1]\ ,\ \cdots\ ,\ F[\varphi_n]$	α
$\forall$	$T[\forall \vec{x}.\varphi]$	$\to$	$T[\varphi](\vec{x})$	γ
	$F[\forall \vec{x}.\varphi]$	$\to$	$F[\varphi](\vec{b})$	δ
$\exists$	$T[\exists \vec{x}.\varphi]$	$\to$	$T[\varphi](\vec{b})$	δ
	$F[\exists \vec{x}.\varphi]$	$\to$	$F[\varphi](\vec{x})$	γ

The positive rules (concerning formulas with sign T) are exactly the ones described in Definition 9, corresponding to coherent clauses generated by the canonical translation of FOL formulas. The extension to the translator we seek to define now corresponds to the negative tableau rules, but *oriented backwards.*

For example, in the positive case of disjunction, we know that if the formula $\bigvee \varphi_i$ is true, then one of the disjuncts φ_i is true, hence the corresponding clause in the naïve translation. In the reverse case, to conclude that the disjunction is false, we must know that each of the disjuncts is false. Hence the clause

$$F_{\varphi_1} \wedge \cdots \wedge F_{\varphi_n} \to F_{\varphi_1 \vee \cdots \vee \varphi_n}$$

that we would get if we reversed the polarity of every disjunct φ_i. (The variables $\vec{x}$ in $F_{\varphi_i}(\vec{x})$ will be suppressed for now.) This corresponds to the negative rule for $\vee$ in Table 4.1, read from right to left.

It may be a little unexpected that the rule for $\forall$ does not exactly correspond to the reversal of the negative rule in the table. This is because flipping the arrow in the quantifier rules also flips the eigenvariable requirement: to conclude $F[\forall\vec{x}.\varphi]$, we don't need to have $F[\varphi](\vec{b})$ for fresh constants $\vec{b}$ — any sequence of closed terms would do. Freshness is rather a prerequisite for reversing the arrow in the *positive* case, since in order to conclude that $\forall\vec{x}.\varphi$ is true, we must be able to derive $\varphi(\vec{t})$ for arbitrary terms $\vec{t}$, so the reversed rule must be subject to the eigenvariable condition.

Let's now move on to the other connectives. The negative rule for conjunction says that if $\varphi = \bigwedge \varphi_i$ is false, then one of the φ_is is false, which can be written as the coherent clause

$$F_\varphi \to F_{\varphi_1} \vee \cdots \vee F_{\varphi_n}$$

Switching the direction of the arrow yields the clause

$$F_{\varphi_1} \vee \cdots \vee F_{\varphi_n} \to F_\varphi$$

that we need for the new translation. However, the coherent format does not admit a disjunction on the left side, which means that it has to be distributed over the clause arrow:

$$\begin{array}{ccc} F_{\varphi_1} & \to & F_\varphi \\ & \vdots & \\ F_{\varphi_n} & \to & F_\varphi \end{array}$$

and it is *this* set of clauses that must be added to the coherent theory.

Flipping the conjunctions is therefore less promising than flipping disjunctions — whereas in the latter case, in order to break even we must only reverse enough disjuncts to cover all of the free variables, conjunction requires *all* of its conjuncts to be reversed, or else there will immediately be a branching clause $\top \to T_{\varphi_i} \vee F_\varphi$ with empty left side, leaving all free variables unguarded. Still, in some cases reversing the conjunction might successfully remove a variable or two bound higher above.

The last connective which remains is the existential quantifier. Unfortunately, here we have a real problem. The reversed existential rule cannot be realized in coherent logic because of the dual eigenvariable condition, requiring the bound variable(s) to appear on the left side as fresh constants. But Definition 1 does not provide for any mechanism of left-sided freshness.

There are two ways to deal with this problem. The first is not to deal with it. The second is to circumvent it.

A good case can be made for the view that moving existential subformulas into negative positions is not desirable. The reason is that existentials are essential anchors of the formula's logical structure — they are responsible for introducing new elements into the Herbrand universe, which determines the domain of the partial model constructed by the prover at a given point of the search. Thus altering their behavior will result in strong deviation from the meaning of the formula.

In fact, for this very reason we will keep existentials in place in our final translation from FOL to CL. This is in line with the criterion from 3.3 stating that the translation should not perform dramatic modifications to the formula's logical structure.

Nevertheless, for the purposes of having a more complete correspondence between FOL and CL proofs, as well as a sense of conceptual closure, there is a partial solution to the problem of reversing the polarity of existential subformulas that was suggested to us by Jens Otten [2009] Although dualizing existentials is not possible directly within the rigid bounds of CL format, an equivalent effect can be achieved using a trick known as *Herbrandization.* This is an operation which is dual to Skolemization, and consists of replacing the universally quantified variables of a given formula by function symbols.

The method is most easily comprehended as the compound operation of first Skolemizing the existential subformula, and then reversing the polarity of the result. For example, suppose we wish to flip the formula $\varphi(\vec{x}) = \exists y.\psi(\vec{x}, y)$. Eliminating the existential quantifier via introduction of a Skolem function yields the new formula $\varphi^{\mathsf{Sk}}(\vec{x}) = \psi(\vec{x}, f_\varphi(\vec{x}))$. Watching the formula's translation into CL while this happens, one observes the clause $T_\varphi(\vec{x}) \to \exists y.T_\psi(\vec{x}, y)$ being transformed into

$$T_{\varphi^{\mathsf{Sk}}}(\vec{x}) \to T_\psi(\vec{x}, f_\varphi(\vec{x}))$$

And the contrapositive of this latter clause is

$$F_\psi(\vec{x}, f_\varphi(\vec{x})) \to F_{\varphi^{\mathsf{Sk}}}(\vec{x}) \tag{4.2}$$

Thus the effect of Skolemization on an existential clause is to replace the existential quantifier with a fact containing one more function (in place of the quantified variable) depending on all of the variables which are free in the corresponding formula. This reflects the fact that the constants introduced by firing the existential rule need only depend on the terms which instantiate the variables, and not necessarily on all of the terms on the given branch. In the tableau world, this observation answers the name of δ^+ *rule* and is viewed as a significant optimization of the Free Variable Tableau procedure.[2]

The intuition behind the above observation is the following. Essentially, the new function symbol simulates the freshness condition in the existential clause by defining a new domain element for every sequence of closed terms that the truth of the formula φ could depend on. Dually, the fact that the function f_φ in (4.2) is new means that the atom $F_\psi(\vec{t}, f_\varphi(\vec{t}))$ could only be derived by firing the (universal) γ rule at the point where $f_\varphi(\vec{t})$ first appears. Since this rule can be fired with any instance, the left side of (4.2) must then be satisfiable with any closed term in place of $f_\varphi(\vec{t})$. So the clause is sound: if $\psi(\vec{t}, f_\varphi(\vec{t}))$ can be refuted, then φ is false for $\vec{x} = \vec{t}$.

In this manner, clauses like (4.2) provide a mechanism of left-sided freshness that we need to reverse the existential rule. In LK, this freshness requirement appears as the eigenvariable condition in the *right* universal introduction rule. The fact that we are able to deal with it means that we can represent right-sided sequent rules as well, and are no longer constrained to

[2]The sensibility of this view is far from immediate — preventing a prover from doing work which is evidently useless is as much optimization as it is debugging.

The same can be said about the so-called δ^{+^+} rule, which "liberalizes" this handling of existentials further by allowing the same function symbol to be used for subformulas which are equal up to variable renaming. Prudent choice of representation of formulas should identify such subformulas automatically, and there are certainly no theoretical grounds for defining Skolemization on subformula occurrences rather than subformulas themselves.

the refutation-based approach of analytic tableau. That is, we can represent full, cut-free LK.

Indeed, the true nature of the "flipping" operation lies not in the negative tableau rules for signed formulas, but in the right-sided rules of the sequent calculus. For example, when the translation of the formula $\neg A \vee \neg B$ is changed from

$$T_{\neg A \vee \neg B} \rightarrow F_A \vee F_B$$

into

$$T_A \wedge T_B \rightarrow F_{\neg A \vee \neg B}$$

the achieved effect is that the last inference of the LK-derivation is transformed from

$$\frac{\neg A \vdash \bot \qquad \neg B \vdash \bot}{\mathsf{NNF}(\neg(A \wedge B)) \vdash \bot}$$

into

$$\frac{\top \vdash A \qquad \top \vdash B}{\top \vdash A \wedge B}$$

So *negative* translation allows us to represent *positive* reasoning!

This presents us with an opportunity which will be explored when we prove completeness of the final translation. For now, we work to put all of the pieces together into a definition. We quickly recap the reversed rules for each connective.

A disjunctive clause of the form

$$\varphi^t \rightarrow \varphi_1^t \vee \cdots \vee \varphi_n^t$$

is reversed by flipping any number of disjuncts and flipping the formula φ. For example, one possible clause is

$$\varphi_2^f \wedge \varphi_3^f \wedge \varphi_n^f \rightarrow \varphi^f \vee \varphi_1^t \vee \varphi_4^t \vee \cdots \vee \varphi_{n-1}^t$$

Here φ_i^f denotes $F_{\varphi_i}(\vec{x})$ if $\varphi_i = \varphi_i(\vec{x})$ is non-literal, and it denotes L^f from Definition 16 when $\varphi_i = L$. We immediately see that there are 2^n possibilities for the top clause when φ is put into negative polarity, each corresponding to some subset of the disjuncts.

Because reducing branching is almost always beneficial, these possibilities should be considered even when φ itself remains positive:

$$\varphi_2^f \wedge \varphi_3^f \wedge \varphi_n^f \wedge \varphi^t \to \varphi_1^t \vee \varphi_4^t \vee \cdots \vee \varphi_{n-1}^t$$

So the clause corresonding to φ is determined by the polarity of φ and a set $\mathfrak{f} \subseteq \{1, ..., n\}$ specifying which disjuncts are flipped. Each disjunct will in turn have two possibilities for its polarity, and we must choose the one consistent with $\mathfrak{f}$. If $i \in \mathfrak{f}$, then φ_i must be flipped, and the clause corresponding to φ_i should look like $\cdots \to \varphi_i^f$. Otherwise φ_i should be treated positively like in the basic translation, with its clause having the usual form $\varphi_i^t \to \cdots$.

To summarize, any disjunction $\varphi = \varphi_1 \vee \cdots \vee \varphi_n$ gives rise to two clauses for any given subset $\mathfrak{f}$ of $\{1, ..., n\}$:

$$\bigwedge_{i\in\mathfrak{f}} \varphi_i^f \wedge \varphi^t \to \bigvee_{i\notin\mathfrak{f}} \varphi_i^t \tag{$C_\mathfrak{f}^{\vee+}$}$$

$$\bigwedge_{i\in\mathfrak{f}} \varphi_i^f \to \varphi^f \vee \bigvee_{i\notin\mathfrak{f}} \varphi_i^t \tag{$C_\mathfrak{f}^{\vee-}$}$$

Similarly, each of the other connectives will generate two or more clauses, corresponding to whether the formula in question is reversed. Translations of the original formula are generated by putting these together in a consistent way.

The universal case is straightforward: when $\varphi = \forall\vec{x}.\psi$, the positive theories begin with

$$\varphi^t \to \psi^t \tag{$C^{\forall+}$}$$

while the negative ones begin with

$$\psi^f \to \varphi^f \tag{$C^{\forall-}$}$$

There is also a clause "in between", which represents the case when the polarity reversal is stopped at the universal node:

$$\varphi^t \wedge \psi^f \to \bot \tag{$C^{\forall\bot}$}$$

Generally, $(C^{\forall\bot})$ is inferior to $(C^{\forall-})$ when it comes to proof search, because the polarity of φ might as well be flipped higher up, where it could potentially eliminate some free variables. (Other things being equal, it is advisable to flip subformulas as high in the syntax tree as possible, so that branches appear closer to the leaves, where they can be closed quicker.) Still, we will keep $(C^{\forall\bot})$ in the translation for now — this keeps it general and also simplifies the completeness proof.

A few pages ago we claimed that when a conjunction is reversed it is best to flip all of its conjuncts. But in the positive case, it makes perfect sense to keep some conjuncts in the positive polarity and flip the others, translating for example the formula $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_n$ into

$$\begin{aligned} \varphi^t &\to \varphi_1^t \wedge \varphi_4^t \wedge \cdots \wedge \varphi_{n-1}^t \\ \varphi_2^f \wedge \varphi^t &\to \bot \\ \varphi_3^f \wedge \varphi^t &\to \bot \\ \varphi_n^f \wedge \varphi^t &\to \bot \end{aligned}$$

Since φ thereby gives rise to a *set* of clauses, we might as well split the one with conjunction into individual implications:

$$\begin{aligned} \varphi^t &\to \varphi_1^t \\ \varphi^t &\to \varphi_4^t \\ &\vdots \\ \varphi^t &\to \varphi_{n-1}^t \end{aligned}$$

This has almost no effect on the meaning of the theory — the normalizations from section 3.1 will pack everything back into a single clause anyway — but now we can express the sets generated by the positive rule concisely as follows. For $\mathfrak{f} \subseteq \{1, ..., n\}$, the positive conjunction is represented by the clauses

$$\{\varphi^t \wedge \varphi_i^f \to \bot \mid i \in \mathfrak{f}\} \cup \{\varphi^t \to \varphi_i^t \mid i \notin \mathfrak{f}\} \qquad (\mathcal{C}_{\mathfrak{f}}^{\wedge +})$$

By symmetry, the clauses

$$\{\varphi_i^f \to \varphi^f \mid i \in \mathfrak{f}\} \cup \{\top \to \varphi_i^t \vee \varphi^f \mid i \notin \mathfrak{f}\} \qquad (\mathcal{C}_{\mathfrak{f}}^{\wedge -})$$

represent the negative case. One can enforce the requirement of reversing all conjuncts in the negative case by only considering $\mathcal{C}_{\mathfrak{f}}^{\wedge -}$ when $\mathfrak{f}$ is the whole set $\{1, ..., n\}$. This restriction will not be incorporated into the definition however, because in some cases the extra flexibility offered by admitting all possibilities might be desirable. (Allowing clauses of the form $\top \to \psi^t \vee \varphi^f$ in one place might reduce their number elsewhere. Also, when the conjunction is high in the tree, it might not have any free variables; then such clauses are not too threatening, as they give rise to only one new branch each.)

As regards the existential formulas, we will not reverse their clauses here. Instead, we will add an extra clause when we wish to flip the matrix of such a formula. So in the positive case we just have the rule

$$\varphi^t \to \exists\vec{x}.\psi^t \tag{$C^{\exists+}$}$$

and when ψ is flipped we also add the bottom clause for ψ:

$$\psi^t \wedge \psi^f \to \bot \tag{$C^{\perp}$}$$

(Notice that this clause depends only on ψ.)

Of course, if we truly wanted to reverse the existentials, we would instead use the clause

$$\psi^f_{\mathsf{Sk}} \to \varphi^f \tag{$C^{\exists-}$}$$

where ψ^f_{Sk} is the formula $F_\psi(\vec{x}, \vec{f_\varphi}(\vec{x}))$.

The translation could be defined explicitly in terms of the sets of polarity choices made at the non-terminal nodes in the syntax tree. Instead, we will define the set of translated theories by induction. When stated in this form, the definition is easier to extend with other features or conditions. It also immediately suggests a recursive implementation algorithm.

Definition 19. Given a first-order formula Φ, its general translation is built up from sets of new predicate symbols, coherent clauses, and coherent theories. These are defined as follows:

Atoms: the fundamental particles of the translation.

- For every atomic predicate $A \in \Phi$, let T_A, F_A be new predicates of the same arity as A.
- For every non-literal subformula $\varphi \subseteq \Phi$, let T_φ, F_φ be new predicates of arity $|\mathsf{FV}(\varphi)|$.
- For every literal $L \subseteq \Phi$, define the formulas L^t, L^f by

L	L^t	L^f
$A(\vec{t})$	$T_A(\vec{t})$	$F_A(\vec{t})$
$\neg A(\vec{t})$	$F_A(\vec{t})$	$T_A(\vec{t})$

- For every non-literal subformula $\varphi \subseteq \Phi$ with $\mathsf{FV}(\varphi) = \vec{x}$, let φ^t, φ^f denote the formulas $T_\varphi(\vec{x})$, $F_\varphi(\vec{x})$.

Clauses: the compound molecules of the translation.

For every subformula $\varphi \subseteq \Phi$, we define (sets of) clauses $C^*(\varphi)$ depending on the top-level connective of φ:

- φ is a literal $L = (\neg)A(\vec{t})$. The clauses are

$$\begin{aligned} C^\top(\varphi) &: \quad \top \to T_A(\vec{x}) \vee F_A(\vec{x}) \\ C^\perp(\varphi) &: \quad T_A(\vec{x}) \wedge F_A(\vec{x}) \to \perp \end{aligned}$$

- $\varphi = \forall \vec{x}.\psi$. The clauses are

$$\begin{aligned} C^{\forall+}(\varphi) &: \quad \varphi^t \to \psi^t \\ C^{\forall\perp}(\varphi) &: \quad \varphi^t \wedge \psi^f \to \perp \\ C^{\forall-}(\varphi) &: \quad \psi^f \to \varphi^f \end{aligned}$$

- $\varphi = \exists \vec{x}.\psi$. The clauses are

$$\begin{aligned} C^{\exists+}(\varphi) &: \quad \varphi^t \to \exists \vec{x}.\psi^t \\ C^\perp(\psi) &: \quad \psi^t \wedge \psi^f \to \perp \end{aligned}$$

- $\varphi = \varphi_1 \vee \cdots \vee \varphi_n$. For $\mathfrak{f} \subseteq \{1, ..., n\}$, the clauses are

$$\begin{aligned} C_{\mathfrak{f}}^{\vee+}(\varphi) &: \quad \bigwedge_{i \in \mathfrak{f}} \varphi_i^f \wedge \varphi^t \to \bigvee_{i \notin \mathfrak{f}} \varphi_i^t \\ C_{\mathfrak{f}}^{\vee-}(\varphi) &: \quad \bigwedge_{i \in \mathfrak{f}} \varphi_i^f \to \varphi^f \vee \bigvee_{i \notin \mathfrak{f}} \varphi_i^t \end{aligned}$$

- $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_n$. For $\mathfrak{f} \subseteq \{1, ..., n\}$, we have *sets* of clauses

$$\begin{aligned} \mathcal{C}_{\mathfrak{f}}^{\wedge+}(\varphi) &: \quad \{\varphi^t \wedge \varphi_i^f \to \perp \mid i \in \mathfrak{f}\} \cup \{\varphi^t \to \varphi_i^t \mid i \notin \mathfrak{f}\} \\ \mathcal{C}_{\mathfrak{f}}^{\wedge-}(\varphi) &: \quad \{\varphi_i^f \to \varphi^f \mid i \in \mathfrak{f}\} \cup \{\top \to \varphi_i^t \vee \varphi^f \mid i \notin \mathfrak{f}\} \end{aligned}$$

Theories: the building blocks of the translation.

For every subformula φ, its general translation consists of two sets of coherent theories $\mathcal{G}(\varphi) = (\mathcal{P}(\varphi), \mathcal{N}(\varphi))$. They are defined by induction on φ:

- $\varphi = L$. Then

$$\mathcal{P}(\varphi) = \mathcal{N}(\varphi) = \{\{C^{\perp}(\varphi), C^{\top}(\varphi)\}\}$$

- $\varphi = \forall\vec{x}.\psi$. Then

$$\begin{aligned}\mathcal{P}(\varphi) =&\{\mathcal{T} \cup \{C^{\forall+}(\varphi)\} \mid \mathcal{T} \in \mathcal{P}(\psi)\} \quad \cup \\ &\{\mathcal{T} \cup \{C^{\forall\perp}(\varphi)\} \mid \mathcal{T} \in \mathcal{N}(\psi)\} \\ \mathcal{N}(\varphi) =&\{\mathcal{T} \cup \{C^{\forall-}(\varphi)\} \mid \mathcal{T} \in \mathcal{N}(\psi)\}\end{aligned}$$

- $\varphi = \exists\vec{x}.\psi$. Then

$$\begin{aligned}\mathcal{P}(\varphi) =& \{\mathcal{T} \cup \{C^{\exists+}(\varphi)\} \mid \mathcal{T} \in \mathcal{P}(\psi)\} \quad \cup \\ &\{\mathcal{T} \cup \{C^{\exists+}(\varphi), C^{\perp}(\psi)\} \mid \mathcal{T} \in \mathcal{N}(\psi)\} \\ \mathcal{N}(\varphi) =& \varnothing\end{aligned}$$

- $\varphi = \varphi_1 \vee \cdots \vee \varphi_n$. For $\mathfrak{f} \subseteq \{1, ..., n\}$, put

$$\mathcal{G}_i(\mathfrak{f}) = \begin{cases}\mathcal{P}(\varphi_i) & i \notin \mathfrak{f} \\ \mathcal{N}(\varphi_i) & i \in \mathfrak{f}\end{cases}$$

(Equivalently, one may write $\mathcal{G}_i(\mathfrak{f}) = \pi_{\chi_{\mathfrak{f}}(i)}\mathcal{G}(\varphi_i)$, where $\chi_{\mathfrak{f}}$ is the characteristic function of $\mathfrak{f}$ and π_j is pair projection.) Then

$$\begin{aligned}\mathcal{P}(\varphi) &= \{\mathcal{T}_1 \cup \cdots \cup \mathcal{T}_n \cup \{C_{\mathfrak{f}}^{\vee+}(\varphi)\} \mid \mathfrak{f} \subseteq \{1, ..., n\},\ \mathcal{T}_i \in \mathcal{G}_i(\mathfrak{f})\} \\ \mathcal{N}(\varphi) &= \{\mathcal{T}_1 \cup \cdots \cup \mathcal{T}_n \cup \{C_{\mathfrak{f}}^{\vee-}(\varphi)\} \mid \mathfrak{f} \subseteq \{1, ..., n\},\ \mathcal{T}_i \in \mathcal{G}_i(\mathfrak{f})\}\end{aligned}$$

- $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_n$. For $\mathfrak{f} \subseteq \{1, ..., n\}$, let $\mathcal{G}_i(\mathfrak{f})$ be defined as in the previous clause. Then

$$\begin{aligned}\mathcal{P}(\varphi) &= \{\mathcal{T}_1 \cup \cdots \cup \mathcal{T}_n \cup \mathcal{C}_{\mathfrak{f}}^{\wedge+}(\varphi) \mid \mathfrak{f} \subseteq \{1, ..., n\},\ \mathcal{T}_i \in \mathcal{G}_i(\mathfrak{f})\} \\ \mathcal{N}(\varphi) &= \{\mathcal{T}_1 \cup \cdots \cup \mathcal{T}_n \cup \mathcal{C}_{\mathfrak{f}}^{\wedge-}(\varphi) \mid \mathfrak{f} \subseteq \{1, ..., n\},\ \mathcal{T}_i \in \mathcal{G}_i(\mathfrak{f})\}\end{aligned}$$

The Translation: its completed edifice. We let *bidirectional translation* of Φ be the set

$$\mathcal{P}(\Phi) = \pi_0\mathcal{G}(\Phi)$$

Example 20. To give the reader a feel for the theories produced by the bidirectional translation, we go back to Example 13, which concerned the formula

$$\Phi = (\forall x.\neg Px \vee \neg Qx) \wedge (\exists y.Py \wedge Qy)$$

If we feed this formula to the bidirectional translation, it produces a set of 128 theories, not counting members of $\mathcal{N}(\Phi)$, of which there are as many. It would be a waste of paper to display all of them here, but a selection of some is given in Figure 4.1. All of the theories displayed must be augmented by the clauses

```
true => tAnd_47.
tQ(V1), fQ(V1) => false.
tP(V1), fP(V1) => false.
true => tQ(V1); fQ(V1).
true => tP(V1); fP(V1).
```

When expanded with the above, all groups of clauses appearing in the figure are actual members of $\mathcal{P}(\Phi)$.

```
tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y), fAnd_44(Y) => false.
true =>                 fAnd_44(Y); tP(Y).
fQ(Y) =>                fAnd_44(Y).
tAnd_47 =>              tForall_42.
dom(X), tForall_42, fOr_40(X) => false.
true => fOr_40(X); fP(X); fQ(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y), fAnd_44(Y) => false.
true =>                 fAnd_44(Y); tP(Y).
fQ(Y) =>                fAnd_44(Y).
tAnd_47 =>              tForall_42.
dom(X), tForall_42, fOr_40(X) => false.
tP(X) => fOr_40(X); fQ(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y), fAnd_44(Y) => false.
true =>                 fAnd_44(Y); tP(Y).
fQ(Y) =>                fAnd_44(Y).
tAnd_47 =>              tForall_42.
dom(X), tForall_42, fOr_40(X) => false.
tP(X), tQ(X) => fOr_40(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y), fAnd_44(Y) => false.
true =>                 fAnd_44(Y); tP(Y).
fQ(Y) =>                fAnd_44(Y).
tAnd_47 =>              tForall_42.
dom(X), tForall_42, fOr_40(X) => false.
tQ(X) => fOr_40(X); fP(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y) =>  tP(Y).
tAnd_44(Y) =>  tQ(Y).
tAnd_47, fForall_42 =>  false.
fOr_40(X) =>    fForall_42.
true => fOr_40(X); fP(X); fQ(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y) =>  tP(Y).
tAnd_44(Y) =>  tQ(Y).
tAnd_47, fForall_42 =>  false.
fOr_40(X) =>    fForall_42.
tP(X) => fOr_40(X); fQ(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y) =>  tP(Y).
tAnd_44(Y) =>  tQ(Y).
tAnd_47, fForall_42 =>  false.
fOr_40(X) =>    fForall_42.
tP(X), tQ(X) => fOr_40(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y) =>  tP(Y).
tAnd_44(Y) =>  tQ(Y).
tAnd_47, fForall_42 =>  false.
fOr_40(X) =>    fForall_42.
tQ(X) => fOr_40(X); fP(X).
```

```
tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y) =>  tP(Y).
tAnd_44(Y) =>  tQ(Y).
tAnd_47, fForall_42 =>  false.
dom(X) =>               fForall_42; tOr_40(X).
tOr_40(X) =>      fP(X); fQ(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y) =>  tP(Y).
tAnd_44(Y) =>  tQ(Y).
tAnd_47, fForall_42 =>  false.
dom(X) =>               fForall_42; tOr_40(X).
tOr_40(X), tP(X) =>   fQ(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y) =>  tP(Y).
tAnd_44(Y) =>  tQ(Y).
tAnd_47, fForall_42 =>  false.
dom(X) =>               fForall_42; tOr_40(X).
tOr_40(X), tP(X), tQ(X) => false.

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y) =>  tP(Y).
tAnd_44(Y) =>  tQ(Y).
tAnd_47, fForall_42 =>  false.
dom(X) =>               fForall_42; tOr_40(X).
tOr_40(X), tQ(X) =>   fP(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y), fAnd_44(Y) => false.
true =>                 fAnd_44(Y); tP(Y).
fQ(Y) =>                fAnd_44(Y).
tAnd_47, fForall_42 =>  false.
dom(X) =>               fForall_42; tOr_40(X).
tOr_40(X) =>      fP(X); fQ(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y), fAnd_44(Y) => false.
true =>                 fAnd_44(Y); tP(Y).
fQ(Y) =>                fAnd_44(Y).
tAnd_47, fForall_42 =>  false.
dom(X) =>               fForall_42; tOr_40(X).
tOr_40(X), tP(X) =>   fQ(X).

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y), fAnd_44(Y) => false.
true =>                 fAnd_44(Y); tP(Y).
fQ(Y) =>                fAnd_44(Y).
tAnd_47, fForall_42 =>  false.
dom(X) =>               fForall_42; tOr_40(X).
tOr_40(X), tP(X), tQ(X) => false.

tAnd_47 =>              tExists_46.
tExists_46 =>  dom(Y), tAnd_44(Y).
tAnd_44(Y), fAnd_44(Y) => false.
true =>                 fAnd_44(Y); tP(Y).
fQ(Y) =>                fAnd_44(Y).
tAnd_47, fForall_42 =>  false.
dom(X) =>               fForall_42; tOr_40(X).
tOr_40(X), tQ(X) =>   fP(X).
```

Figure 4.1: Some translations of the drinker's paradox

The example above might already be disconcerning to the reader: if such a trivial formula produces such an immense number of theories, how will the translator perform when faced with a medium-sized problem? Indeed, the final refinements of the algorithm will deal precisely with this issue. For while the improved translation might return exponential number of theories in the worst case, bidirectional translation will necessarily split at every subformula. So it's rather imperative to find some way of controlling the exponential explosion of theories. This will be achieved in the next section.

We conclude with a few comments on how the translation thus proposed influences proof search. The most curious effect of reversing the rules (as in the above example) on the structure of coherent proofs is the following. Whereas in the basic and improved translations the syntactic depth of subformulas always increases downwards to the leaves (atomic formulas) — with the firing of a non-literal clause always generating nodes corresponding to the formula's immediate subformulas — in the translation presented here firing clauses of non-literal subformulas can affect the state of subformulas that contain them. That is, the construction of the model proceeds not just downwards from the root of the formula, but *upwards* from the leaves as well. Furthermore, these two regimes of proof search are mixed together without any awareness by the prover. It is for this reason that the translation described in Definition 19 was given the name *Bidirectional*: in a very literal sense, it allows the proof search to follow a pattern of fact inference which goes both up and down in the complexity of the formula.

This by itself is not a new idea: the *Handbook of Automated Reasoning* (**?**) devotes a whole chapter to the so-called 'Inverse Method', characterized by proof search that proceeds from axioms to the goal rather than vice versa. But the manner in which Bidirectional translation determines a hybrid proof search *before* envocation of the prover is, to our knowledge, new.

4.2 Atomic constraints and parametric optimization

In this section we develop a powerful method for controlling both the number and form of translated theories in a very general way. But first we extend Definition 19 with a simple amendment which weeds out a good bunch of useless elements of $\mathcal{P}(\Phi)$.

The motivation is as follows. One certainly wants the advanced translation to be at least as good as the improved translation from the last chapter. The latter operates by moving literals to the left as much as possible while keeping the top clause $C^\top$ redundant. But Definition 19 both allows disjuncts to stay positive after they have already been moved elsewhere, and sometimes moves literals to the left in both positive and negative polarity, necessitating inclusion of the branching clause $C^\top$. Applying the improved translation to such theories would make them better. At the same time, the theories thus produced are already members of $\mathcal{P}(\Phi)$. The goal is to eliminate all of the ones which "reduce" to them by moving atoms to the left, and the ones which need the top clause.

A natural way to accomplish this goal is to incorporate the mechanism of improved translation into Definition 19 by associating to each theory the choice of atomic polarities that its clauses must respect. That is, each theory $\mathcal{T} \in \mathcal{G}(\varphi)$ should carry with itself a set of *atomic constraints* $\mathfrak{p}(\mathcal{T})$ mapping predicates of φ to polarities $\{T, F\}$. Because they concern polarities of atoms, the constraints should be introduced in the base (literal) case of the translation.

In short, when the translation is applied to a literal L equal to $A(\vec{t})$, the returned theories $\mathcal{P}$ and $\mathcal{N}$ both consist of the bottom clause, but the first also has the constraint $\mathfrak{p} = \{(A, T)\}$ while the second has $\mathfrak{p} = \{(A, F)\}$. When $L = \neg A(\vec{t})$, the constraints are flipped. (This implies that even in the positive case we require each negative occurrence of an atom to be on the left side of the clause, with the only exception of existentials — which produce clauses of the form $T_{\exists x.Px} \to T_P(x)$ even when $(P, F) \in \mathfrak{p}(\mathcal{T})$. Still, the fact $T_P(c)$ can only be used by the bottom clause, which immediately closes the current branch.)

The constraints are then propagated and collected as the translation is computed bottom-up. In the branching clauses, whenever theories are combined for a given $\mathfrak{f}$, we must make sure that their constraints are consistent and take their union. This guarantees that in the final set of translations any given theory will have literals on the left in at most one of the two polarities, rendering the clause $C^\top$ extraneous.

Since we are now interested in making the translated theories as few and as efficient as possible, we will also use this opportunity to get rid of the "linking clause" $C^{\forall\bot}(\varphi)$, since it does not contribute any practical value to the set of theories produced.

It is important to take notice that every coherent theory returned by the definition below does also occur among those returned by the previous version. Hence it suffices to prove soundness and completeness for the bidirectional translation.

Notation. In the following definition, a *theory* is a pair $\mathcal{T} = (\mathcal{C}, \mathfrak{p})$ where $\mathcal{C}$ is a set of coherent clauses and $\mathfrak{p}$ is a partial map from the set of predicate symbols $\mathcal{A}$ to the set $\{T, F\}$. Given $\mathfrak{p}_i : \mathcal{A} \rightharpoonup \{T, F\}$ for $1 \leqslant i \leqslant m$, let $\bigvee \mathfrak{p}_i$ be defined as $\mathfrak{p}_1 \cup \cdots \cup \mathfrak{p}_m$ whenever this union is a (single-valued) function. Write $\bigvee \mathfrak{p}_i{\downarrow}$ if $\bigvee \mathfrak{p}_i$ is defined.

Definition 21. The *Bidirectional Translation with Atomic Constraints* gives, for a given FOL formula Φ, two families of theories $\mathcal{G} = (\mathcal{P}, \mathcal{N})$. The atoms and clauses are defined as in Definition 19. The theories are defined by induction on $\varphi \subseteq \Phi$.

- $\varphi = L$. Let the constraints $\mathfrak{p}^+(L)$ and $\mathfrak{p}^-(L)$ be given by

L	$\mathfrak{p}^+(L)$	$\mathfrak{p}^-(L)$
$A(\vec{t})$	$\{(A, T)\}$	$\{(A, F)\}$
$\neg A(\vec{t})$	$\{(A, F)\}$	$\{(A, T)\}$

  Then

$$\begin{aligned} \mathcal{P}(\varphi) &= \{(\{C^{\perp}(\varphi)\}, \mathfrak{p}^+(L))\} \\ \mathcal{N}(\varphi) &= \{(\{C^{\perp}(\varphi)\}, \mathfrak{p}^-(L))\} \end{aligned}$$

- $\varphi = \forall \vec{x}.\psi$. Then

$$\begin{aligned} \mathcal{P}(\varphi) &= \{(\mathcal{C} \cup \{C^{\forall +}(\varphi)\}, \mathfrak{p}) \mid (\mathcal{C}, \mathfrak{p}) \in \mathcal{P}(\psi)\} \\ \mathcal{N}(\varphi) &= \{(\mathcal{C} \cup \{C^{\forall -}(\varphi)\}, \mathfrak{p}) \mid (\mathcal{C}, \mathfrak{p}) \in \mathcal{N}(\psi)\} \end{aligned}$$

- $\varphi = \exists \vec{x}.\psi$. Then

$$\begin{aligned} \mathcal{P}(\varphi) = {} & \{(\mathcal{C} \cup \{C^{\exists +}(\varphi)\}, \mathfrak{p}) \mid (\mathcal{C}, \mathfrak{p}) \in \mathcal{P}(\psi)\} \quad \cup \\ & \{(\mathcal{C} \cup \{C^{\exists +}(\varphi), C^{\perp}(\psi)\}, \mathfrak{p}) \mid (\mathcal{C}, \mathfrak{p}) \in \mathcal{N}(\psi)\} \\ \mathcal{N}(\varphi) = {} & \varnothing \end{aligned}$$

- $\varphi = \varphi_1 \vee \cdots \vee \varphi_n$. For $\mathfrak{f} \subseteq \{1, ..., n\}$, put

$$\mathcal{G}_i(\mathfrak{f}) = \begin{cases} \mathcal{P}(\varphi_i) & i \notin \mathfrak{f} \\ \mathcal{N}(\varphi_i) & i \in \mathfrak{f} \end{cases}$$

  Then

$$\mathcal{P}(\varphi) = \{(\bigcup \mathcal{C}_i \cup \{C_{\mathfrak{f}}^{\vee +}(\varphi)\}, \bigvee \mathfrak{p}_i) \mid \mathfrak{f} \subseteq \{1, ..., n\}, (\mathcal{C}_i, \mathfrak{p}_i) \in \mathcal{G}_i(\mathfrak{f}), \bigvee \mathfrak{p}_i\downarrow\}$$
$$\mathcal{N}(\varphi) = \{(\bigcup \mathcal{C}_i \cup \{C_{\mathfrak{f}}^{\vee -}(\varphi)\}, \bigvee \mathfrak{p}_i) \mid \mathfrak{f} \subseteq \{1, ..., n\}, (\mathcal{C}_i, \mathfrak{p}_i) \in \mathcal{G}_i(\mathfrak{f}), \bigvee \mathfrak{p}_i\downarrow\}$$

- $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_n$. For $\mathfrak{f} \subseteq \{1, ..., n\}$, let $\mathcal{G}_i(\mathfrak{f})$ be defined as in the previous clause. Then

$$\mathcal{P}(\varphi) = \{(\bigcup \mathcal{C}_i \cup \mathcal{C}_{\mathfrak{f}}^{\wedge +}(\varphi), \bigvee \mathfrak{p}_i) \mid \mathfrak{f} \subseteq \{1, ..., n\},\ (\mathcal{C}_i, \mathfrak{p}_i) \in \mathcal{G}_i(\mathfrak{f}), \bigvee \mathfrak{p}_i\downarrow\}$$
$$\mathcal{N}(\varphi) = \{(\bigcup \mathcal{C}_i \cup \mathcal{C}_{\mathfrak{f}}^{\wedge -}(\varphi), \bigvee \mathfrak{p}_i) \mid \mathfrak{f} \subseteq \{1, ..., n\},\ (\mathcal{C}_i, \mathfrak{p}_i) \in \mathcal{G}_i(\mathfrak{f}), \bigvee \mathfrak{p}_i\downarrow\}$$

We will not distract the reader with any examples at this point, since we're about to give the final definition which will make use of the atomic constraints as well.

The primary difficulty in choosing the best theory from those generated by Definition 21 is that moving some formula to the left in one place might require moving something to the right elsewhere. For example, if a literal is used to guard some of the free variables of a disjunction, then it must be added to the set of constraints with the appropriate polarity. This will prevent its movement at a place where guarding the variables might be more decisive, for example in front of an existential clause. Because small changes in one place of the translation unpredictably affect the others, finding the best one should probably involve some kind of optimization mechanism. But what would be the quantity being optimized?

In the previous chapter, we gave a list of properties that a good translator should possess. One of these was:

- Highly customizable, it should be easy to change behavior by a small change in a parameter. We want a translator which can be much better used to navigate the (hopefully) large space of possible translations.

We argue that whenever one has a preference for a particular feature that the translated theories should possess, then if this feature can be precisely defined, it must be based on syntactic considerations. For example, one can always wish for a translation that generates the shortest proofs. However, one cannot a priori know which theory will yield the most effecient proof. Instead, one must rely on heuristics or intuitive measures, such as the number of disjunctions, in order to state this preference clearly. It is then based on the structure of the coherent theories.

If we allowed the user to state such preferences in whatever manner she chooses, and use the aggregation of these preferences as the parameter to be optimized over, we would solve several problems at once:

1. Contain the explosion by filtering out theories with low aggregate score.
2. Provide for an extremely powerful yet simple mechanism to choose the best theories from the class it produces.
3. Give an abstract interface to the decision-making core of the translator.

But how should these preferences be presented to us? We should let the user decide. The only thing we should care about is being able to add the "scores" of given theories in order to create a bigger one, and to compare the scores of different theories. We need four things:

- A datatype $\mathscr{M}$. An object with type $\mathscr{M}$ will be called a *mass.*
- An operation $\oplus : \mathscr{M} \times \mathscr{M} \to \mathscr{M}$ representing addition of masses.
- A relation $\preceq : \mathscr{M} \times \mathscr{M} \to \mathbf{Prop}$ which will be assumed to be a partial order between masses.
- An element $o : \mathscr{M}$ representing the zero (empty) mass.

Once we have these data, the only remaining parameter is a function which computes the mass added to a coherent theory by a given clause. This assumes the principle of *locality*, that the preferences depend on local elements of the coherent theory's structure. Aggregating these together produces a compound mass of the theory which we are to optimize over.

In any optimization context, there is a trivial duality between maximizing and minimizing the objective function. Here we choose to *minimize* the mass of the theories, and think of mass as measuring the difficulty of proof search as assessed by local syntactic criteria.

We now define the ultimate form of our translation from first-order logic to coherent logic.

Let the quadruple $(\mathscr{M}, \oplus, \preceq, o)$ be fixed.

Definition 22. The *General Translation* from first-order logic to coherent logic with *mass structure* $(\mathscr{M}, \oplus, \preceq, o)$, *weighing function* $\bumpeq$ mapping coherent clauses to elements of $\mathscr{M}$, and *cutoff mass* Λ gives for each FOL formula Φ a family of coherent theories $\mathcal{P}(\Phi) = \pi_0\mathcal{G}(\Phi)$. Here a *coherent theory* consists of a set of coherent clauses $\mathcal{C}$, atomic constraints $\mathfrak{p} : \mathcal{A} \rightharpoonup \{T, F\}$, and mass $m : \mathscr{M}$.

The function $\bumpeq$ is extended to finite sets of clauses by

$$\bumpeq(\mathcal{C}) = \bigoplus_{C \in \mathcal{C}} \bumpeq(C)$$

where association is, for example, to the right.

A set of coherent theories $\{\mathcal{T}_i\}$ is *compatible* if $\bigvee \mathfrak{p}_i$ exists[3] and $\bigoplus m_i \preceq \Lambda$.

Atoms and clauses are defined as in the previous definitions. Theories are defined by the following:

- $\varphi = L$. Then $\mathcal{G}(\varphi)$ is given by

L	$\mathcal{P}(\varphi)$	$\mathcal{N}(\varphi)$
$A(\vec{t})$	$\{(\{C^{\perp}(\varphi)\}, \{(A, T)\}, o)\}$	$\{(\{C^{\perp}(\varphi)\}, \{(A, F)\}, o)\}$
$\neg A(\vec{t})$	$\{(\{C^{\perp}(\varphi)\}, \{(A, F)\}, o)\}$	$\{(\{C^{\perp}(\varphi)\}, \{(A, T)\}, o)\}$

- $\varphi = \forall \vec{x}.\psi$. Then

$$\mathcal{P}(\varphi) = \{(\mathcal{C} \cup \{C^{\forall+}(\varphi)\}, \mathfrak{p}, \bumpeq(C^{\forall+}(\varphi)) \oplus m) \mid (\mathcal{C}, \mathfrak{p}, m) \in \mathcal{P}(\psi)\}$$
$$\mathcal{N}(\varphi) = \{(\mathcal{C} \cup \{C^{\forall-}(\varphi)\}, \mathfrak{p}, \bumpeq(C^{\forall-}(\varphi)) \oplus m) \mid (\mathcal{C}, \mathfrak{p}, m) \in \mathcal{N}(\psi)\}$$

- $\varphi = \exists \vec{x}.\psi$. Let $\mathcal{C}^{\exists-}(\varphi) = \{C^{\exists+}(\varphi), C^{\perp}(\psi)\}$. Then

$$\begin{aligned}\mathcal{P}(\varphi) &= \{(\mathcal{C} \cup \{C^{\exists+}(\varphi)\}, \mathfrak{p}, \bumpeq(C^{\exists+}(\varphi)) \oplus m) \mid (\mathcal{C}, \mathfrak{p}, m) \in \mathcal{P}(\psi)\} \\ &\quad \cup \{(\mathcal{C} \cup \mathcal{C}^{\exists-}(\varphi), \mathfrak{p}, \bumpeq(\mathcal{C}^{\exists-}(\varphi)) \oplus m) \mid (\mathcal{C}, \mathfrak{p}, m) \in \mathcal{N}(\psi)\} \\ \mathcal{N}(\varphi) &= \varnothing\end{aligned}$$

- $\varphi = \varphi_1 \vee \cdots \vee \varphi_n$. For $\mathfrak{f} \subseteq \{1, ..., n\}$, put

$$\mathcal{G}_i(\mathfrak{f}) = \begin{cases} \mathcal{P}(\varphi_i) & i \notin \mathfrak{f} \\ \mathcal{N}(\varphi_i) & i \in \mathfrak{f} \end{cases}$$

[3]Recall that $\bigvee \mathfrak{p}_i$ is $\bigcup \mathfrak{p}_i$ whenever the latter relation is single valued.

$$\mathcal{P}_{\mathfrak{f}} = \Big\{ \big(\bigcup \mathcal{C}_i \cup \{C_{\mathfrak{f}}^{\vee +}(\varphi)\}, \bigvee \mathfrak{p}_i, \hat{=}(C_{\mathfrak{f}}^{\vee +}(\varphi)) \ \oplus\ \bigoplus_i m_i \ \big) \ \Big| \ \mathcal{T}_i = (\mathcal{C}_i, \mathfrak{p}_i, m_i) \in \mathcal{G}_i(\mathfrak{f}), \{\mathcal{T}_i\} \text{ compatible} \Big\}$$

$$\mathcal{N}_{\mathfrak{f}} = \Big\{ \big(\bigcup \mathcal{C}_i \cup \{C_{\mathfrak{f}}^{\vee -}(\varphi)\}, \bigvee \mathfrak{p}_i, \hat{=}(C_{\mathfrak{f}}^{\vee +}(\varphi)) \ \oplus\ \bigoplus_i m_i \ \big) \ \Big| \ \mathcal{T}_i = (\mathcal{C}_i, \mathfrak{p}_i, m_i) \in \mathcal{G}_i(\mathfrak{f}), \{\mathcal{T}_i\} \text{ compatible} \Big\}$$

Then $\mathcal{T}(\varphi) = (\bigcup \mathcal{P}_{\mathfrak{f}}, \bigcup \mathcal{N}_{\mathfrak{f}})$.

- $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_n$. Let $\mathcal{G}_i(\mathfrak{f})$ be defined as in the previous clause. Then $\mathcal{T}(\varphi) = (\bigcup \mathcal{P}_{\mathfrak{f}}, \bigcup \mathcal{N}_{\mathfrak{f}})$, where for $\mathfrak{f} \subseteq \{1, ..., n\}$

$$\mathcal{P}_{\mathfrak{f}} = \Big\{ \big(\bigcup \mathcal{C}_i \cup \mathcal{C}_{\mathfrak{f}}^{\wedge +}(\varphi), \bigvee \mathfrak{p}_i, \hat{=}(\mathcal{C}_{\mathfrak{f}}^{\wedge +}(\varphi)) \ \oplus\ \bigoplus_i m_i \ \big) \ \Big| \ \mathcal{T}_i = (\mathcal{C}_i, \mathfrak{p}_i, m_i) \in \mathcal{G}_i(\mathfrak{f}), \{\mathcal{T}_i\} \text{ compatible} \Big\}$$

$$\mathcal{N}_{\mathfrak{f}} = \Big\{ \big(\bigcup \mathcal{C}_i \cup \mathcal{C}_{\mathfrak{f}}^{\wedge -}(\varphi), \bigvee \mathfrak{p}_i, \hat{=}(\mathcal{C}_{\mathfrak{f}}^{\wedge -}(\varphi)) \ \oplus\ \bigoplus_i m_i \ \big) \ \Big| \ \mathcal{T}_i = (\mathcal{C}_i, \mathfrak{p}_i, m_i) \in \mathcal{G}_i(\mathfrak{f}), \{\mathcal{T}_i\} \text{ compatible} \Big\}$$

Example 23. We will now present the result of applying the general translation to the formula of Example 13, for which the bidirectional translation produced 128 theories. Recall that this was the formula

$$\Phi = (\forall x. \neg Px \vee \neg Qx) \wedge (\exists y. Py \wedge Qy)$$

Together with the simplifications of section 3.1, our final translator, when fed the above formula as the input, gives the theories

```
%% Total mass: 4.0
tP(V1), fP(V1) => false.
tQ(V1), fQ(V1) => false.
tAnd_6(Y), fP(Y) =>       false.
tQ(X) =>                  fP(X).
true =>                   dom(Y), tAnd_6(Y), tQ(Y).


Next theory:
```

```
%% Total mass: 4.0
tP(V1), fP(V1) => false.
tQ(V1), fQ(V1) => false.
tAnd_6(Y), fQ(Y) =>         false.
tP(X) =>                    fQ(X).
true =>                     dom(Y), tAnd_6(Y), tP(Y).


Next theory:

%% Total mass: 8.0
tP(V1), fP(V1) => false.
tQ(V1), fQ(V1) => false.
tP(X), tQ(X) =>             false.
true =>                     dom(Y), tP(Y), tQ(Y).
```

The last of these yields a contradiction after two inference steps. While in this case the mass assigned to the theory is not the least, this is an artifact of the formula's simplicity: for more complex examples, the mass is a pretty good indicator of how long the proof search is likely to take (relative to other theories).

The use of cut-off mass in the definition above yields a positive effect which might have been unexpected. Because weighing against the maximum mass is done at every node, any high-mass theories are discarded much faster than they would be had mass acted merely as a sorting order. In fact, low values of Λ allow us to control the rate of the exponential explosion. So in addition to its use in selecting the better theories, the mass mechanism also acts as a valve of the flexibility-versus-speed tradeoff inherent in the translator.

4.3 Soundness and completeness

The purpose of this section is to show that, if $\mathcal{T}$ is any of the coherent theories returned by the bidirectional translation of φ, then

$$\neg\varphi \text{ is a tautology} \iff \varphi^t \vdash_{\mathcal{T},C^\perp(\varphi)} \perp$$

The clause $C^{\perp}(\varphi)$ was added to accomodate those translations where φ appears in the negative position. In such a case, the right side could equivalently be written as $\top \vdash_{\mathcal{T}} \varphi^f$.

We begin with a few basic notions and properties concerning coherent provability.

Definition 24. A coherent theory $\mathcal{T}$ is *stronger* than $\mathcal{T}'$ if it proves at least as much. Precisely, $\mathcal{T}$ is stronger than $\mathcal{T}'$ if for any context Γ and fact G we have

$$\Gamma \vdash_{\mathcal{T}'} G \Longrightarrow \Gamma \vdash_{\mathcal{T}} G$$

Definition 25. $\mathcal{T}, \mathcal{T}'$ are *equientailing*, or just *equivalent*, if $\mathcal{T}$ is stronger than $\mathcal{T}'$ and $\mathcal{T}'$ is stronger than $\mathcal{T}$.

The notion of equientailence turns out to be too strong for our purposes; it is more convenient to relativize it to some (excluded) set of atoms, or to the particular formula φ.

Definition 26.
- An atom $A(\vec{t})$ is *φ-pure* if its predicate symbol is not one of T_ψ, F_ψ for some non-literal $\psi \subseteq \varphi$.
- A context Γ is φ-pure if for each $A \in \Gamma$, A is φ-pure. A clause $\Gamma \to \Delta$ is φ-pure if Γ and Δ are φ-pure. A coherent theory $\mathcal{T}$ is φ-pure if every $C \in \mathcal{T}$ is φ-pure.
- Two coherent theories $\mathcal{T}_1, \mathcal{T}_2$ are *φ-equivalent*, written $\mathcal{T}_1 \approx_\varphi \mathcal{T}_2$, if for all φ-pure Γ, G we have

$$\Gamma \vdash_{\mathcal{T}_1} G \iff \Gamma \vdash_{\mathcal{T}_2} G$$

Proposition 27. *(Weakening)*

$$\Gamma \vdash_{\mathcal{T}} Q \Longrightarrow \Gamma, \Delta \vdash_{\mathcal{T}} Q$$

Proof. Induction. (If δ is a derivation of $\Gamma \vdash Q$, then δ is also a derivation of $\Gamma, \Delta \vdash Q$.) □

Our next result is the "easy part" of the cut-elimination theorem in the coherent setting.

Let $\mathcal{T}$ be a fixed CL theory. We write $\vdash$ in place of $\vdash_{\mathcal{T}}$.

Proposition 28. *(Cut-Elimination) Suppose that for given Γ and Q we have*

$$\Gamma \vdash Q \qquad\qquad \Gamma, Q \vdash G$$

Then $\Gamma \vdash G$.

Proof. We proceed by induction on the derivation of $\Gamma \vdash Q$.

Base case: If $Q \in \Gamma$ then $\Gamma = \Gamma \cup \{Q\} \vdash G$.

Induction: Suppose that $C \in \mathcal{T}$ is a clause of the form $\bigwedge A_m \to \bigvee B_n$, with $\{A_m^\sigma\} \subseteq \Gamma$ and $\Gamma, B_n^\sigma \vdash Q$ for each n. From $\Gamma, Q \vdash G$ we get, by weakening, that $\Gamma, B_n^\sigma, Q \vdash G$ for each n. Now we apply this proposition (the IH) to $(\Gamma, B_n^\sigma) \vdash Q$, which gives $\Gamma, B_n^\sigma \vdash G$, for each n. Clause C is applicable (with the same σ), and we have $\Gamma \vdash G$.

This completes the proof. □

Remark 29. The previous theorem could perhaps be better termed as "cut admissibility", since it asserts that the derivation rule in its statement is subsumed by Definition 1. This is also true in the classical case — if one is looking for a complete first-order calculus, the simplest option is simply not to include the cut as a rule of inference in the first place. Hence the choice of name "cut-elimination" puts great emphasis on how essential the rule is for having a good theory of proofs, suggesting priority even over the issue of completeness. Indeed, if the above proposition failed, we would not have an interesting logic — a point made by Girard et al. [1989].

The following characterization of CL provability is sometimes useful.

Corollary 30. *In order that $\Gamma \vdash Q$ it is necessary and sufficient that for an arbitrary fact G we have*

$$\Gamma \vdash G \iff \Gamma, Q \vdash G \tag{4.3}$$

Proof. Suppose that (4.3) holds for arbitrary G, and consider $G = Q$. Clearly, $\Gamma, Q \vdash Q$. Hence $\Gamma \vdash Q$.

Conversely, assume $\Gamma \vdash Q$. We verify (4.3).
($\Rightarrow$) Weakening.
($\Leftarrow$) Proposition 28. □

In what follows $\mathcal{T}, C$ is the notation for $\mathcal{T} \cup \{C\}$.

Proposition 31. *Let $C, \overline{C}$ be coherent clauses with the following form:*

$$\begin{array}{ll} C: & P_0(\vec{x}) \wedge P_1(\vec{x}) \to \bot \\ \overline{C}: & \top \to P_0(\vec{x}) \vee P_1(\vec{x}) \end{array}$$

1. *Suppose that for some i, P_i occurs neither in Γ nor on the right side of a clause in $\mathcal{T}$. Then*

$$\Gamma \vdash_{\mathcal{T}} Q \iff \Gamma \vdash_{\mathcal{T},C} Q$$

2. *Suppose that for some i, P_i occurs neither in Q nor on the left side of a clause in $\mathcal{T}$. Then*

$$\Gamma \vdash_{\mathcal{T}} Q \iff \Gamma \vdash_{\mathcal{T},\overline{C}} Q$$

Proof. 1. Since one of the P_i never occurs in $\mathcal{T}$ on the right, C^σ can never be used in the induction step (Definition 1) unless P_i^σ was in Γ to begin with.

2. Dual. Since P_i never occurs on the left, we have, for any Γ,

$$\Gamma, P_i^\sigma \vdash_{\mathcal{T}} Q \Longrightarrow \Gamma \vdash_{\mathcal{T}} Q$$

by taking the same derivation. Hence any application of $\overline{C}$ is superfluous. $\square$

Next comes the key lemma of the correspondence proof. Essentially, it states the soundness and completeness of the hypertableau technique.

Proposition 32. *Suppose that $C_P = (P_0(\vec{x}) \wedge P_1(\vec{x}) \to \bot)$ is the unique clause in $\mathcal{T}$ containing the atom P_i on the left side of the arrow (for some $i \in \{0, 1\}$). If C is any clause of the form $\bigwedge A_m \to P_i(\vec{t}) \vee \bigvee B_n$, write $\overline{C}$ for the clause $P_{1-i}(\vec{t}) \wedge \bigwedge A_m \to \bigvee B_n$. Then for any such C that also satisfies $P_i \notin \{A_m\}$, and $P_i \notin Q$, we have*

$$\Gamma \vdash_{\mathcal{T},\overline{C}} Q \iff \Gamma \vdash_{\mathcal{T},C} Q$$

Proof. Let $\mathcal{T}, P_0, P_1, i, C_P, C, \Gamma, Q$ be as above.

($\Rightarrow$) Let γ be a derivation of $\Gamma \vdash_{\mathcal{T},\overline{C}} Q$. We show by induction on γ that each use of $\overline{C}$ in γ can be replaced by one application of C followed by C_P.

The base case is trivial: if $Q \in \Gamma$, then certainly $\Gamma \vdash_{\mathcal{T},C} Q$.

The induction case is also trivial, unless it uses $\overline{C}$. So assume that $\{A_m^\sigma\} \subseteq \Gamma$, $P_{1-i}^\sigma \in \Gamma$ (where $P^\sigma = P(\sigma(t_1), ..., \sigma(t_k))$), and $\Gamma, B_n^\sigma \vdash_{\mathcal{T},\overline{C}} Q$ for each n.

By the inductive hypothesis, $\Gamma, B_n^\sigma \vdash_{\mathcal{T},C} Q$ for all n. If we can show that $\Gamma, P_i^\sigma \vdash_{\mathcal{T},C} Q$, then we can conclude that $\Gamma \vdash_{\mathcal{T},C} Q$ by using clause C.

But P_{1-i}^σ is already in Γ. Hence clause C_P is applicable to the context $\Gamma \cup \{P_i^\sigma\}$, yielding $\perp$. Indeed, $\Gamma, P_i^\sigma \vdash_{\mathcal{T},C} Q$.

($\Leftarrow$) Now suppose that γ is a derivation of $\Gamma \vdash_{\mathcal{T},C} Q$. We prove by induction that $\Gamma \vdash_{\mathcal{T},\overline{C}} Q$. Again, the only interesting case is when the last clause used in γ is C.

So suppose that $\{A_m^\sigma\} \subseteq \Gamma$, that for each n we have $\Gamma, B_n^\sigma \vdash_{\mathcal{T},C} Q$ and $\Gamma, P_i^\sigma \vdash_{\mathcal{T},C} Q$. By the inductive hypothesis, we also have $\Gamma, B_n^\sigma \vdash_{\mathcal{T},\overline{C}} Q$ and $\Gamma, P_i^\sigma \vdash_{\mathcal{T},\overline{C}} Q$. The former could yield the conclusion by a single application of $\overline{C}$ if P_{1-i}^σ were in Γ. Alas, it might not be. Instead, we must follow the derivation δ of $\Gamma, P_i^\sigma \vdash_{\mathcal{T},\overline{C}} Q$ until such an instance of $P_{1-i}(\vec{x})$ appears, if it ever does.

We now show by induction on δ that if Δ is any set of facts, then

$$\Gamma, P_i^\sigma, \Delta \vdash_{\mathcal{T},\overline{C}} Q \Longrightarrow \Gamma, \Delta \vdash_{\mathcal{T},\overline{C}} Q \tag{4.4}$$

Base case: $Q \in \Gamma \cup \{P_i^\sigma\} \cup \Delta$. By assumption, $P_i \notin Q$. Hence $Q \in \Gamma \cup \Delta$.

Induction: Let $C' \in \mathcal{T} \cup \{\overline{C}\}$ be a clause having the form $\bigwedge A'_m \to \bigvee B'_n$, and let τ be such that $\{A'^\tau_m\} \subseteq \Gamma \cup \Delta \cup \{P_i^\sigma\}$ and $\Gamma, \Delta, P_i^\sigma, B'^\tau_n \vdash_{\mathcal{T},\overline{C}} Q$ for each n. By the inductive hypothesis $\Gamma, \Delta, B'^\tau_n \vdash_{\mathcal{T},\overline{C}} Q$, and we can use C' to yield the desired $\Gamma, \Delta \vdash_{\mathcal{T},\overline{C}} Q$ whenever $P_i^\sigma \notin \{A'^\tau_m\}$.

But one of the assumptions guarantees that P_i does not occur on the left side of any clause except C_P. So if indeed $P_i^\sigma \in \{A'^\tau_m\}$, then $C' = C_P$ and $\sigma = \tau$. Hence $P_{i-1}^\sigma \in \Gamma \cup \Delta$, and, recalling that $\{A_m^\sigma\} \subseteq \Gamma$, we use clause $\overline{C}$ to derive

$$\Gamma, \Delta \vdash_{\mathcal{T},\overline{C}} Q$$

(Note that we used weakening to get that $\Gamma, \Delta, B_n^\sigma \vdash_{\mathcal{T},\overline{C}} Q$.) This completes the induction.

To finish the proof, take $\Delta = \varnothing$. Since we have $\Gamma, P_i^\sigma \vdash_{\mathcal{T},\overline{C}} Q$, (4.4) gives us $\Gamma \vdash_{\mathcal{T},\overline{C}} Q$. ☐

Corollary 33. *Let $\mathcal{T}$ be a coherent theory.*

1. *Suppose that there exist $C, C' \in \mathcal{T}$ which have the form*

$$\begin{aligned} C = \quad & A_1 \wedge \cdots \wedge A_m \to P(\vec{t}) \vee B_1 \vee \cdots \vee B_n \\ C' = \quad & A'_1 \wedge \cdots \wedge A'_{m'} \wedge P(\vec{t}) \to B'_1 \vee \cdots \vee B'_{n'} \end{aligned}$$

Suppose further that these are the only occurrences of the atom P in the theory $\mathcal{T}$. Let $\overline{P}$ be a new predicate symbol of the same arity as P, and let $\overline{C}, \overline{C}'$ be the following clauses:

$$\begin{aligned} \overline{C} = &\quad A_1 \wedge \cdots \wedge A_m \wedge \overline{P}(\vec{t}) \to B_1 \vee \cdots \vee B_n \\ \overline{C}' = &\quad A'_1 \wedge \cdots \wedge A'_{m'} \to \overline{P}(\vec{t}) \vee B'_1 \vee \cdots \vee B'_{n'} \end{aligned}$$

Let $\overline{\mathcal{T}}$ be obtained from $\mathcal{T}$ by replacing C and C' with $\overline{C}$ and $\overline{C}'$. Then

$$\mathcal{T} \approx_{P,\overline{P}} \overline{\mathcal{T}}$$

2. *Suppose that $C = (\bigwedge A_i \to P(\vec{t}) \vee \bigvee B_j)$ is as before, but now we have q clauses*

$$\begin{aligned} C^1 = &\quad A^1_1 \wedge \cdots \wedge A^1_{m^1} \wedge P(\vec{t}_1) \to B^1_1 \vee \cdots \vee B^1_{n^1} \\ &\quad \vdots \\ C^q = &\quad A^q_1 \wedge \cdots \wedge A^q_{m^q} \wedge P(\vec{t}_q) \to B^q_1 \vee \cdots \vee B^q_{n^q} \end{aligned}$$

Let $\overline{P}$, $\overline{C}$ be as above, and let $\overline{C}^k$ be obtained from C^k by removing $P(\vec{t}_k)$ from the right side and adding $\overline{P}(\vec{t}_k)$ to the left side. Then

$$\mathcal{T} \approx_{P,\overline{P}} \{\overline{C}, \overline{C}^1, \ldots, \overline{C}^q\} \cup \mathcal{T} \setminus \{C, C^1, \ldots, C^q\}$$

Proof. 1. Let $\mathcal{T}_0 = \mathcal{T} \backslash \{C, C'\}$. Consider the clause

$$C^{\perp}_P = P(\vec{x}) \wedge \overline{P}(\vec{x}) \to \perp$$

We have

$$\begin{aligned} \mathcal{T} &= \mathcal{T}_0, C, C' \\ &\approx_{\{P,\overline{P}\}} \mathcal{T}_0, C, C', C^{\perp}_P && \text{Prop. 31} \\ &\approx_{\{P,\overline{P}\}} \mathcal{T}_0, C, \overline{C}', C^{\perp}_P && \text{Prop. 32} \\ &\approx_{\{P,\overline{P}\}} \mathcal{T}_0, \overline{C}, \overline{C}', C^{\perp}_P && \text{Prop. 32} \\ &\approx_{\{P,\overline{P}\}} \mathcal{T}_0, \overline{C}, \overline{C}' && \text{Prop. 31} \\ &= \overline{\mathcal{T}} \end{aligned}$$

2. Similarly, with the first application of Proposition 32 above repeated $q-1$ times.

   □

The above corollary is all that we need to prove the completeness of the bidirectional translation with atomic constraints. Since it returns more theories than the general translation, this suffices to prove the correctness of our translation algorithm. However, we will provide the proof of the full bidirectional translation, just for fun. For this we need one more simple lemma.

Proposition 34. *Suppose that the following clauses both belong to $\mathcal{T}$:*

$$\begin{gathered} P_0(\vec{x}) \wedge P_1(\vec{x}) \to \bot \\ \top \to P_0(\vec{x}) \vee P_1(\vec{x}) \end{gathered}$$

For C having the form $\bigwedge A_m \to P_i(\vec{t}) \vee \bigvee B_n$, let $\overline{C}$ be defined as in the previous proposition. Then

$$\mathcal{T}, \overline{C} \approx \mathcal{T}, C$$

Proof. The proof of $(\Rightarrow)$ goes exactly as before — it doesn't use any additional assumptions on P_i. The proof of the converse is dual. □

Finally, we will also need that right conjunctions distribute over the clausal arrow.

Proposition 35. *Let C_0 be a clause having only conjunction on the right:*

$$\bigwedge_{m=1}^{M} A_m \to \bigwedge_{n=1}^{N} B_n$$

Let $\mathcal{C}$ be the set $\{\bigwedge A_m \to B_n\}_{1 \leqslant n \leqslant N}$. Then for any theory $\mathcal{T}$:

$$\mathcal{T}, C_0 \approx_{\varnothing} \mathcal{T}, \mathcal{C}$$

i.e. $\mathcal{T}, C_0$ and $\mathcal{T}, \mathcal{C}$ are equivalent.

Proof. If γ is a derivation of $\Gamma \vdash_{\mathcal{T},\mathcal{C}} G$, then replacing every application in γ of a clause $C \in \mathcal{C}$ by C_0 gives, with weakening, $\Gamma \vdash_{\mathcal{T},C_0} G$.

Conversely, if δ is a derivation of $\Gamma \vdash_{\mathcal{T},C_0} G$, then we replace in δ every application of C_0 by a sequence of applications of every clause in $\mathcal{C}$ with the same substitution instance. This is always possible because the left sides are the same. □

Theorem 36. *Let Φ be a FOL formula. Let $(P, N) \in \mathcal{G}(\Phi)$ be a positive and a negative bidirectional translation of Φ. Then*

$$\begin{aligned} \neg\Phi \text{ is a tautology} &\iff \Phi^t \vdash_P \bot \\ &\iff \top \vdash_N \Phi^f \end{aligned}$$

Proof. We will prove a stronger fact, namely that

$$\mathcal{T}_0 \approx_\Phi P, (\top \to \Phi^t) \approx_\Phi N, (\Phi^f \to \bot)$$

where $\mathcal{T}_0$ is the basic translation of Φ from Definition 4. This we prove by induction on the number of "flips" in P and N.

For the rest of the proof, let $\mathcal{T} \in \mathcal{G}(\Phi)$ be fixed.

For a subformula $\varphi \subseteq \Phi$, the bidirectional translation which yielded $\mathcal{T}$ picked either a positive or a negative theory from $\mathcal{G}(\varphi)$. We write this theory as $\mathcal{T}_\varphi$. With this notation, put

$$\chi(\varphi) = \begin{cases} 0 & \mathcal{T}_\varphi \in \mathcal{P}(\varphi) \\ 1 & \mathcal{T}_\varphi \in \mathcal{N}(\varphi) \end{cases} \tag{4.5}$$

While $\mathcal{P}(\varphi)$ and $\mathcal{N}(\varphi)$ are the same when φ is atomic, the translation still makes a choice for the polarity of φ. This is the choice referred to in (4.5).

The *rank of reversal* is defined by induction as the number of times the translation took the negative choice in the subformulas of Φ. Formally,

- For φ atomic, $R(\mathcal{T}) = \chi(\mathcal{T})$.
- For φ of the form $\forall\vec{x}.\psi$, $R(\mathcal{T}) = \chi(\varphi) + R(\mathcal{T}_\psi)$, where $\mathcal{T}_\psi$ is the translation of ψ that was chosen in Definition 19.
- Similarly, for $\varphi = \exists\vec{x}.\psi$, we define $R(\mathcal{T}) = \chi(\varphi) + R(\mathcal{T}_\psi)$.
- If φ is of the form $\varphi_1 \vee \cdots \vee \varphi_n$, $\mathcal{T} = \mathcal{T}_1 \cup \cdots \cup \mathcal{T}_n \cup C$ for $\mathcal{T}_i \in \mathcal{G}_i(\mathfrak{f})$ and $\mathfrak{f} \subseteq \{1, ..., n\}$, then $R(\mathcal{T}) = \chi(\varphi) + R(\mathcal{T}_1) + \cdots + R(\mathcal{T}_n)$.
- Similarly, if φ is a conjunction $\varphi_1 \wedge \cdots \wedge \varphi_n$, then

$$R(\mathcal{T}) = \chi(\varphi) + R(\mathcal{T}_{\varphi_1}) + \cdots + R(\mathcal{T}_{\varphi_n})$$

Let $C^{\top +} = (\top \to \Phi^t)$, $C^{\top -} = (\Phi^f \to \bot)$, and put

$$\mathcal{T}^\top = \begin{cases} \mathcal{T}, C^{\top +} & \mathcal{T} \in \mathcal{P}(\Phi) \\ \mathcal{T}, C^{\top -} & \mathcal{T} \in \mathcal{N}(\Phi) \end{cases}$$

In either case, let $R(\mathcal{T}^\top) = R(\mathcal{T})$.

We're to show:

$$\mathcal{T}_0 \approx_\Phi \mathcal{T}^\top \tag{4.6}$$

We prove (4.6) by induction on $R(\mathcal{T})$. The induction proceeds by applying Corollary 33 to two clauses of $\mathcal{T}^\top$, which at all times decreases the rank of reversal of our coherent theory. The clauses are chosen by pivoting around the maximal subformula $\varphi \subseteq \Phi$ for which $\chi(\varphi) = 1$, i.e. the highest subformula whose polarity was reversed in its translation to $\mathcal{T}$.

Base case: If $R(\mathcal{T}) = 0$, then $\mathcal{T} \in \mathcal{P}(\Phi)$, and $\mathcal{T}^\top = \mathcal{T} \cup (\top \to \Phi^t)$ is in fact the canonical translation of Φ — which takes the positive choice for *every* subformula of Φ. Hence $\mathcal{T}_0$ and $\mathcal{T}^\top$ are the same theory, in particular they're Φ-equivalent.

Induction: Let $\psi \subseteq \Phi$ be a maximal subformula with $\chi(\psi) > 0$, in the sense that $\psi \subseteq \varphi \subseteq \Phi, \chi(\varphi) > 0 \Longrightarrow \psi = \varphi$. Such a ψ exists if $R(\Phi) > 0$.

We will now select the unique clause in $\mathcal{T}^\top$ which contains ψ^f on the left side of its arrow. This clause will be denoted by $C^\sharp$.

If $\psi = \Phi$, then $\mathcal{T}^\top = \mathcal{T} \cup C^{\top -}$, and we put $C^\sharp = C^{\top -}$.

If $\psi \subset \Phi$, then let φ be the subformula immediately containing ψ, i.e. let φ be $\subseteq$-minimal with the property that $\psi \subset \varphi \subseteq \Phi$. Then $\chi(\varphi) = 0$, and, since $\chi(\psi) = 1$, $C^\sharp$ must be one of the following:

- If $\varphi = \forall x.\psi$, then $C^\sharp = C^{\forall \bot}(\varphi) = (\varphi^t \wedge \psi^f \to \bot)$.
- If $\varphi = \exists x.\psi$, then $C^\sharp = C^\bot(\psi) = (\psi^t \wedge \psi^f \to \bot)$.
- If $\varphi = \bigvee \varphi_n$ and $\psi = \varphi_i$, then $C^\sharp = C_{\mathfrak{f}}^{\vee +}(\varphi)$, where $\mathfrak{f} \subseteq \{1, \ldots, n\}$ is the set of polarities chosen by the translator at node φ. In this case we will also have $i \in \mathfrak{f}$.
- If $\varphi = \bigwedge \varphi_n$ and $\psi = \varphi_i$, then $C^\sharp = (\varphi^t \wedge \psi^f \to \bot)$. (In this case, $C^\sharp \in \mathcal{C}_{\mathfrak{f}}^{\wedge +}(\varphi)$, with $i \in \mathfrak{f}$.)

Notice that in all cases the clause $C^\sharp$ has the form

$$A_1 \wedge \cdots \wedge A_m \wedge \psi^f \to B_1 \vee \cdots \vee B_n$$

with $n > 0$ only possible when φ is a disjunction. Replacing $C^\sharp$ with

$$\overline{C^\sharp} : \qquad A_1 \wedge \cdots \wedge A_m \to \psi^t \vee B_1 \vee \cdots \vee B_n$$

exactly produces the dual clause that would be a member of $\mathcal{T}$ if the positive polarity was chosen for ψ. Thus $C^{\top -}$, $C^{\forall \perp}$, $C_{\mathfrak{f}}^{\vee -}$, etc. are respectively changed into $C^{\top +}$, $C^{\forall +}$, $C_{\mathfrak{f}}^{\vee +}$ etc. (The existential clause simply becomes redundant, leaving behind $C^{\exists +}$.)

To complete the proof, we now consider the structure of ψ.

- ψ is atomic. Then $\mathcal{T}_\varphi = \{C^\perp(\varphi), C^\top(\varphi)\}$, and Proposition 32 gives

  $$\mathcal{T}^\top \approx_\Phi \mathcal{T}^\top \cup \{\overline{C^\sharp}\} \setminus \{C^\sharp\}$$

  The theory on the right has the rank of reversal one less than that on the left, and is therefore Φ-equivalent with $\mathcal{T}_0$ by the inductive hypothesis. Thus $\mathcal{T}^\top \approx_\Phi \mathcal{T}_0$.

- $\psi = \forall y.\psi'$ is universal. Because $\chi(\psi) = 1$, the topmost clause in $\mathcal{T}_\psi$ is $C^{\forall -}(\psi) = (\psi'^f \to \psi^f)$. We call this clause $C^\flat$.

  Now, applying Corollary 33 to $\mathcal{T}^\top$ with $(C^\sharp, C^\flat)$ as (C, C') gives us that $\mathcal{T}^\top \approx_\Phi \overline{\mathcal{T}^\top}$, where the latter is obtained from $\mathcal{T}^\top$ by replacing $C^\sharp$ with $\overline{C^\sharp}$ and $C^\flat = C^{\forall -}(\psi)$ with $\overline{C^\flat} = C^{\forall \perp}(\psi)$. Hence $R(\overline{\mathcal{T}^\top}) = R(\mathcal{T}^\top) - 1$, and by the inductive hypothesis $\mathcal{T}_0 \approx_\Phi \overline{\mathcal{T}^\top} \approx_\Phi \mathcal{T}^\top$.

- ψ is an existential. Then $\mathcal{T}_\psi \in \mathcal{P}(\psi)$. Since $\chi(\psi) = 1$, this case is impossible.

- $\psi = \bigvee \psi_j$ is a disjunction. As in the universal case, put $C^\flat = C_{\mathfrak{f}}^{\vee -}(\psi)$ and apply Corollary 33 to $\mathcal{T}^\top$ and the clauses $(C^\sharp, C^\flat)$. The resulting theory $\overline{\mathcal{T}^\top}$ differs from $\mathcal{T}^\top$ in that ψ was moved from left to right in $C^\sharp$, and from right to left in $C^\flat$. This switches the ψ clause from the negative $C_{\mathfrak{f}}^{\vee -}(\psi)$ to the positive $C_{\mathfrak{f}}^{\vee +}(\psi)$, decreasing the rank of reversal of the theory. The inductive hypothesis yields $\mathcal{T}_0 \approx_\Phi \overline{\mathcal{T}^\top}$, while $\overline{\mathcal{T}^\top} \approx_\Phi \mathcal{T}^\top$ by the corollary.

- If $\psi = \bigwedge \psi_j$ is a conjunction, then $\mathcal{T}_\psi = \mathcal{T}_1 \cup \mathcal{C}_{\mathfrak{f}}^{\wedge -}(\psi)$ for some $\mathfrak{f}$ (where $\mathcal{T}_1$ is the union of the translations of ψ_j). Writing $\mathcal{T}^\top$ as $\mathcal{T}_2 \cup \mathcal{C}_{\mathfrak{f}}^{\wedge -}(\psi) \cup \{C^\sharp\}$, we apply the second part of Corollary 33 to $\mathcal{T}^\top$ with $C^\sharp$ and $\mathcal{C}^{\wedge -}(\psi)$ to get

  $$\mathcal{T}^\top \approx_\Phi \mathcal{T}_2 \cup \mathcal{C}_{\mathfrak{f}}^{\wedge +}(\psi) \cup \{\overline{C^\sharp}\}$$

  The theory on the right hand side has lower rank of reversal, and is Φ-equivalent with $\mathcal{T}_0$ by the inductive hypothesis.

This completes the induction.

Having established the fact that $\mathcal{T}^\top \approx \mathcal{T}_0$ in all cases, we note that

1. $\{\varphi^t\} \vdash_{\mathcal{T}} \bot \iff \top \vdash_{\mathcal{T}, C^{\top +}} \bot$
2. $\top \vdash_{\mathcal{T}} \varphi^f \iff \top \vdash_{\mathcal{T}, C^{\top -}} \bot$
3. $\neg\varphi$ is a tautology $\iff \top \vdash_{\mathcal{T}_0} \bot$

These facts together give the desired result. $\square$

Chapter 5

Conclusion

In the previous chapters, we presented a general translation algorithm from first-order logic into coherent logic with a strong view toward efficiency of proof search in the resulting theory. The algorithm is able to return a number of quite different equivalent theories while providing considerable leeway for fine-tuning the preference-selection mechanism. Furthermore, while the variation in the logical structure of the resulting theories is significant, the task of proof object reconstruction remains very simple.

One methodological problem we encountered in our work is the difficulty of assessing overall translation–prover performance on a massive scale. The Thousands Problems for Theorem Provers (TPTP) library (**?**) is maintained for the purposes of testing theorem provers. However, the library places a very marked accent on the method of resolution, and therefore the majority of its first-order problems are presented in the format of clause normal forms. (Most of the remaining problems contain either equality or function symbols, neither of which we support natively.) Since formulas given in clause normal forms are already Skolemized, this bias toward resolution provers made it difficult for us to evaluate how useful the resulting system can be in formalizing general mathematics.

The overall level of contribution this vector of research will make to the formalization problem therefore remains to be seen. Ultimately, one would like to reduce the difficulty of formalization to the point at which it is commensurate with the task of preparing a mathematical result in the Latex format for publication. The ratio of time required to formalize a result versus the time required to "tex it up" is known as the *de Bruijn factor*. The work of Bezem and Hendriks [2008] showed that using coherent logic one could, in some cases, make the factor approach unity. The possibility that the method could be made general enough to cover full first-order logic clearly should have been explored.

Having begun this exploration in the present thesis, we certainly believe that the approach holds great promise. One piece of evidence is our work with Stefan Berghofer, which will be reported in a joint article currently in preparation. The principal result of that investigation was a full first-order tactic for the Isabelle theorem prover, which was able to quickly solve the vast majority of the problems used as test benchmarks for `Blast`, the fastest first-order tactic of Isabelle (which implements a tableau prover).

What these partial positive results demonstrate most convincingly is that coherent logic is a field which is very much awaiting further research. The performance of coherent logic-based provers on some very difficult problems is impressive. Our translator makes some of this power accessible to general first-order formulas. Yet we believe that the most spectacular applications of coherent logic await just beyond the horizon. The enterprise of formalization will undoubtedly see significant progress in the coming years; it is very likely that coherent logic will be an integral element of this development. In some respects, it already is.

Part II

Lambda Calculus

Chapter 6

The Range Property

In this chapter we construct a counterexample to the range property for the λ-theory $\mathcal{H}$. Additionally, we show that if a certain technical conjecture holds then every term violating the range property must possess some of the characteristic properties of this counterexample.

6.1 Introduction, previous work

A model $\mathfrak{M}$ of untyped lambda calculus has *the range property* if every element m of $\mathfrak{M}$, considered as a map $m : \mathfrak{M} \to \mathfrak{M}$, is either constant or has infinite image. A λ-theory has the range property if its closed term model has the range property. A long-standing conjecture of Barendregt et al. [1976] states that the λ-theory $\mathcal{H}$ identifying all unsolvables has the range property.

The range property was originally conjectured for the lambda theory $\boldsymbol{\lambda}_\beta$ by Böhm [1968]. It was proved independently by Myhill and Barendregt [1984]. Statman and Barendregt have isolated the recursion-theoretic content of the theorem (Barendregt [1993]), which yielded a considerable generalization of it, including in particular all c.e. theories. Barendregt [1993] also provides several constructive proofs.

The case of the lambda theory $\mathcal{B}$ — equating all terms with the same Böhm tree — is resolved quite easily. If $(\lambda x.M)$ is a closed lambda term, then its range in Λ^0 modulo $\mathcal{B}$ is a singleton if $x \notin \mathsf{BT}(M)$. When $x \in \mathsf{BT}(M)$, the Böhm out theorem yields the existence of $\overrightarrow{P} \in \Lambda^0$ with $M\overrightarrow{P} = x\overrightarrow{Q}$. Then the range of $(\lambda x.M)$ is infinite.

These results, together with the fact that all models usually considered for the lambda calculus have the range property, naturally lead to the question of whether it holds for $\mathcal{H}$. The latter is in many ways a natural theory — in accordance with the identification

$$\text{unsolvable} = \text{meaningless}$$

it simply makes all meaningless values the same (the "$\uparrow$-value"). As we will discuss shortly, there are many good reasons to believe the conjecture.

Partial results are given by Intrigila and Statman [2007]. Statman gives some ideas for approaching the problem in its entry in the TLCA list (Statman [1993]). Barendregt [2008] also discusses some proof strategies.

This chapter is organized as follows. First we recall some common notions and notations from pure lambda calculus that will be used subsequently. The exposition is minimal, so any reader in want of a more thorough introduction is hereby referred to the book by Barendregt [1984]. Thereafter we make some comments on the latest paper of Barendregt dealing with the range problem, from 2008, and give some results about sequences of lambda terms viewed as applicative input streams. Next we present our construction, which depends on certain recursion-theoretic facts. The resulting term is rather complicated; consequently we provide a simplification, and a thorough proof that the term has finite range. While this negatively settles the question of the range property for $\mathcal{H}$, some outstanding issues remain. We strongly believe that every term violating the range property must necessarily possess some of the characteristic features of our counterexample. While we have not been able to prove all of our conjectures, we report the partial progress in this converse direction as well. Finally, we discuss some other interesting open problems brought out by the range property question.

6.2 Setup

Lambda terms are given by the grammar

$$\Lambda ::= \; x \,|\Lambda\Lambda|\, \lambda x.\Lambda$$

Here $x \in \{v_0, v_1, \dots\}$ denotes a variable. The terms are identified modulo α-equivalence. Elements of Λ come to life once we stipulate that

$$(\lambda x.M)N = M[x := N]$$

where $M[x := N]$ denotes the lambda term obtained by renaming bound variables of M to be distinct from the free variables of N, and substituting N for x in M. (This is called the β-rule.)

The following notation is standard:

- $\mathtt{I} = \lambda x.x$
- $\mathtt{K} = \lambda xy.x$
- $\mathtt{U}_i^n = \lambda x_0 \dots x_n.x_i$
- $\mathtt{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$
- $[M, N] = \lambda x.xMN, \qquad x \notin MN$
- $\langle M_1, \dots, M_n \rangle = \lambda x.xM_1 \dots M_n, \qquad x \notin \overrightarrow{M}$

From the inductive definition of λ-terms above it follows that any $M \in \Lambda$ must have one of the following forms:

1. $\lambda x_1 \dots x_l.yP_1 \dots P_m$
2. $\lambda x_1 \dots x_l.(\lambda z.Q)P_0 \dots P_m$

with $l, m \geqslant 0$. (If this notation is unfamiliar, see Barendregt [1984].) In the first case we say that M *is a head normal form*; in the second case we say that M *has a head redex* $(\lambda z.Q)P_0$. We say that M *has a head normal form* if there is some $N \in \Lambda$ which is (β-)convertible with M and is a head normal form.

If M has a head redex, it can be contracted to yield a new term M', which will again have one of the forms above. If M' is not a head normal form, its head redex can be contracted again, yielding M'', and so on. If eventually this process brings us to a term which is a head normal form, then we call this term *the principal head normal form of M* and say that *the head reduction of M terminates.* The principal head normal form of M, if it exists, is denoted by $\mathsf{phnf}(M)$.

With the Church–Rosser theorem, it is not difficult to see that M has a head normal form if and only if the head reduction sequence of M terminates (especially if one recalls the result known as Standardization).

A closed term M is *solvable* if there is a sequence of terms $N_1 \dots N_k$ such that $M\overrightarrow{N} = \mathtt{I}$. A general term is solvable if its closure is solvable. The following proposition is one of many (Barendregt [1984]) reasons to consider terms without head normal form as divergent computations.

Fact (Wadsworth)**.** *M is solvable $\iff$ M has a head normal form.*

We recall the notion of Böhm trees. For a set S, the set of all finite sequences of elements of S will be denoted as S^* or $S^{<\omega}$.

Definition 37. For $M \in \Lambda$, the *Böhm tree* of M is a partial map

$$\mathsf{BT}(M) : \mathbb{N}^* \rightharpoonup \Lambda$$

defined by induction on $\sigma \in \mathbb{N}^*$.

If M is unsolvable, then $\mathsf{dom}(\mathsf{BT}(M)) = \varnothing$. In this case, we write $\mathsf{BT}(M)(\sigma)\uparrow$, or $\mathsf{BT}(M)(\sigma) = \bot$.

If $\mathsf{phnf}(M) = \lambda\vec{x}.yP_0...P_{m-1}$, then

- $\mathsf{BT}(M)(\langle\rangle) = \lambda\vec{x}.y$,
- $\mathsf{BT}(M)(\langle i\rangle * \sigma) = \mathsf{BT}(P_i)(\sigma) \quad$ for $i < m$,
- $\mathsf{BT}(M)(\langle i\rangle * \sigma) = \bot \qquad$ whenever $i \geqslant m$.

A λ-*theory* is a set of equations between lambda terms which is closed under the congruence axioms and includes β-conversion. The canonical λ-theory is the one which includes nothing else:

$$\boldsymbol{\lambda} = \{M = N \mid M =_\beta N\}$$

Two other theories are of importance to us: $\mathcal{H}$ and $\mathcal{B}$. The former is generated by taking as axioms the equations

$$\{M = \Omega \mid M \text{ is unsolvable}\}$$

Equality in $\mathcal{H}$ can be characterized by Ω reduction: if we add to β the rule

$$\Omega : \qquad\qquad M \twoheadrightarrow_\Omega \Omega \qquad M \text{ unsolvable}$$

then the corresponding Church–Rosser theorem holds (Barendregt [1984]):

$$M =_{\mathcal{H}} N \iff \exists Z \quad M \twoheadrightarrow_{\beta\Omega} Z \twoheadleftarrow_{\beta\Omega} N$$

$\mathcal{B}$ is the theory which identifies all terms with the same Böhm tree:

$$\mathcal{B} = \{M = N \mid \mathsf{BT}(M) = \mathsf{BT}(N)\}$$

Finally, the theory $\mathcal{H}^*$ is the unique Hilbert–Post completion of $\mathcal{H}$ (as well as $\mathcal{B}$), which characterizes equality in domain models of the lambda calculus. It can be defined as the theory of contextual equivalence:

$$\mathcal{H}^* = \{M = N \mid \forall C[\quad] \quad C[M] \text{ solvable} \iff C[N] \text{ solvable}\}$$

Recall that a λ-theory $\mathscr{T}$ *has the range property* if for each $F \in \Lambda^0$, the set $\{[FX]_{\mathscr{T}} | X \in \Lambda^0\}$ is either infinite or consists of one element. It is known that $\boldsymbol{\lambda}$, $\mathcal{B}$, and $\mathcal{H}^*$ have the range property. Here we show that $\mathcal{H}$ does not.

6.3 Intuition

We begin with an incisive observation made by Barendregt [2008].

First of all, adding equations to a λ-theory $\mathcal{T}$ only decreases the number of $=_{\mathcal{T}}$-equivalence classes in the image of Λ^0 under a given term F. Therefore,

$$\mathcal{T} \subseteq \mathcal{T}' \Longrightarrow |\mathsf{Range}^{\mathcal{T}}(F)| \geqslant |\mathsf{Range}^{\mathcal{T}'}(F)|$$

In particular,

$$|\mathsf{Range}^{\lambda}(F)| \geqslant \; |\mathsf{Range}^{\mathcal{H}}(F)| \geqslant |\mathsf{Range}^{\mathcal{B}}(F)| \tag{6.1}$$

Secondly, we know that the range property holds for the first and last theories in the above sequence. Hence the condition $\infty > |\mathsf{Range}^{\mathcal{H}}(F)| > 1$ showing that the range property fails for $\mathcal{H}$ can only be consistent with (6.1) if $\mathsf{Range}^{\lambda}(F)$ is infinite and $\mathsf{Range}^{\mathcal{B}}(F)$ is a singleton.

The conclusion is that x does not occur on the Böhm tree of M in $F = \lambda x.M$, but it does occur in every β-reduct of M (indeed, in every $\beta\Omega$-reduct). This is a strong condition. For example, it implies that the range property holds for all lambda terms with finite Böhm tree, which subsumes (and gives a soft proof[1] of) Proposition 3 from (Intrigila and Statman [2007]).

Barendregt [2008] concludes with the following discussion:

[1]A Lambda Calculus proof is "soft" if it does not contain analysis of the syntax of lambda terms. Soft proofs use high-level results like fixed-point theorems, continuity, or semantic arguments. In contrast, proofs by induction on derivation of equality, or on the syntax of terms, are "hard" proofs. They use concepts like reduction and contexts, and often quote the Church–Rosser theorem.

Let F be a possible counterexample. [...] Then $x \notin \mathsf{BT}(Fx)$, but $x \in \mathsf{FV}(M)$ for all $M =_{\mathcal{H}} Fx$. This means that during the growth of the Böhm tree of M the free variable x is "pushed into infinity". If some trace of x towards infinity occurs in a context $xP_1 \dots P_n$ with n maximal, then the range of F is infinite by considering $F(\lambda x_1 \dots x_n.\mathsf{c}_k)$. The case that is left is that in Fx the free variable x is pushed into infinity and gets more and more arguments to eat. An example of this situation is an F such that $Fx =_\beta \lambda z.z(F(x\Omega)z)$. Then

$$\begin{aligned} Fx &= \lambda z.z(F(x\Omega)z) = \lambda z.z(z((F(x\Omega\Omega)z)) = \dots \\ &= \lambda z.z^n(F(x\Omega^{\sim n})z) = \dots \end{aligned}$$

In this case $\mathsf{Range}^{\mathcal{H}}(F)$ has cardinality one, as sooner or later $M\Omega^{\sim n} =_{\mathcal{H}} \Omega$. The difficulty is that in general, x, while being pushed to infinity, may get an infinite sequence $P_1, P_2, P_3, \dots$ as arguments (possibly containing the x) and that it is not clear which arguments M can "eat themselves through" this sequence. (We saw that through the sequence $\Omega, \Omega, \cdots$ of cumulative arguments, no M can eat its way, i.e. eventually becomes unsolvable.) It is not decidable which terms can eat themselves through a given infinite sequence.

Regarding the last claim, let us remark that the decidability question does not make sense if we consider both the sequence and the "eater" M as inputs to an algorithm. (Obviously — sequences are infinite objects, and an uncountable set cannot be enumerated by finite inputs over a countable language.) Hence the question can only be formulated as follows: For a sequence S of terms, is there an oracle Turing machine that, given S as an oracle, recognizes the set of lambda terms M which stay solvable even as they are applied to ever more elements of S?

In this interpretation, it is indeed pretty obvious that the question must be Π^S_2-complete for general S. That it is Π^S_2 is immediate; that it is Π^S_2-complete can already be seen for S computable. For consider the term $M = \lambda nx.\langle M(S^+n)(x\langle n\rangle)\rangle$, where S^+ is the successor map on the Church numerals. For a partial computable $f : \mathbb{N} \rightharpoonup \mathbb{N}$, let $L_f \in \Lambda^0$ lambda-define f, and let $X_f = \mathtt{Y}(\lambda xz.zL_f\mathtt{I}x)$. As is known, the problem of testing whether f

is total is Π_2-complete. Yet we have

$$\begin{aligned} f \text{ total} &\iff (M\mathtt{c}_0)X_f \neq_{\mathcal{H}} (M\mathtt{c}_0)\Omega \\ &\iff \forall n \quad X_f\langle \mathtt{c}_0\rangle \ldots \langle \mathtt{c}_n\rangle \neq_{\mathcal{H}} \Omega \end{aligned}$$

i.e. f is total if and only if X_f can eat its way through $\langle \mathtt{c}_0\rangle, \langle \mathtt{c}_1\rangle, \ldots$.

One could suppose that the sequence $P_1, P_2, \ldots$ in the infinite applicative context of x must have a measure of effectiveness because it is constructed by a lambda term. However, the term $\mathtt{Y}(\lambda f x.[f(x\mathtt{K}), f(x(\mathtt{KI}))])$ pushes its argument x to infinity, the traces of x form a Cantor space, and each gives x an infinite applicative context distinct from all others. In this case the range is infinite, but the example shows that the sequences fed to x don't have to be effective — and can in fact have arbitrary Turing degree.

The suggestions of Barendregt are indeed elucidating. They explain why the counterexample to the range property, if it exists, must be complicated. With these constraints, we set out to prove the range property for $\mathcal{H}$ — and were quite surprised to discover a term with the behavior described above.

We begin exactly where the previous discussion left off. While the final proof does not require the notion of "trace" to be formalized, we provide the following definitions to make the intuition precise.

Definition 38. Let $M \in \Lambda$. The *Gross–Knuth sequence of* M is the sequence of λ-terms $\langle M_n\rangle = M_0, M_1, \ldots$ defined by

- $M_0 = M$.
- M_{n+1} is the full development of M_n.

Recall that a *development* of a lambda term consists of recursively contracting all of its redexes and their descendants, until only newly created redexes remain. The *finite developments theorem* states that all developments are finite — and in the λ_β case end in a unique term. This is the manner in which M_{n+1} is obtained from M_n.

Alternatively, the Gross–Knuth sequence is the sequence of reducts of M produced by the Gross–Knuth reduction strategy. By (Barendregt [1984]), this sequence is cofinal in the reduction graph of M.

Definition 39. A *position* π in a lambda term M is just a one-hole context $C[\quad]$ such that $M \equiv C[N]$ for some N ($\equiv$ denotes syntactic equality). Since this N, if it exists, is unique, we call it *the subterm of* M *at position* π.

Just as in the case of first-order terms, contexts have an obvious partial order between them. (Unless stated otherwise, all contexts are assumed to be one-hole contexts.)

Definition 40. 1. Let $C[\ \ ], C'[\ \ ]$ be two contexts. We say that $C[\ \ ]$ is *more general* than $C'[\ \ ]$, or that $C'[\ \ ]$ is *an instance* of $C[\ \ ]$, if there exists a context $D[\ \ ]$ such that $C'[\ \ ] = C[D[\ \ ]]$. In this case we write $C[\ \ ] \subseteq C'[\ \ ]$.

2. Let (N, π) be an occurrence of N in M. Let $D[\ \ ]$ be the most general ($\subseteq$-minimal) context with the property that $\pi = D[[\ \ ]\overrightarrow{P}]$ for some $\overrightarrow{P}$. Then $C[\ \ ] = [\ \ ]\overrightarrow{P}$ is called *the applicative context of* N, and $C[N]$ is called the *scope* of (N, π).

3. A subterm occurrence (N, π) *commands* (N', π') if π' occurs in the scope of (N, π). That is, there exist contexts $C[\ \ ], D[\ \ ]$ such that $\pi = C[[\ \ ]P_1 \dots D[N'] \dots P_k]$ and $\pi' = C[NP_1 \dots D[\ \ ] \dots P_k]$.

Note that the last two notions make sense for positions themselves, not just for subterm occurrences. Hence we will sometimes talk about the applicative context of π when N is clear from the context.

Linguists use the expressions "c-commands", "m-commands", or "governs" when they say that X commands Y; this is a central notion in binding theory. All it says is that Y occurs in an argument of X. In particular, the head reduction strategy evaluates X before Y.

Definition 41. A *trace* of a subterm $N \subseteq M$ is a sequence of positions $\Pi = \langle \pi_n \rangle$ such that π_0 is an occurrence of N in M, and for each n, the subterm of M_{n+1} at position π_{n+1} is a descendant of the subterm of M_n at position π_n, where $\langle M_n \rangle$ is the Gross–Knuth sequence of M.

One shortcoming of the above definition is that it does not take into account Ω reduction, which is necessary to characterize equality in $\mathcal{H}$. There are several ways to deal with this: we could single out those traces that never get inside of an Ω-redex, we could make use of the finite developments theorem for $\beta\Omega$-reduction (Barendregt et al. [1976]), or we could use the quasi–Gross-Knuth strategy, which interleaves full β-developments with Ω-normalization (Barendregt [1984]). Either of these approaches will generalize to η. We will go with the first one since it keeps traces effective.

Definition 42. A trace Π is *imperishable* if for each n, the occurrence π_n is not inside an unsolvable subterm of M.

Definition 43. A trace Π of N is *maximal* if for each n, no descendant of π_n commands π_{n+1}. Equivalently, Π is maximal if for any other trace Π' of N, π'_n never commands π_n.

Now we briefly explore for which input streams (sequences of terms) one can always find terms that eat themselves through the sequence.

Definition 44. Let $\mathscr{S} = \langle M_n \rangle$ be a sequence of lambda terms. Write $X \sim_{\mathscr{S}} Y$ if $XM_0 \ldots M_{n-1} =_{\mathcal{H}} YM_0 \ldots M_{n-1}$ for some n. We write $\Lambda^0/\mathscr{S}$ for the set $\{[M]_{\sim_{\mathscr{S}}} \mid M \in \Lambda^0\}$.

Example 45. Let $\mathscr{S}$ be one of the following sequences:

- $\mathscr{S} = \langle M_n \rangle = \langle \mathtt{I}, \mathtt{I}, \mathtt{I}, \ldots \rangle$
- $\mathscr{S} = \langle M_n \rangle = \langle \mathtt{I}, \Omega, \mathtt{I}, \Omega, \mathtt{I}, \ldots \rangle$
- $\mathscr{S} = \langle M_n \rangle = \langle \mathtt{I}, \mathtt{I}, \Omega, \mathtt{I}, \Omega, \Omega, \mathtt{I}, \ldots \rangle$

Then $|\Lambda^0/\mathscr{S}| = \omega$.

Proof. A simple exercise.

- Let $Xn = \langle Xn \rangle$. Then $n \neq m \Longrightarrow X\mathtt{c}_n \not\sim_{\mathscr{S}} X\mathtt{c}_m$.
- Take $Xn = \langle \lambda y.Xn \rangle$.
- Take $Xkn = \langle k\mathtt{K}(X(\mathtt{S}^+k)n) \rangle$. Then $X\mathtt{c}_0\mathtt{c}_m \sim_{\mathscr{S}} X\mathtt{c}_0\mathtt{c}_n \Longrightarrow n = m$.

Indeed, $\Lambda^0/\mathscr{S}$ is infinite in all cases above. $\square$

As can be seen in the pattern above, an input stream $\langle M_n \rangle$ has "infinite range" if an argument can be passed through $\langle M_n \rangle$ by hopping from one solvable term to the next one. The next proposition formalizes this intuition.

Definition 46. A λ-term M is *fully solvable* if there are terms $N_1, \ldots, N_k$ such that $M\overrightarrow{N} = \mathtt{I}$.

Notice that $M \in \Lambda$ is fully solvable iff $M = \lambda\vec{x}.x_i\overrightarrow{P}$ iff M is solvable and its head variable is bound.

Proposition 47. *Let $\langle M_n \rangle$ be given. If infinitely many fully solvable members of $\langle M_n \rangle$ can be computably enumerated, then $\Lambda^0/\langle M_n \rangle$ is infinite.*

Proof. Suppose there exists an infinite c.e. set $\{(n_0, m_0), (n_1, m_1), \dots\}$ such that for all i, $m_i = \#M_{n_i}$, and M_{n_i} are fully solvable for infinitely many i. By basic computability theory, this is equivalent to the existence of a partial computable function $h : \mathbb{N} \rightharpoonup \mathbb{N}$ with infinite domain such that

$$\forall n.h(n)\downarrow \implies h(n) = \#M_n,\ M_n \text{ fully solvable}$$

(A function is partial computable iff its graph is c.e., and we can make h diverge whenever M_{n_i} is not fully solvable.)

Let G λ-define the following pcf:

$$g(n) = \quad \mathsf{let}\ (k = \mu k \geqslant n.h(k)\downarrow)\ \mathsf{in}\ (h(k), k - n)$$

We assume the coding is such that, if $g(n) = (\#M, d)$, then $G\mathsf{c}_n\mathtt{U}^1_0 = \ulcorner M \urcorner$ and $G\mathsf{c}_n\mathtt{U}^1_1 = \mathsf{c}_d$, where d is the distance from n to the next element enumerated into the domain of h above n.

Next, define

$$R\ulcorner X \urcorner =_{\mathcal{H}} \begin{cases} [\mathsf{c}_l, \mathsf{c}_p] & \mathsf{phnf}(X) = \lambda x_1 \dots x_l.x_i P_1 \dots P_p \\ \Omega & X \text{ unsolvable} \end{cases}$$

$$\begin{aligned} Qan = {} & \mathsf{let}\ [m, d] = Gn \\ & \qquad [l, p]\ = Rm \\ & \quad \mathsf{in}\ \lambda v_1 \dots v_d x.x(\mathtt{U}^p_p)^{\sim l} Qa(n + d) \\ = {} & (\lambda g.(\lambda r.(g\mathtt{U}^1_1\mathtt{K}(\lambda x.r\mathtt{U}^1_0\langle r\mathtt{U}^1_1\mathtt{KI}\rangle xQa(\mathsf{c}_+ n(g\mathtt{U}^1_1)))))(R(g\mathtt{U}^1_0)))(Gn) \end{aligned}$$

Finally, put

$$X_j = Q\mathsf{c}_j\mathsf{c}_0$$

By induction one verifies that when n is in the domain of h, $X_j\langle M_i \rangle^{\sim n}$ has the form $\lambda z.z \dots$ while $X_j\langle M_i \rangle^{\sim n+1}$ is $Q\mathsf{c}_j \dots$. So X_j stays solvable — it survives $\langle M_i \rangle$. Yet $X_j \not\sim_{\langle M_i \rangle} X_{j'}$ if $j \neq j'$, because the parameter c_j is always propagated through. □

Corollary 48. *Let $\langle M_n \rangle$ be a computable sequence of closed terms. Then $\Lambda/{\sim_{\langle M_n \rangle}}$ is either infinite or is singleton.*

Proof. If $\Lambda^0/\sim_{\langle M_n\rangle}$ does not collapse to a single $\sim_{\langle M_n\rangle}$-class, there exists some $X \in \Lambda^0$ such that X survives $\langle M_n\rangle$ (that is, $X \not\sim_{\langle M_n\rangle} \Omega$). This is only possible if $\langle M_n\rangle$ contains infinitely many solvable terms — for otherwise no X could eat its way through the sequence. Since M_n are closed, these terms are moreover fully solvable. Now the previous proposition applies, because we can enumerate these terms by waiting for their head reduction to converge (which is easy to do when $\langle M_n\rangle$ is computable). □

Moral: If an input stream has finite range, then it cannot be given by any effective method. This fact is at tension with the goal of constructing a lambda term that can generate the stream.

6.4 The Devil's tunnel

Recall that for a given $\langle M_n\rangle$, $\Lambda^0/\langle M_n\rangle$ is infinite if one can "hop" through the solvable elements of $\langle M_n\rangle$ while carrying a parameter. We have seen that among sufficient conditions for the existence of such a "hopper" are that the sequence is closed and computable, or that infinitely many fully solvable terms M_{n_i} can be effectively enumerated. If we could find an effectivity condition which was *necessary* for the existence of these hoppers, then we could try to construct a counterexample by making "solvability gaps" so large that no lambda term could hop through them while carrying a state.

We will now define a family of sequences in which the islands of survivability are given by an arbitrary function $f : \mathbb{N} \to \mathbb{N}$. All of these sequences will have $\mathtt{I}$ as a canonical "stateless" survivor. The counterexample is obtained by letting the functions grow so fast that no other terms can survive the sequence.

Definition 49. Let $f : \mathbb{N} \to \mathbb{N}$ be strictly monotone.

1. $\triangle f : \mathbb{N} \to \mathbb{N}$ is given by $\triangle f(n) := f(n+1) - f(n) - 1$.
2. $\vec{f} : \mathbb{N} \to \mathbb{N}^*$ is given by $\vec{f}(n) := \langle f(0), ..., f(n-1)\rangle$.

Definition 50. Let $f : \mathbb{N} \to \mathbb{N}$ be strictly monotone with $f(0) = 0$. The *Devil's tunnel* induced by f is the sequence of terms $\langle F_n\rangle$ defined by

$$F_n := \begin{cases} \Omega & n \notin \mathsf{Range}(f) \\ \mathsf{U}^{\triangle f(k)}_{\triangle f(k)} & n = f(k) \end{cases}$$

Proposition 51. *For any f, $\mathtt{I}$ survives the Devil's tunnel $\langle F_n \rangle$.*

Proof. By induction on k, we show that

$$\mathtt{I}\langle F_n\rangle^{\sim f(k)} = \mathtt{I} \tag{6.2}$$

Base case: Since $f(0) = 0$, $\mathtt{I}\langle F_n\rangle^{\sim f(0)} = \mathtt{I}$ by definition.

Induction: Assume $\mathtt{I}\langle F_n\rangle^{\sim f(k)} = \mathtt{I}$. Then

$$\begin{aligned}
\mathtt{I}\langle F_n\rangle^{\sim f(k+1)} &= \mathtt{I}\langle F_n\rangle^{\sim f(k)} F_{f(k)} F_{f(k)+1} \cdots F_{f(k+1)-1} \\
&= \mathtt{I} F_{f(k)} F_{f(k)+1} \cdots F_{f(k)+\triangle f(k)} \\
&= \mathtt{U}^{\triangle f(k)}_{\triangle f(k)} F_{f(k)+1} \cdots F_{f(k)+\triangle f(k)} \\
&= (\lambda x_0 ... x_{\triangle f(k)}.x_{\triangle f(k)}) F_{f(k)+1} \cdots F_{f(k)+\triangle f(k)} \\
&= (\lambda x_{\triangle f(k)}.x_{\triangle f(k)}) \\
&= \mathtt{I}
\end{aligned}$$

By (6.2) we have for every $n \in \mathbb{N}$:

$$\mathtt{I} = \mathtt{I}\langle F_i\rangle^{\sim n} F_n F_{n+1} \cdots F_{f(n)-1}$$

Thus $\mathtt{I}\langle F_i\rangle^{\sim n}$ is solvable for all n, so $\mathtt{I}$ survives $\langle F_n\rangle$. □

Definition 52. A *tree* is a partial map $T : \mathbb{N}^* \rightharpoonup \mathbb{N}$ such that

- $\sigma \subseteq \tau \in \mathsf{dom}(T) \Rightarrow \sigma \in \mathsf{dom}(T)$
- $\sigma * \langle i \rangle \in \mathsf{dom}(T) \Rightarrow i < T(\sigma)$

Definition 53. Let $f : \mathbb{N} \to \mathbb{N}$. Say that f is *amenable* if there exists a computably enumerable tree T with $\vec{f}$ an infinite path in T.

Proposition 54. *Let $\langle F_n\rangle$ be the Devil's tunnel of f. Then*

$$|\Lambda^0/{\sim_{\langle F_n\rangle}}| > 2 \iff \triangle f \text{ is amenable.}$$

Note that by Proposition 51 there are always at least two distinct $\sim_{\langle F_n\rangle}$-classes.

Proof. We prove ($\Leftarrow$) first.
Let $\alpha = \overrightarrow{\triangle f}$, so that $\alpha(n) = \langle \triangle f(0), ..., \triangle f(n-1)\rangle$. Assume T is a c.e. tree such that $\forall n.\alpha(n) \in T$. Define a Böhm-like tree B as follows:

- $B(\langle\rangle) = (\mathtt{I}, T(\langle\rangle))$
- $B(\sigma * \langle i\rangle) = (\mathtt{U}^{T(\sigma)-1}_{T(\sigma)-1}, T(\sigma * \langle i\rangle))$

Since T is c.e. so is B. By the Characterization Theorem (Barendregt [1984, 10.1.23]) $B = \mathsf{BT}(M)$ for some $M \in \Lambda$. Since $M_\sigma \equiv \lambda\vec{x}.x_n\overrightarrow{M_{\sigma*\langle i\rangle}} \Rightarrow \vec{x} \notin \mathsf{FV}(\mathsf{BT}(M_{\sigma*\langle i\rangle}))$, we can assume without loss of generality that

$$\vec{x}_\sigma \notin M_{\sigma*\langle i\rangle} \tag{6.3}$$

(If the reader is not convinced, she may interpret the equality in what follows as being in $\mathcal{B}$ — this will not affect the conclusions.)

We claim that M is a survivor of $\langle F_n\rangle$ and that $M \not\sim_{\langle F_n\rangle} \mathtt{I}$. To this end, we show by induction that

$$M\langle F_i\rangle^{\sim f(n)} = \lambda z.zM_{\alpha(n)*\langle 0\rangle}\cdots M_{\alpha(n)*\langle T(\alpha(n))-1\rangle}$$

Let $t(n) = T(\alpha(n))$. The above can be rewritten as

$$M\langle F_i\rangle^{\sim f(n)} = \lambda z.z\langle M_{\alpha(n)*\langle i\rangle}\rangle^{\sim t(n)} \tag{6.4}$$

From the definition of M, we have, for $n > 0$:

$$M_{\alpha(n)} \equiv \lambda x_0 ... x_{t(n-1)-1}.x_{t(n-1)-1}\langle M_{\alpha(n)*\langle i\rangle}\rangle^{\sim t(n)} \tag{6.5}$$

We now proceed with the proof of (6.4).

Base case: By definition, $\alpha(0) = \langle\rangle$ and so $t(0) = T(\langle\rangle)$. Thus

$$\begin{aligned}
&M\langle F_i\rangle^{\sim f(0)}\\
&= M\langle F_i\rangle^{\sim 0} = M\\
&= \lambda z.zM_{\langle 0\rangle}\cdots M_{\langle T(\langle\rangle)-1\rangle}\\
&= \lambda z.zM_{\alpha(0)*\langle 0\rangle}\cdots M_{\alpha(0)*\langle t(0)-1\rangle}\\
&= \lambda z.z\langle M_{\alpha(0)*\langle i\rangle}\rangle^{\sim t(0)}
\end{aligned}$$

Induction: Assume (6.4) holds for n. Then

$$\begin{aligned}
&M\langle F_i\rangle^{\sim f(n+1)}\\
&= M\langle F_i\rangle^{\sim f(n)} F_{f(n)} \cdots F_{f(n+1)-1}\\
&=_{(\mathrm{IH})} (\lambda z.z\langle M_{\alpha(n)*\langle i\rangle}\rangle^{\sim t(n)}) F_{f(n)} \cdots F_{f(n+1)-1}\\
&= F_{f(n)}\langle M_{\alpha(n)*\langle i\rangle}\rangle^{\sim t(n)} F_{f(n)+1} \cdots F_{f(n+1)-1}\\
&= \mathtt{U}^{\triangle f(n)}_{\triangle f(n)} M_{\alpha(n)*\langle 0\rangle} \cdots M_{\alpha(n)*\langle t(n)-1\rangle} F_{f(n)+1} \cdots F_{f(n+1)-1}\\
&=_{(\alpha(n)\in T)} M_{\alpha(n)*\langle \triangle f(n)\rangle} \cdots M_{\alpha(n)*\langle t(n)-1\rangle} F_{f(n)+1} \cdots F_{f(n+1)-1}\\
&= M_{\alpha(n+1)} M_{\alpha(n)*\langle \triangle f(n)+1\rangle} \cdots M_{\alpha(n)*\langle t(n)-1\rangle} F_{f(n)+1} \cdots F_{f(n)+\triangle f(n)}\\
&= M_{\alpha(n+1)} M'_{\triangle f(n)+1} \cdots M'_{t(n)-1} F'_1 \cdots F'_{\triangle f(n)}\\
&= M_{\alpha(n+1)} N_1 \cdots N_{t(n)-1}\\
&=_{(6.5)} (\lambda x_0 ... x_{t(n)-1}.x_{t(n)-1}\langle M_{\alpha(n+1)*\langle i\rangle}\rangle^{\sim t(n+1)}) N_1 \cdots N_{t(n)-1}\\
&=_{(6.3)} (\lambda x_1 ... x_{t(n)-1} z.z\langle M_{\alpha(n+1)*\langle i\rangle}\rangle^{\sim t(n+1)}) N_1 \cdots N_{t(n)-1}\\
&= \lambda z.z\langle M_{\alpha(n+1)*\langle i\rangle}\rangle^{\sim t(n+1)}
\end{aligned}$$

This establishes (6.4). As a consequence, we get that $\mathsf{BT}(M\langle F_i\rangle^{\sim f(n)})$ is infinite for all n. Since $N\langle F_i\rangle^{\sim n} = M\langle F_i\rangle^{\sim n} \Longrightarrow N\langle F_i\rangle^{\sim k} = M\langle F_i\rangle^{\sim k}$ for all $k \geqslant n$, we know that N can be neither $\mathtt{I}$ nor Ω, for otherwise $N\langle F_i\rangle^{\sim f(n)}$ would have a finite Böhm tree (see Proposition 51) while $N\langle F_i\rangle^{\sim k} =_{\mathcal{H}} M\langle F_i\rangle^{\sim k} \Longrightarrow N\langle F_i\rangle^{\sim k} =_{\mathcal{B}} M\langle F_i\rangle^{\sim k}$. This concludes the proof of ($\Leftarrow$).

Conversely, suppose $\Omega \not\sim_{\langle F_i\rangle} M \not\sim_{\langle F_i\rangle} \mathtt{I}$.

Let T be the partially Λ-labelled tree defined as follows:

$$T(\langle\rangle) = (M^{\langle\rangle}, f(1)), \qquad \text{where } M^{\langle\rangle} \equiv M$$

$$T(\sigma * \langle i\rangle) = \begin{cases} (M^{\sigma*\langle i\rangle}, \max(l,m)) & \begin{aligned} M^{\sigma*\langle i\rangle} &\equiv \mathsf{phnf}((\pi_1 T(\sigma))\mathtt{U}^i_i\langle\Omega\rangle^{\sim i})\\ &= \lambda x_0 ... x_l.yP_1\cdots P_m\end{aligned}\\ \uparrow & \pi_1 T(\sigma)\mathtt{U}^i_i\Omega\cdots\Omega \text{ is unsolvable}\end{cases}$$

Note that there are i omegas following $\mathtt{U}^i_i$. Also T is c.e., since given σ and i one only needs to wait for the head reduction of a lambda term depending on σ and i to terminate in order to get the label of $\sigma * \langle i\rangle$.

We now prove:

$$M\langle F_i\rangle^{\sim f(n)} = \pi_1 T(\overrightarrow{\triangle f}(n)) \tag{6.6}$$

Base case: $M\langle F_i\rangle^{\sim f(0)} = M\langle F_i\rangle^{\sim 0} = M = \pi_1 T(\langle\rangle) = \pi_1 T(\overrightarrow{\triangle f}(0))$.

Induction: Assume $M\langle F_i\rangle^{\sim f(n)} = \pi_1 T(\overrightarrow{\triangle f}(n))$. Then

$$\begin{aligned} M\langle F_i\rangle^{\sim f(n+1)} &= M\langle F_i\rangle^{\sim f(n)} F_{f(n)} \cdots F_{f(n+1)-1} \\ &=_{(\mathrm{IH})} \pi_1 T(\overrightarrow{\triangle f}(n)) \mathtt{U}^{\triangle f(n)}_{\triangle f(n)} \Omega \cdots \Omega \\ &= \pi_1 T(\overrightarrow{\triangle f}(n) * \langle \triangle f(n) \rangle) \end{aligned}$$

as long as $M\langle F_i\rangle^{\sim f(n+1)}$ is solvable and $\triangle f(n) < \pi_2 T(\overrightarrow{\triangle f}(n))$. The former holds since $M \not\sim_{\langle F_i\rangle} \Omega$. As for the latter, suppose that $T(\overrightarrow{\triangle f}(n)) = (M\langle F_i\rangle^{\sim f(n)}, k)$ with $k \leqslant \triangle f(n)$. Then

$$M\langle F_i\rangle^{\sim f(n)} = \lambda x_0 ... x_l . x_i P_1 \cdots P_m$$

with $l, m \leqslant k \leqslant \triangle f(n)$. But then

$$\begin{aligned} M\langle F_i\rangle^{\sim f(n+1)} &= (\lambda x_0 ... x_l . x_i P_1 \cdots P_m) F_{f(n)} \cdots F_{f(n+1)-1} \\ &= (\lambda x_0 ... x_l . x_i P_1 \cdots P_m) F_{f(n)} \langle \Omega \rangle^{\sim \triangle f(n)} \end{aligned}$$

Among the first $l+1 \leqslant \triangle f(n)+1$ arguments of $M\langle F_i\rangle^{\sim f(n)}$, the only one which is not Ω is $F_{f(n)}$. Thus we must have $i = 0$ in order for $M\langle F_i\rangle^{\sim f(n+1)}$ to be solvable. But

$$\begin{aligned} x_0 P_1 \cdots P_m [x_0 := F_{f(n)}] &= \mathtt{U}^{\triangle f(n)}_{\triangle f(n)} P'_1 \cdots P'_m \\ &= \mathtt{U}^{\triangle f(n)-m}_{\triangle f(n)-m} \end{aligned}$$

since $\triangle f(n) \geqslant m$. Putting $d = (\triangle f(n) - m) + l$, we get

$$M\langle F_i\rangle^{\sim f(n)+1} = (\lambda x_1 ... x_l . \mathtt{U}^{\triangle f(n)-m}_{\triangle f(n)-m}) = \mathtt{U}^d_d$$

Since M is a survivor, $F_{f(n)+1+d}$ must be $F_{f(n')}$ for some $n' > n$. Then

$$\begin{aligned} M\langle F_i\rangle^{\sim f(n')} &= \mathtt{U}^d_d F_{f(n)+1} \cdots F_{f(n')-1} \\ &= \mathtt{U}^d_d F_{f(n)+1} \cdots F_{f(n)+d} \\ &= \mathtt{I} =_{(6.2)} \mathtt{I}\langle F_i\rangle^{\sim f(n')} \end{aligned}$$

contradicting the fact that $M \not\sim_{\langle F_i\rangle} \mathtt{I}$.

We conclude that $\triangle f(n)$ is indeed less than $\pi_2 T(\overrightarrow{\triangle f}(n))$ and that

$$M\langle F_i\rangle^{\sim f(n+1)} = \pi_1 T(\overrightarrow{\triangle f}(n+1))$$

By (6.6) $\overrightarrow{\triangle f}$ is an infinite path in T. Thus it is also an infinite path in the tree $\pi_2 \circ T$, still c.e. So $\triangle f$ is amenable. $\square$

Proposition 55. *There exists a function $f : \mathbb{N} \to \mathbb{N}$ which is not amenable.*

Proof. We will construct f using the finite injury method. Let $\{\varphi_e\}$ be an enumeration of the partial recursive functions. Let $W_e = \mathsf{dom}(\varphi_e)$ be the eth c.e. set. $W_{e,s}$ will denote the finite set of elements enumerated into W_e by stage s. Formally, $W_{e,s} = \{x : \varphi_{e,s}(x)\downarrow\} \cap \{0, ..., s-1\}$, where $\varphi_{e,s}(x)\downarrow$ means that the eth partial function (eth Turing machine) halts on input x in at most s steps. The function f will be constructed by a sequence of finite approximations $\{f_s\}$ which at any given point stabilize after finitely many steps. We will then take $f := \mathsf{lim}_s f_s$. The resulting function f must satisfy, for every e, the following requirement:

$$R_e : \qquad \varphi_e \text{ a tree} \Longrightarrow \exists n.\vec{f}(n) \notin W_e$$

In this proof, all coding of $\mathbb{N}^*$ by elements of $\mathbb{N}$ will be done implicitly. The requirements are ordered as usual: R_e has higher priority than $R_{e'}$ iff $e < e'$. We maintain a list n_e of *guards* to keep track of which part of f_s must remain unchanged to satisfy a requirement of lower priority than e.

Stage s=0: $f_0 = \varnothing$, $n_e = 0$ for all e.

Stage s+1: Suppose f_s is defined. Let $m = \langle e, t\rangle$ be the least code of a pair $\langle e, t\rangle$ with R_e not yet satisfied and

$$f_s \upharpoonright n_e = \langle f_s(0), ..., f_s(n_e) - 1\rangle \subseteq \sigma \in W_{e,t} \tag{6.7}$$

Set $f_{s+1} = \sigma * \langle \varphi_e(\sigma)\rangle$, $n_e = |\sigma| + 1$, $n_{e'} = n_e$ for $e' \geqslant e$. The requirement R_e is now considered satisfied, while $R_{e'}$ are injured for all $e' > e$.

Observe that f_s gets extended infinitely often, since given f_s, $n_e = n_{e'}$ for all $e' > e$, we always have $f_s \in W_i$ where φ_i is the identity function with an arbitrary large index $i > e$, satisfying (6.7). Furthermore, any requirement can only be injured by those with lower priority. Thus we have

$$\forall n \; \exists l \; \forall s > l \quad f_s(n) = f_l(n)$$

Finally, put

$$f = \lim_s f_s$$

Now we prove that for each e, R_e is satisfied. So suppose that φ_e is a tree and R_d are satisfied for each $d < e$. Let s be the last stage when R_e can get injured, so that $\{R_d\}_{d<e}$ are satisfied from s on. If R_e is not satisfied, then $\vec{f}(n) \in W_e$ for all n. So let t be least for which $\vec{f}(n) \in W_{e,t}$ with $n > |f_s|$. Now (6.7) is satisfied by (e, t), and at some stage $s' > s$, (e, t) becomes the smallest pair with this property. At this point $f_{s'}$ is defined to be $\vec{f}(n) * \langle \varphi_e(\vec{f}(n)) \rangle$ and remains (uninjured) with this initial segment from now on. But now if $\vec{f}(n + 1) \in W_e$, then φ_e is not a tree, for it violates the second clause of Definition 52. Specifically, $\varphi_e(\vec{f}(n)) \nless \varphi_e(\vec{f}(n))$, while $\vec{f}(n) * \langle \varphi_e(\vec{f}(n)) \rangle \in \mathsf{dom}(\varphi_e) = W_e$.

We conclude that R_e is indeed satisfied for every e. As a corollary, we get that $\vec{f}(n)$ is not an infinite path in W_e for any c.e. tree φ_e, being what was required to show. □

Let us reflect on what we have gathered so far. For any strictly monotone $f : \mathbb{N} \to \mathbb{N}$ with $f(0) = 0$ we have a sequence of terms $\langle F_n \rangle$ — the Devil's tunnel of f — such that $\Lambda^0/\langle F_n \rangle$ is never a singleton, and moreover has size two if $\triangle f$ is not amenable. We have also shown that non-amenable functions exist. Since f can be recovered from $\triangle f$ via the recurrence $f(0) = 0$, $f(n + 1) = f(n) + \triangle f(n)$, it follows that there exists an f with the Devil's tunnel having range of size 2.

In light of observations of section 6.3, we could attempt to use these facts to disprove the range property for $\mathcal{H}$ as follows. First, we define a term $\Xi \equiv \lambda x.M$ in which the variable x gets pushed to infinity during the growth of $\mathsf{BT}(\Xi x)$ and every surviving trace of x gets an infinite applicative context. Now, if we can arrange it such that the *only* trace of x in Ξ which does not get Ωd gets the Devil's tunnel of some f with $\triangle f$ not amenable, then the range of Ξ will be of size 2. Indeed, in order to disprove the range property, it suffices to carry out the construction of the previous proposition inside a λ-term.

However, although the priority argument above is effective, the constructed f is computable only in the limit. In fact, f cannot be computable by Corollary 48. This problem can be circumvented by using equality of the λ-theory $\mathcal{H}$. Instead of defining a single trace deterministically, we make a branch at every step in the construction when f is extended. In the first case, we assume that the extension is correct — that it will not be injured by any later extensions. In the second case, we assume the very opposite, and wait for some later stage to injure the currently suggested extension.

Now, if the extension is final, meaning it will not be injured again, then the second choice will be stuck in an unbounded search, and hence will become unsolvable. We say that the *trace gets* Ωd. Note that this immediately makes all accumulated information, including the argument variable, disappear from the branch.

However, if the extension is wrong (is not a part of the final f), then the first choice will be false. Since it is imperative that there is only one trace of x, we must ensure that it will be deleted from the first branch in this case. To do this, we have to check, at every iteration of the loop, that the initial segment of f we have accumulated so far is the correct one. If we ever find that an extension injures our current initial segment, we must intentionally force the current branch to become unsolvable. In other words, our process must Ω *itself.*

The next proposition formalizes this final step of the construction. Its proof might appear very technical, but in reality it is little more than a direct transcription of the previous priority construction into a lambda term, with the provisions noted above.

Proposition 56. *The range property fails for the λ-theory $\mathcal{H}$.*

Proof. We adapt the previous construction to Λ.

As is known, all finite subsets, functions, and increasing sequences in $\mathbb{N}$ can be coded bijectively by elements of $\mathbb{N}$. Thus for $A, B \subseteq \mathbb{N}$ finite sets, $F, G \subseteq \mathbb{N} \times \mathbb{N}$ finite functions, and $\sigma, \tau \in \mathbb{N}^*$ strictly monotone, we will write $\#A, \#B, \#F, \#G, \#\sigma, \#\tau$, etc. to denote the elements of $\mathbb{N}$ corresponding to the *codes* of A, B, f, g, σ, τ, etc. The following λ-term $\mathtt{X}$ performs the induction step in (6.7). Its definition uses a fixed-point operator (many times).

$$\mathtt{X}\mathsf{c}_{\#F}\mathsf{c}_{\#A}\mathsf{c}_{\#\rho}\mathsf{c}_m = \begin{cases} \mathsf{c}_{\#\langle n,\#(A\cup\{e\}),\#\tau\rangle} & \begin{cases} m = \#\langle e, t, \#\sigma\rangle \\ e \notin A, \sigma \supseteq \overrightarrow{F}(k), \#\sigma \in W_{e,t} \\ k := \max\{\rho(i) | i \leqslant e, |\rho|\} \\ \tau := \rho \upharpoonright e * \langle|\sigma|\rangle, \\ \quad \tau(i) = k \text{ for } |\rho| \leqslant i < e \\ n := \#(\sigma * \langle\varphi_e(\#\sigma)\rangle) \end{cases} \\ \mathtt{X}\mathsf{c}_{\#F}\mathsf{c}_{\#A}\mathsf{c}_{\#\rho}(S^+\mathsf{c}_m) & \text{otherwise} \end{cases}$$

Given f_s, A, ρ, where $\rho = \langle n_e\rangle_{e<\max A}$ and $A = \{e \mid R_e \text{ has been satisfied}\}$, the term $\mathtt{X}\mathsf{c}_{\#f_s}\mathsf{c}_{\#A}\mathsf{c}_{\#\rho}\mathsf{c}_m$ checks if m codes a triple $\langle e, t, \sigma\rangle$ satisfying (6.7). If so, it returns f_{s+1} and the next $\langle n_e\rangle$, else it continues the search for m.

In what follows we'll also need the following terms:

$$W\mathsf{c}_{\#G}\mathsf{c}_{\#\langle \#F,a,r\rangle} = \begin{cases} \mathsf{c}_{\#\langle \#F,a,r\rangle} & G \nsubseteq F \\ W\mathsf{c}_{\#G}(\mathtt{X}\mathsf{c}_{\#F}\mathsf{c}_a\mathsf{c}_r c_0) & \text{otherwise} \end{cases}$$

$$Vxs = \begin{cases} x & s = \mathsf{c}_{\#\langle\rangle} \\ V(x\mathtt{U}^k_k\langle\Omega\rangle^{\sim k})t & s = \mathsf{c}_{\#\langle k\rangle * \tau}, t := \mathsf{c}_{\#\tau} \\ \Omega & \text{otherwise} \end{cases}$$

For $F \subseteq G$ sequences, we write $G\backslash F$ for the sequence $\langle G(|F|)...G(|G|-1)\rangle$.

Now take $\pi_i\mathsf{c}_{\#\langle n_1,n_2,n_3\rangle} = \mathsf{c}_{n_i}$ with $\pi_i\Omega =_{\mathcal{H}} \Omega$. Put

$$Mxis = (\lambda far.Nxifar(\mathtt{X}far0))(\pi_1 s)(\pi_2 s)(\pi_3 s)$$

$$Nxifarn = \begin{cases} \lambda z.z(Myfn)(Mxi(Wfn)) & \begin{cases} i = \mathsf{c}_{\#F_i}, f = \mathsf{c}_{\#F_s}, F_i \subseteq F_s \\ G := F_s\backslash F_i, y := Vx\mathsf{c}_{\#G} \end{cases} \\ \Omega & \text{otherwise} \end{cases}$$

The operation of M can be intuitively described as follows. Given an initial segment F_i and a triple $\langle f, a, \sigma\rangle$ with $f = \#F_s$, crash if F_s is not an extension of F_i. If it is, split into two processes: one where x is applied to the new part of F — which is what F_s added over F_i — digging further the Devil's tunnel, another where F_s is assumed to be a false extension and the search for an alternative one is performed. Note that if F_s was indeed the right trace, then the second process is gonna get Ωd. Conversely, if $\mathtt{X}$ at any point t beyond s yields an F_t which is not consistent with F_s, then all the children of the first process will get Ωd, because at stage t the condition $F_i \subseteq F_s$ is not going to hold for them anymore and they will crash. Yet if F_i is indeed a part of the final F, then as per the proof of the previous proposition, we know that it will be extended. In this case one of the processes will always have a solvable child.

We define

$$\Xi = \lambda x.Mx\mathsf{c}_{\#\langle\rangle}\mathsf{c}_{\#\langle\langle\rangle\langle\rangle\langle\rangle\rangle}$$

As $\mathsf{BT}(\Xi)$ grows, the occurrences of x in Ξ have the structure of a binary tree, with exactly one path being extended infinitely often. Along this path, the argument x gets the Devil's tunnel of a function f, with $\triangle f$ not amenable by construction. Therefore there are exactly two elements in $\mathsf{Range}(\Xi)$ modulo the theory identifying all unsolvables: $\Xi\Omega$ and $\Xi\mathtt{I}$.

We conclude that Ξ is a counterexample to the range property in $\mathcal{H}$. $\square$

The proof above, while falsifying the conjecture, is quite indirect. First of all, it makes use of Proposition 54, which translates λ-calculus properties of $\langle F_n\rangle$ into recursion-theoretic properties of $\triangle f$. The main construction again uses an argument from recursion theory. Although the underlying idea is rather simple, the machinery supporting the proof is rather bloated. To improve this situation, we tried to look for a more direct path to falsifying the conjecture. The next section presents this improved result.

6.5 A simplified counterexample

Now we simplify the previous construction, such that it does not make use of Proposition 54. We briefly recall the notation.

Notation. For $M \in \Lambda^0$, $\#M$ is the *Gödel number* of M, and $\ulcorner M \urcorner = \mathsf{c}_{\#M}$ is the *quote* of M. Let $\mathtt{E}$ be an *enumerator*: a λ-term satisfying $\mathtt{E}\ulcorner M \urcorner = M$ for each M. For $\sigma \in 2^*$, let $|\sigma|$ be the length of σ, $\#\sigma = 2^{|\sigma|} - 1 + \sum_{i<|\sigma|} \sigma(i)2^i$, and $\ulcorner \sigma \urcorner = \mathsf{c}_{\#\sigma}$. We write $M{\downarrow}^k$ if the head reduction of M terminates after at most k steps. If such a k exists, we may write $M{\downarrow}$, otherwise we write $M{\uparrow}$. If $M{\downarrow}$, then $\mathsf{phnf}(M)$ denotes the principal head normal form of M. Finally,

1. $\|M\| = \begin{cases} l+m & \mathsf{phnf}(M) = \lambda x_0 ... x_l . y P_1 \cdots P_m, \\ \uparrow & M \text{ unsolvable} \end{cases}$

2. $L_n = \mathtt{E}\mathsf{c}_n$, so that $\Lambda^0 = \{L_0, L_1, ...\}$.

Definition 57. We begin by defining a family of λ-terms $\{M^\sigma\}_{\sigma \in 2^*}$ with $\mathsf{FV}(M^\sigma) \subseteq \{x\}$ by induction on the indices σ:

$$\begin{array}{lcl} M^{\langle\rangle} & = & x \\ M^{\sigma * \langle 0 \rangle} & = & M^\sigma \mathtt{I} \\ M^{\sigma * \langle 1 \rangle} & = & \begin{cases} M^\sigma \mathtt{U}^k_k \Omega^{\sim k} & k = \|M^\sigma[x := L_{|\sigma|}]\| \\ \Omega & M^\sigma[x := L_{|\sigma|}] \text{ unsolvable} \end{cases} \end{array}$$

For $N \in \Lambda^0$, let $M^\sigma_N = M^\sigma[x := N]$. Note that $M^\sigma_\Omega =_{\mathcal{H}} \Omega$. Finally, let $M^\sigma_\natural$ denote $M^\sigma_{L_{|\sigma|}}$.

Definition 58. The above is rendered effective by going over to the codes; let F be the term which λ-defines the following pcf f:

$$
\begin{array}{lcl}
f(\#\langle\rangle) & = & \#\mathtt{I} \\
f(\#(\sigma * \langle 0\rangle)) & = & \#\lambda z.L_{f(\#\sigma)}z\mathtt{I} \\
f(\#(\sigma * \langle 1\rangle)) & = & \begin{cases} \#\lambda z.L_{f(\#\sigma)}z\mathtt{U}^k_k\Omega^{\sim k} & k = \|L_{f(\#\sigma)}L_{|\sigma|}\| \\ \uparrow & L_{f(\#\sigma)}L_{|\sigma|} \text{ unsolvable} \end{cases}
\end{array}
$$

From now on, the equality between terms is always taking place in $\mathcal{H}$. With the exception of $\{M^\sigma\}$, we will only consider closed terms.

Proposition 59. *The following properties are easily seen through induction on σ:*

1. $M^\sigma\!\downarrow \implies M^\sigma = x\overrightarrow{Q}$*, with* $Q_i \in \{\Omega, \mathtt{I}, \mathtt{U}^1_1, \mathtt{U}^2_2, \dots\}$ *for all* i.
2. $M^\sigma\!\uparrow,\ \sigma \subseteq \tau \implies M^\tau\!\uparrow$.
3. $M^\sigma = \mathtt{E}(F\ulcorner\sigma\urcorner)x,\ M^\sigma_\natural = L_{f(\#\sigma)}L_{|\sigma|} = \mathtt{E}(F\ulcorner\sigma\urcorner)(\mathtt{E}\mathtt{c}_{|\sigma|})$.

Definition 60. The following terms are defined using a fixed-point combinator. The references to M^σ are handled by looking at its code $F\ulcorner\sigma\urcorner$, as per Proposition 59.c.

$$
S\ulcorner\sigma\urcorner = \begin{cases} \lambda z.z\mathtt{U}^k_k\Omega^{\sim k} & k = \|M^\sigma_\natural\| \\ \Omega & M^\sigma_\natural \text{ unsolvable} \end{cases}
$$

$$
T\ulcorner\sigma\urcorner = \begin{cases} \Omega & M^\sigma\!\uparrow \text{ or } \exists\ \tau * \langle 0\rangle \subseteq \sigma.\ M^\tau_\natural\!\downarrow^{|\sigma|}, \\ \mathtt{I} & \text{otherwise} \end{cases}
$$

$$
X\ulcorner\sigma\urcorner x = T\ulcorner\sigma\urcorner[X\ulcorner\sigma * \langle 0\rangle\urcorner(x\mathtt{I}),\ X\ulcorner\sigma * \langle 1\rangle\urcorner(S\ulcorner\sigma\urcorner x)]
$$

$$
\Xi = X\ulcorner\langle\rangle\urcorner \qquad (=_\eta \lambda x.X\mathtt{c}_0x)
$$

Theorem 61. $|\mathsf{Range}^{\mathcal{H}}(\Xi)| = 2$.

Proof. We begin by introducing some terminology. Let $\sigma \in 2^*$.

$$\begin{array}{lcl} \sigma \text{ is } \textit{doomed} & \iff & \exists\ \tau * \langle 0\rangle \subseteq \sigma.\ M_\natural^\tau\!\downarrow \\ \sigma \text{ is } \textit{healthy} & \iff & \sigma \text{ is not doomed and } M^\sigma \text{ is solvable} \end{array}$$

The following two observations are immediate:

$$\begin{array}{lcl} \sigma \text{ is healthy},\ \tau \subseteq \sigma & \Longrightarrow & \tau \text{ is healthy} \\ \sigma \text{ is healthy},\ M_\natural^\sigma\!\downarrow & \Longrightarrow & \sigma * \langle 1\rangle \text{ is healthy} \end{array}$$

Step 1. Assume σ is healthy. Then

$$\sigma * \langle 1\rangle \text{ is healthy} \iff \sigma * \langle 0\rangle \text{ is doomed}$$

Indeed, $\sigma * \langle 1\rangle$ is healthy $\Longrightarrow M_\natural^\sigma\!\downarrow \Longrightarrow \sigma * \langle 0\rangle$ is doomed.

Conversely, suppose that $\sigma * \langle 0\rangle$ is doomed, so let $\tau * \langle 0\rangle \subseteq \sigma * \langle 0\rangle$ be such that $M_\natural^\tau\!\downarrow$. Since σ is healthy, $\tau * \langle 0\rangle \nsubseteq \sigma$. Thus $\tau = \sigma$ and $M_\natural^\sigma\!\downarrow$. By the second observation, $\sigma * \langle 1\rangle$ is healthy.

Step 2. For each $n \in \mathbb{N}$, there exists a unique σ of length n which is healthy.

Base case: $\langle\rangle$ is healthy since $M^{\langle\rangle} = x$ is solvable and $\neg\exists\tau * \langle 0\rangle \subseteq \langle\rangle$. Furthermore, $\langle\rangle$ is the only sequence of length 0.

Induction: Suppose σ is the unique healthy σ with $|\sigma| = n$. By the first observation, if τ is healthy with $|\tau| = n+1$, then $\tau \supseteq \sigma$. Thus we are to show that $\sigma * \langle i\rangle$ is healthy for exactly one $i \in \{0, 1\}$.

If $\sigma * \langle 1\rangle$ is healthy, then by the Step 1 $\sigma * \langle 0\rangle$ isn't.

So suppose $\sigma * \langle 1\rangle$ is not healthy. Combining the inductive hypothesis with the second observation, we see that this can only happen when $M_\natural^\sigma\!\uparrow$. Thus $M_\natural^\tau\!\downarrow$ holds neither for $\tau = \sigma$, nor (once again by the inductive hypothesis) for $\tau * \langle 0\rangle \subseteq \sigma$. Hence $\sigma * \langle 0\rangle$ is not doomed and moreover is healthy, for $M^\sigma\!\downarrow \Longrightarrow M^{\sigma*\langle 0\rangle}\!\downarrow$.

For $\alpha \in \mathsf{BT}(N)$, let N_α denote the subterm of N at α. Note that

$$\alpha \in \mathsf{BT}(X^{\ulcorner}\sigma^{\urcorner}) \Longrightarrow \alpha \in 2^*$$

Step 3. Let $\tau \in \mathsf{BT}(X\ulcorner\sigma\urcorner)$. We have

$$(X\ulcorner\sigma\urcorner M^\sigma)_\tau = X\ulcorner\sigma * \tau\urcorner M^{\sigma*\tau}$$

In particular, $(\Xi x)_\tau = (X\mathsf{c}_0 x)_\tau = (X\ulcorner\langle\rangle\urcorner M^{\langle\rangle})_\tau = X\ulcorner\tau\urcorner M^\tau$.

We proceed by induction on τ.

Base case: $(X\ulcorner\sigma\urcorner M^\sigma)_{\langle\rangle} = X\ulcorner\sigma\urcorner M^\sigma$.

Induction: Suppose $\forall\sigma.\quad (X\ulcorner\sigma\urcorner M^\sigma)_\tau = X\ulcorner\sigma * \tau\urcorner M^{\sigma*\tau}$.

If $T\ulcorner\sigma\urcorner = \Omega$, then $T\ulcorner\rho\urcorner = \Omega$ for any $\rho \supseteq \sigma$, so

$$X\ulcorner\rho\urcorner M^\rho = \Omega = X\ulcorner\sigma\urcorner M^\sigma = (X\ulcorner\sigma\urcorner M^\sigma)_{\langle i\rangle*\tau}$$

But if $T\ulcorner\sigma\urcorner\downarrow$, then

$$\begin{aligned}
X\ulcorner\sigma\urcorner M^\sigma &= [X\ulcorner\sigma * \langle 0\rangle\urcorner(M^\sigma\mathtt{I}), X\ulcorner\sigma * \langle 1\rangle\urcorner(S\ulcorner\sigma\urcorner M^\sigma)] \\
&= [X\ulcorner\sigma * \langle 0\rangle\urcorner M^{\sigma*\langle 0\rangle}, X\ulcorner\sigma * \langle 1\rangle\urcorner M^{\sigma*\langle 1\rangle}] \\
&= \lambda z.z(X\ulcorner\sigma * \langle 0\rangle\urcorner M^{\sigma*\langle 0\rangle})(X\ulcorner\sigma * \langle 1\rangle\urcorner M^{\sigma*\langle 1\rangle})
\end{aligned}$$

Therefore,

$$\begin{aligned}
(X\ulcorner\sigma\urcorner M^\sigma)_{\langle i\rangle*\tau} &= (X\ulcorner\sigma * \langle i\rangle\urcorner M^{\sigma*\langle i\rangle})_\tau \\
&=_{\mathrm{IH}} X\ulcorner\sigma * \langle i\rangle * \tau\urcorner M^{\sigma*\langle i\rangle*\tau}
\end{aligned}$$

Step 4. $X\ulcorner\sigma\urcorner M^\sigma =_{\mathcal{H}} X\ulcorner\sigma\urcorner\Omega$ unless σ is healthy.

If σ isn't healthy, then either it's doomed or M^σ is unsolvable. In the latter case, $T\ulcorner\sigma\urcorner = \Omega$, so certainly $X\ulcorner\sigma\urcorner M^\sigma = X\ulcorner\sigma\urcorner\Omega = \Omega$.

Suppose σ is doomed. Let n be such that $M^\tau_\natural\downarrow^n$ for some $\tau * \langle 0\rangle \subseteq \sigma$. Then for any $\rho \supseteq \sigma$ with $|\rho| \geqslant n$ we have $T\ulcorner\rho\urcorner = \Omega$, so by Step 3

$$(X\ulcorner\sigma\urcorner M^\sigma)_{\rho\backslash\sigma} = X\ulcorner\rho\urcorner M^\rho = \Omega$$

Thus the Böhm tree of $X\ulcorner\sigma\urcorner M^\sigma$ has depth at most $n - |\sigma|$, and, being finite, is the $\beta\Omega$-nf of $X\ulcorner\sigma\urcorner M^\sigma$. Since x does not occur on the tree, applying the substitution $[x := \Omega]$ leaves the $\beta\Omega$-nf unchanged. At the same time, $X\ulcorner\sigma\urcorner M^\sigma[x := \Omega] = X\ulcorner\sigma\urcorner M^\sigma_\Omega = X\ulcorner\sigma\urcorner\Omega$. We conclude that the two terms are $\mathcal{H}$-equal, being what was required to show.

For $n \in \mathbb{N}$, let σ_n be the healthy σ with $|\sigma| = n$. We have seen that

$$\sigma_0 \subseteq \sigma_1 \subseteq \cdots$$

Step 5. $\Xi N \neq \Xi\Omega \iff \forall n\ M_N^{\sigma_n}\!\downarrow$.

Suppose N is such that $M_N^{\sigma_n} = \Omega$ for some n. By Step 3 we have

$$(\Xi N)_{\sigma_n} = (\Xi x)_{\sigma_n}[x := N] = X\ulcorner\sigma_n\urcorner M_N^{\sigma_n} = X\ulcorner\sigma_n\urcorner\Omega = (\Xi\Omega)_{\sigma_n}$$

But by Step 4, $(\Xi N)_\rho = X\ulcorner\rho\urcorner\Omega = (\Xi\Omega)_\rho$ for all remaining ρ of length n. Thus $\Xi N = \Xi\Omega$ (by first unfolding their Böhm trees up to depth n, and then converting the corresponding subterms at that depth).

(At this point, the reader is invited to the following refreshment: suppose $A = \mathtt{Y}(\lambda fxz.z(f(x\mathtt{I})z))$; why does one have $A\mathtt{I} \neq_{\mathcal{H}} A\Omega$?)

Conversely, suppose $\Xi N = \Xi\Omega$. By the Church–Rosser theorem for $\mathcal{H}$, let C be the common $\beta\Omega$-reduct of these. By Step 3, the Böhm tree of C has solvable subterms at σ_n for each n. So let n be maximal with position σ_n fully developed in C — that is, n is the greatest number with no redexes along σ_n occurring in the syntax tree of C. We then have a subterm D at position σ_n in C with $D = X\ulcorner\sigma_n\urcorner(M_N^{\sigma_n})$. Reducing D at the root until the appearance of the last head redex yields a term $D' \equiv AB$, with $X\ulcorner\sigma_n\urcorner \twoheadrightarrow A$ and $M_N^{\sigma_n} \twoheadrightarrow B \twoheadleftarrow M_\Omega^{\sigma_n} = \Omega$. (This is ensured by using reducing fixed point combinators such as Θ in the definition of X.) Now B, being a reduct of an unsolvable term, must itself be unsolvable. At the same time, $M_N^{\sigma_n}$ reduces to B, so it cannot be solvable either. We conclude that $M_N^{\sigma_n}\!\uparrow$ for some n.

Step 6. $\Xi N \neq \Xi\Omega \implies \Xi N = \Xi\mathtt{I}$.

Suppose $\forall n\ M_N^{\sigma_n}\!\downarrow$. Let $N = L_c$: c is the code of N. Since $M_\natural^{\sigma_c} = M_{L_c}^{\sigma_c}\!\downarrow$, we know that $\sigma_c * \langle 0\rangle$ is doomed. Hence σ_{c+1} is $\sigma_c * \langle 1\rangle$, because it's healthy. Let $M_\natural^{\sigma_c} = M_N^{\sigma_c} = \lambda x_0 ... x_l.x_i P_1 \cdots P_m$. Then

$$\begin{aligned} M_N^{\sigma_{c+1}} &= M_N^{\sigma_c}\mathtt{U}_{m+l}^{m+l}\Omega^{\sim m+l} \\ &= (\lambda x_0 ... x_l.x_i P_1 \cdots P_m)\mathtt{U}_{m+l}^{m+l}\Omega^{\sim m+l} \end{aligned}$$

Since $M_N^{\sigma_{c+1}}\!\downarrow$, i must be 0, for otherwise x_i would be bound to Ω. Hence

$$\begin{aligned} M_N^{\sigma_{c+1}} &= (\lambda x_0 ... x_l.x_0 P_1 \cdots P_m)\mathtt{U}_{m+l}^{m+l}\Omega^{\sim m+l} \\ &= (\lambda x_1 ... x_l.\mathtt{U}_{m+l}^{m+l} P_1 \cdots P_m)\Omega^{\sim m+l} \\ &= \mathtt{U}_l^l\Omega^{\sim m} \end{aligned}$$

Here $M_N^{\sigma_{c+1}}$ again becomes Ω unless $m \leqslant l$. Set $r = l - m$, and we get

$$M_N^{\sigma_{c+1}} = \mathtt{U}_r^r$$

Recall that $M^{\tau * \langle i \rangle}\downarrow \implies M^{\tau * \langle i \rangle} = M^\tau Q_1 \cdots Q_j$ and $j \geqslant 1$. By way of this observation, let $d \in \{c, ..., c + r\}$ be least with the property that $M^{\sigma_{d+1}} = M^{\sigma_{c+1}} Q_1 \cdots Q_t$ and $t \geqslant r$. Then, unless $d = c$ and $t = r = 0$, we have $M^{\sigma_d} = M^{\sigma_{c+1}} Q_1 \cdots Q_s$ for some $s < r$.

Suppose $t > r$. Then $t - s \geqslant 2$, and hence $\sigma_{d+1} = \sigma_d * \langle 1 \rangle$, for

$$M^{\sigma_{d+1}} = M^{\sigma_d} Q_{s+1} Q_{s+2} \cdots Q_t \neq M^{\sigma_d} \mathtt{I} = M^{\sigma_d * \langle 0 \rangle}$$

But then $M^{\sigma_{d+1}} = M^{\sigma_d} \mathtt{U}_k^k \Omega^{\sim k}$, and hence

$$\begin{aligned} M_N^{\sigma_{d+1}} &= M_N^{\sigma_{c+1}} Q_1 \cdots Q_s \mathtt{U}_k^k \Omega \cdots \Omega \\ &= \mathtt{U}_r^r Q_1 \cdots Q_s \mathtt{U}_k^k \Omega \cdots \Omega \\ &= \Omega \end{aligned}$$

since $s < r < t$, contradicting the fact that $M_N^{\sigma_{d+1}}\downarrow$.

So $t = r$ — this also covers the case when $d = c$. Then

$$M_N^{\sigma_{d+1}} = M_N^{\sigma_{c+1}} Q_1 \cdots Q_t = \mathtt{U}_r^r Q_1 \cdots Q_r = \mathtt{I}$$

We deduce that $M_N^{\sigma_{d+1}} = M_{\mathtt{I}}^{\sigma_{d+1}}$. By Step 3 it follows that ΞN and $\Xi \mathtt{I}$ have the same subterm at position σ_{d+1}. Yet by Step 4, they also agree on all other positions of length $d + 1$. Indeed, the two terms are $\mathcal{H}$-equal.

Step 7. $\Xi \mathtt{I} \neq \Xi \Omega$.

By routine induction on n one easily verifies that $\forall n. M_{\mathtt{I}}^{\sigma_n} = \mathtt{I}$.

This completes the proof of the theorem. □

Corollary 62. *The range property fails for $\mathcal{H}\eta$ and $\mathcal{H}\omega$.*

Proof. The same term Ξ that worked for $\mathcal{H}$ also works for $\mathcal{H}\eta$ and $\mathcal{H}\omega$.

That the range of Ξ in $\mathcal{H}\eta$ is still a doubleton is immediate because the Church–Rosser theorem also holds for $\beta\eta\Omega$-reduction (Barendregt [1984, 15.2.15.ii]).

For the $\mathcal{H}\omega$ case, we augment our argument with (Barendregt [1984, 17.2.17]) to conclude that also $\Xi \mathtt{I} \neq_{\mathcal{H}\omega} \Xi \Omega$. □

Remark 63. It is not difficult to generalize this construction to define terms whose range has size n for arbitrary n. The easiest approach digs n tunnels using Ξ, but it is also possible to have one tunnel with $n-1$ survivors.

If the side condition in the definition of $T\ulcorner\sigma\urcorner$ is changed to $M^{\tau}_{\natural}\downarrow^{|\sigma|}\neq \mathtt{I}$, then one gets a term with doubleton range in which the survivable traces form a perfect set of measure zero.

6.6 Possible shapes of terms with finite range

In this section, we prove that, under a very plausible hypothesis, any counterexample to the range property must necessarily possess some of the key features of Ξ.

We call this hypothesis "The Scope Lemma." Although its status might make the name "The Scope Conjecture" more appropriate, we are so convinced of its validity that we adopt the more euphonic *Scope Lemma*.

Recall that a trace of a subterm (N,π) of F is a sequence of descendants of π under the Gross–Knuth sequence $\langle F_n\rangle$ of F.

Conjecture 64 (The Scope Lemma)**.** *Let* Π *be a maximal, imperishable trace of* x *in* Fx*. (Here* x *is fresh for* F*.) If* $FX_0 =_{\mathcal{H}} FX_1$*, then for some* n*,* $A[X_0] = A[X_1]$*, where* $A[\quad]$ *is the applicative context of* (x,π_n) *in* F_n*.*

Motivation. Intuitively, the proposition seems rather obvious.

If FX_0 and FX_1 are convertible, then their reductions to a common term induce two reductions of Fx; the descendants of x in these reductions should acquire scopes that coincide under substitutions $[x := X_i]$ unless they are eliminated by Ω-reduction. Since Π is imperishable, no Ω-redexes exist above the scope $A'[x]$ of any ancestor of (x,π_n). Hence $A[X_0]$ and $A[X_1]$ should be convertible. $\square$

The following is an attempt to prove the Scope Lemma along the lines above. We have included it in case the reader is interested to see where the approach breaks down.

Partial proof. Suppose that $FX_0 =_{\mathcal{H}} FX_1$. By the Church–Rosser theorem for $\beta\Omega$-reduction, there is a term Z such that $FX_i \twoheadrightarrow_{\beta\Omega} Z$ for $i \in \{0,1\}$. By postponement of Ω-reduction, (Barendregt [1984, 15.2.7]) there exist N_0, N_1 such that

$$FX_i \twoheadrightarrow_{\beta} N_i \twoheadrightarrow_{\Omega} Z$$

Now we apply Barendregt's Lemma. This is a classic result which was only recently introduced into the literature by de Vrijer [2007]. We use the formulation in (Barendregt [1984, 14.4.8.ii]), which is very convenient in this particular context.

By part (ii) of that exercise, there exists a multihole context $C_0[\quad,\dots,\quad]$ and a term $M_0 \equiv C_0[x\overrightarrow{P}^0_1,\dots,x\overrightarrow{P}^0_{m_0}]$ such that the reduction $FX_0 \twoheadrightarrow N_0$ can be lifted to

$$Fx \twoheadrightarrow M_0, \quad N_0 \equiv C_0[Q^0_1,\dots,Q^0_{m_0}], \quad X_0\overrightarrow{P}^0_j \twoheadrightarrow Q^0_j$$

Similarly, we have $M_1 \equiv C_1[x\overrightarrow{P}^1_1,\dots,x\overrightarrow{P}^1_{m_1}]$ with $N_1 \equiv C_1[Q^1_1,\dots,Q^1_{m_1}]$, $X_1\overrightarrow{P}^1_j \twoheadrightarrow Q^1_j$, and $Fx \twoheadrightarrow M_1$. Furthermore, the contexts $C_i[\dots]$ lead to the outermost occurrences of x.

By the cofinality of the sequence $\langle F_n\rangle$, there is an n_0 such that $M_i \twoheadrightarrow F_{n_0}$ for both i. Since Π is maximal, the subterm x at position π_{n_0} in F_{n_0} must be a descendant of one of its occurrences within $x\overrightarrow{P}^i_j$ in $C_i[x\overrightarrow{P}^i_1,\dots]$ (as they are precisely those occurrences of x in M_i which are not commanded by any others.) Since Π is imperishable, this occurrence cannot be inside an unsolvable, either.

Now, plugging X_i into $C_i[\dots]$ yields, after reduction that takes place *inside* the holes, terms N_0 and N_1 which are alpha-equivalent up to Ω-reduction. This strongly suggests that the contexts $C_i[\dots]$ are the same, outside their holes and Ω redexes. Toward this end, we now precisely analyze the effects of Ω-reduction from N_i to Z.

Let $\Omega(X)$ denote the Ω-normal form of a lambda term X. One possible reduction from X to $\Omega(X)$ consists of contracting all $\subseteq$-maximal (hence disjoint) Ω-redexes. (See Barendregt [1984, 15.2.10.ii] and the proof there.)

We have

$$\begin{aligned} C_0[Q^0_1,\dots,Q^0_{m_0}] &\equiv N_0 \twoheadrightarrow_\Omega Z \\ C_1[Q^1_1,\dots,Q^1_{m_1}] &\equiv N_1 \twoheadrightarrow_\Omega Z \end{aligned}$$

and therefore $\Omega(N_0) \equiv \Omega(Z) \equiv \Omega(N_1)$.

For a fixed i, let Δ be an occurrence of an Ω-redex in $N_i \equiv C_i[\dots]$. There are three possibilities:

1. Δ is disjoint from every hole of $C_i[\dots]$. In this case, Δ occurs in C_i itself and does not intersect with any Q^i_j.

2. Δ strictly contains one or more holes of $C_i[\dots]$. Contracting Δ then makes these holes disappear. If this Ω-redex is already present in $C_i[x\overrightarrow{P}^i_1, \dots]$, its contraction decreases the number of outermost occurrences of x in M_i.

3. Δ occurs in one of the holes. That is, Δ is a subterm of Q^i_j, occurring in the jth hole of $C_i[\dots]$.

Now, for $i \in \{0, 1\}$, let σ_i be the reduction from N_i to $\Omega(N_i)$ that contracts all maximal Ω-redexes in the above order. Write

$$N_i \overset{\sigma^a_i}{\twoheadrightarrow}_\Omega N'_i \overset{\sigma^b_i}{\twoheadrightarrow}_\Omega N''_i \overset{\sigma^c_i}{\twoheadrightarrow}_\Omega \Omega(Z)$$

where $\sigma_i = \sigma^a_i + \sigma^b_i + \sigma^c_i$. Then we have

1. $N'_i \equiv C''_i[Q^i_1, \dots, Q^i_{m_i}]$, since the redexes contracted in σ^a_i lie outside the holes of $C_i[\dots]$. We can write this using a more explicit notation of multihole contexts:

$$C_i[\quad]_1 \cdots [\quad]_{m_i} \overset{\sigma^a_i}{\twoheadrightarrow} C'_i[\quad]_1 \cdots [\quad]_{m_i}$$

   In particular,

$$C_i[x\overrightarrow{P}^i_1, \dots, x\overrightarrow{P}^i_{m_i}] \overset{\sigma^a_i}{\twoheadrightarrow} C'_i[x\overrightarrow{P}^i_1, \dots, x\overrightarrow{P}^i_{m_i}]$$

   The term to the right of the arrow we denote as M'_i.

2. $N''_i \equiv C''_i[Q^i_{j_1}, \dots, Q^i_{j_{m'_i}}]$. It might seem that $\langle j_1, \dots, j_{m'_i}\rangle$ should be the subsequence of $\langle 1, \dots, m_i\rangle$ that lists all the holes of $C'_i[\dots]$ not contained inside an Ω-redex of N'_i. However, then we would not necessarily have that

$$M'_i \equiv C'_i[x\overrightarrow{P}^i_1, \dots, x\overrightarrow{P}^i_{m_i}] \overset{\sigma^b_i}{\twoheadrightarrow} C''_i[x\overrightarrow{P}^i_{j_1}, \dots, x\overrightarrow{P}^i_{j_{m'_i}}]$$

   because a subterm containing $x\overrightarrow{P}^i_{m_i}$ may only become unsolvable after x is instantiated by X_i. Yet this is only possible if the head reduction of this subterm depends on x — that is, if its principal head normal form contains x as the head variable. Therefore we must reduce C'_i until this happens.

In fact, we will simply assume this has already been done before step a: there are finitely many subcontexts of $C_i[\dots]$ whose head reduction brings one of the holes into the head position (and then this hole commands the other holes of the subcontext, making the corresponding occurrences of x irrelevant). To preserve confluence at Z, the corresponding reductions can be performed in $C_{1-i}[\dots]$ as well.

In summary, we have

$$C'_i[x\overrightarrow{P}^i_1, \dots, x\overrightarrow{P}^i_{m_i}] \overset{\sigma^b_i}{\twoheadrightarrow} C''_i[x\overrightarrow{P}^i_{j_1}, \dots, x\overrightarrow{P}^i_{j_{m'_i}}]$$

3. $N''_i \overset{\sigma^c_i}{\twoheadrightarrow} \Omega(Z)$, for $i \in \{0,1\}$. As all reductions of σ^c_i take place *inside* the context $C''_i[\dots]$, this implies that $C''_0[\dots] \equiv C''_1[\dots]$. That is, C''_0 and C''_1 are the same context. Then $m'_0 = m'_1$, and $Q^0_{j_k} =_\Omega Q^1_{j_k}$ for $1 \leqslant k \leqslant m'_0$. Since $Q^i_{j_k} = X_i\overrightarrow{P}^i_{j_k}$ with $[\ \ ]\overrightarrow{P}^i_{j_k}$ the applicative context of the k'th hole of $C''_i[\ \ ]$, we have that

$$X_0\overrightarrow{P}^0_{l_0} = X_1\overrightarrow{P}^1_{l_1}$$

Clearly, the descendants of both sides of the last equation under the reductions to $F_{n_0}[x := X_i]$ are equal as well, being what was required to show. □

We thank Roel de Vrijer for popularizing Barendregt Lemma in Henk's festschrift and providing the reference to the exercise.

Unfortunately, the proof above has a gap: it is not at all immediate why the two contexts at the end (in part c) should be the same. This inference would be validated by the following.

Conjecture 65. *Suppose that $x \notin F$. Then*

$$Fx = F(C[x]), \quad x \in_{\mathcal{H}} Fx \quad \Longrightarrow \quad C[x] = x$$

The conjecture asserts that F cannot recursively put a non-trivial context around its argument while keeping things solvable. Note that the conjecture fails for $\boldsymbol{\lambda}_\beta$ since $F = \mathtt{Y}(\lambda fx.fC[x])$ has $x \in_\beta Fx$ unless $x \notin_\beta C[x]$. But it seems that the application of the fixpoint combinator is necessary in a manner that makes F unsolvable.

We believe the conjecture can be established by the method of λ-clocks, a new proof technique introduced by Endrullis et al. [2010]. Still, the conjecture appears to be considerably deeper than the Scope Lemma it is supposed to prove!

Assumption. The Scope Lemma is valid.

The assumption above is to be implicitly imputed as a hypothesis to all of the results proved in the remainder of this section.

The first is a rigorous demonstration of Barendregt's observations quoted earlier. Without the assumption — even without the last conjecture — we do not immediately see the truth of the statement

> If some trace of x towards infinity occurs in a context $xP_1 \dots P_n$ with n maximal, then the range of F is infinite by considering $F(\lambda x_1 \dots x_n.\mathsf{c}_k)$.

With the Scope Lemma, the statement can be proved as follows.

Proposition 66. *Let F be given. Suppose there exists an imperishable, maximal trace Π of x in Fx such that for some n_0, m the the scope of (x, π_n) has the form $xP_1 \dots P_m$ for all $n \geqslant n_0$. Then the range of F is infinite.*

Proof. Recall that $\mathtt{U}_i^m = \lambda x_0 \cdots x_m.x_i$. Let $X_k = \mathtt{U}_0^m \mathsf{c}_k$, the m-ary constant function with value c_k. Suppose that for some k, k' we have $FX_k = FX_{k'}$. By the Scope Lemma, there is an n_1 such that $X_k\overrightarrow{Q} = X_{k'}\overrightarrow{Q}$, where $xQ_1 \dots Q_l$ is the scope of (x, π_{n_1}). Hence $X_k\overrightarrow{Q}\overrightarrow{Q}' = X_{k'}\overrightarrow{Q}\overrightarrow{Q}'$ for every descendant (x, π_n) of (x, π_{n_1}) with scope $x\overrightarrow{Q}\overrightarrow{Q}'$, for $n \geqslant n_1$. In particular, $X_k\overrightarrow{P} = X_{k'}\overrightarrow{P}$ (where $X_iP_1 \dots P_l$ are descendants of $X_i\overrightarrow{Q}$ at π_n, for $n = n_0 + n_1$). So we have

$$\begin{aligned} X_k\overrightarrow{P} &= X_{k'}\overrightarrow{P} \\ \mathtt{U}_0^m \mathsf{c}_k P_1 \dots P_m &= \mathtt{U}_0^m \mathsf{c}_{k'} P_1 \dots P_m \\ \mathsf{c}_k &= \mathsf{c}_{k'} \end{aligned}$$

Hence $k = k'$. That is, $k \neq k' \implies FX_k \neq FX_{k'}$. $\square$

Remark 67. The proposition above does not hold for Π which are not maximal. For an example, take $F = \lambda x.\Sigma^{\Omega}(x(x\Omega))$, where $\Sigma^N x = \langle \Sigma^N(xN) \rangle$. Although the applicative context of the inner x is always finite, it is easy to see that $FX = F\Omega$ for all X.

Definition 68. For $F \in \Lambda$, let x be fresh ($x \notin F$) and let $\mathscr{M}$ denote the set of maximal, imperishable traces of x in Fx.

- The *trace tree of* F is the set of finite sequences of positions which are initial segments of elements of $\mathscr{M}$:

$$\mathsf{TT}(F) = \{\langle \pi_0, \ldots, \pi_{n-1}\rangle \mid \Pi \in \mathscr{M}\}$$

- The *trace space of* F is the topological space $\mathsf{TS}(F) = (\mathscr{M}, \mathcal{O})$ where $\mathcal{O}$ is generated by taking as basis the cylinder sets

$$\mathcal{C}_\rho = \{\Pi \mid \rho \subseteq \Pi\}$$

  for each $\rho \in \mathsf{TT}(F)$.

- For $\Pi \in \mathscr{M}$, the *tunnel at* Π is the sequence of lambda terms $\langle T_n\rangle$ defined by:

$$T_n = P_n$$

  where $xP_0 \ldots P_n$ is the first applicative context of x in Π that has more than n arguments. We write $\mathcal{T}(\Pi)$ for the tunnel at Π.

By definition, the trace space is the set of infinite paths through the trace tree, topologized as usual. It is a subspace of the larger space of *all* traces.

Proposition 69. *Given any trace* Π*, it is effective to compute the tunnel* $\langle T_n\rangle$ *at* Π*. (That is,* $\mathcal{T}(\Pi)$ *is Turing-reducible to* Π*.)*

Proof. Given n, we find the least m such that the applicative context $C[x\overrightarrow{P}]$ of x at π_m has at least n elements, and return P_n. By definition, this is the n'th element of the tunnel $\langle T_n\rangle$ at Π. (If Π is imperishable, then the search is guaranteed to terminate, but generally we may not know whether or not $n \in \mathsf{dom}(\mathcal{T}(\Pi))$.)

Since Π is a sequence of positions in the Gross–Knuth sequence $\langle M_n\rangle$ of M, the applicative context at π_n can be computed effectively by fully developing the redexes n times and looking at the syntax of M_n. □

Proposition 70. *The trace space* $\mathsf{TS}(F)$ *is compact.*

Proof. We will show that the set $\mathscr{M}$ is closed in the space of all traces of x in Fx, and that the latter is compact.

In the manner of Definition 68, let $\mathsf{TS}^*(F)$ be the set of all traces Π of x, let $\mathsf{TT}^*(F)$ be the set $\{\langle \pi_0, \ldots, \pi_n\rangle \mid \Pi \in \mathsf{TS}^*(F)\}$, and topologize $\mathsf{TS}^*(F)$ as the set of infinite paths through $\mathsf{TT}^*(F)$. (The basic opens are $\{\Pi \mid \Pi \supseteq \rho\}$, with $\rho \in \mathsf{TT}^*(F)$.) Clearly, the topology on $\mathsf{TS}(F)$ is the topology it inherits as a subspace of $\mathsf{TS}^*(F)$.

Since the space of paths through a finitely branching tree is known to be compact (by the same argument as for the Cantor space), it sufficies to show that $\mathsf{TT}^*(F)$ is finitely branching. Toward that end, let $\rho \in \mathsf{TT}^*(F)$, and let n be the length of ρ.

If $n = 0$, then the only successor of ρ is the singleton $\langle \pi_0 \rangle$, where π_0 is the context $C[\quad] = F[\quad]$. (Since $C[x] = Fx$ and x is fresh for F.)

If $n = m+1$, and Π is a trace which begins with ρ, then π_n is a descendant of π_m under the full development from F_m to F_n. Since the developments are finite in length and in number, the number of descendants of π_m in F_n is finite as well. Hence there are finitely many α such that $\rho * \langle \alpha \rangle \in \mathsf{TT}^*(F)$.

So $\mathsf{TS}^*(F)$ is compact. Now we show that $\mathscr{M}$ is closed in $\mathsf{TS}^*(F)$.

If Π is not imperishable, then π_n is inside an unsolvable term for some n, and none of the traces through π_n are imperishable. So the complement of imperishable traces is open.

If Π is not maximal, then π_n is commanded by π'_n for some other trace Π', and again it follows that no trace through $\langle \pi_0, \ldots, \pi_n \rangle$ is a maximal trace. So the complement of maximal traces is open.

Hence the set of imperishable traces, the set of maximal traces, and their intersection $\mathscr{M}$ are all compact, being closed subsets of a compact space. $\square$

Remark 71. The trace tree is related to two other trees which are important for the analysis of the behavior of a lambda term F. The first is the Böhm tree of F, the relationship with which consists in synchronizing the Gross–Knuth reduction with the growth of $\mathsf{BT}(F)$ and factoring π_n through positions in the Böhm tree. The second is the tree of applicative contexts of maximal traces of x in Fx. The tunnels associated to traces are infinite paths through this "scope tree".

We will need the following elementary fact.

Proposition 72. *Let E be an equivalence relation on a set X. If the set $\{[x]_E \mid x \in X\}$ is finite, then there are finitely many equivalence relations $E' \subseteq \wp(X \times X)$ with $E \subseteq E'$.*

Proof. If E' is $\subseteq$-above E, then some equivalence classes of E are merged in E'. If the set of these classes is finite, then only finitely many mergers are possible. $\square$

Definition 73. Let $\langle M_n \rangle = M_0, M_1, \ldots$ be a sequence of λ-terms.

- $X\langle M_n \rangle^{\sim k} = X M_0 \cdots M_{k-1}$
- For a λ-theory $\mathscr{T}$, $X \sim^{\mathscr{T}}_{\langle M_n \rangle} Y$ if $\exists k \quad X\langle M_n \rangle^{\sim k} =_{\mathscr{T}} Y\langle M_n \rangle^{\sim k}$
- $\langle M_n \rangle$ is *survivable* if $\exists X \in \Lambda^0\ X \not\sim^{\mathcal{H}}_{\langle M_n \rangle} \Omega$. We say X *survives* $\langle M_n \rangle$.
- A trace Π is *survivable* if the tunnel at Π is survivable.

Definition 74. Let $\mathcal{E} \subseteq \wp(\Lambda^0 \times \Lambda^0)$ be the set of equivalences relations on Λ^0. Fix a λ-theory $\mathscr{T}$ and a term $F \in \Lambda^0$.

- The *input stream equivalence operator*

$$\mathsf{ISE} : (\mathbb{N} \to \Lambda) \to \mathcal{E}$$

  associates to each sequence of terms $\langle M_n \rangle$ an equivalence on Λ^0 according to the rule

$$\langle M_n \rangle \quad \longmapsto \quad \sim^{\mathscr{T}}_{\langle M_n \rangle}$$

- The map $\Upsilon : \mathscr{M} \to \mathcal{E}$ is defined by

$$\Upsilon = \mathsf{ISE} \circ \mathcal{T}$$

- The map $\upsilon : \mathsf{TT}(F) \to \mathcal{E}$ is defined by

$$\upsilon(\rho) = \bigcap_{\Pi \supseteq \rho} \Upsilon(\Pi)$$

Remark 75. $\Upsilon(\Pi)$ is the value of the stream equivalence operator at the tunnel of Π. $\upsilon(\rho)$ is the greatest lower bound of $\{\Upsilon(\Pi) \mid \Pi \supseteq \rho\}$ with respect to the inclusion order.

Proposition 76. *υ is monotone.*

Proof. $\sigma \subseteq \tau \Rightarrow \mathcal{C}_\sigma \supseteq \mathcal{C}_\tau \Rightarrow \bigcap_{\Pi \in \mathcal{C}_\sigma} \Upsilon(\Pi) \subseteq \bigcap_{\Pi \in \mathcal{C}_\tau} \Upsilon(\Pi) \Rightarrow \upsilon(\sigma) \subseteq \upsilon(\tau)$. $\square$

Proposition 77. *For any $\rho \in \mathsf{TT}(F)$,*

$$FM = FN \Longrightarrow (M, N) \in \upsilon(\rho)$$

Proof. Suppose that $FM = FN$, and let $\langle T_0, T_1, \ldots\rangle$ be the tunnel at some trace $\Pi \supseteq \rho$. By the Scope Lemma, $M\langle T_n\rangle^{\sim k} = N\langle T_n\rangle^{\sim k}$ for large k. Hence $M \sim_{\Upsilon(\Pi)} N$. Since $\Pi \supseteq \rho$ was arbitrary, $M \sim_{\upsilon(\rho)} N$. □

The following theorem was independently discovered by David and Nour [2010].

Theorem 78. *Suppose that* $\mathsf{TT}(F)$ *has a computable infinite path leading to a survivable tunnel consisting of closed terms. Then the range of* F *is infinite.*

Proof. If Π is computable, then so is $\mathcal{T}(\Pi)$ (Proposition 69). By Corollary 48, $\Upsilon(\Pi)$ is either trivial or partitions Λ^0 into infinitely many classes. Since Π is survivable, it cannot be the former. Then the set of $\upsilon(\langle\rangle)$-classes is infinite as well, and by Proposition 77 so is the image of F. □

Corollary 79. *Suppose that* Fx *has only one trace of* x*, and the tunnel at this trace consists of closed terms. Then the range of* F *is either infinite or is a singleton.*

Proof. If Fx has only one trace Π, then it can be computed by tracking the descendants of x under the Gross–Knuth reduction, which is effective (Barendregt [1984]). If some descendant of x occurs inside an unsolvable subterm, then $x \notin_{\mathcal{H}} Fx$, and thus F is a constant map. This is also the case if $(X, \Omega) \in \Upsilon(\Pi)$ for all $X \in \Lambda^0$. However, if Π is both imperishable and survivable, then the previous corollary applies, and the range of F is infinite. □

Remark 80. The corollary fails if we merely require that there is a unique trace which is survivable, or imperishable. Indeed, our counterexample contains only one imperishable trace. That is, the presence of traces which eventually come under an unsolvable are enough to render the path to the survivable tunnel non-computable.

Definition 81. A position $\rho \in \mathsf{TT}(F)$ is *principal* if there exists a trace $\Pi \supseteq \rho$ with $\upsilon(\rho) = \Upsilon(\Pi)$. In this case, we say that Π is a *generator* for ρ.

Proposition 82. *Suppose that* $\mathsf{Range}^{\mathcal{H}}(F)$ *is finite. Then every trace* $\Pi \in \mathsf{TS}(F)$ *is a generator of a principal node.*

Proof. Let $\Pi = \langle \pi_0, \pi_1, \ldots \rangle$ be given. We're to show that there exists a $\rho \subseteq \Pi$ such that $\upsilon(\rho) = \Upsilon(\Pi)$.

For any n, put $\sigma_n = \langle \pi_0, \pi_1, \ldots, \pi_{n-1} \rangle$. The sequence $\langle \sigma_n \rangle$ is monotone:

$$\langle\rangle = \sigma_0 \subseteq \sigma_1 \subseteq \cdots \subseteq \sigma_n \subseteq \cdots$$

By Proposition 76, so is the corresponding sequence in the image of υ:

$$\upsilon(\sigma_0) \subseteq \upsilon(\sigma_1) \subseteq \cdots \subseteq \upsilon(\sigma_n) \subseteq \cdots$$

But the range of F is finite. By Proposition 77, $\upsilon(\sigma)$ has finitely many equivalence classes for every $\sigma \in \mathsf{TT}(F)$. In particular, $\upsilon(\langle\rangle)$ induces a finite partition of Λ^0. By Proposition 72, there are finitely many equivalence relations E with $\upsilon(\sigma_0) \subseteq E$. Hence the chain above becomes constant after some point on. That is, there exists some n_0 such that $\upsilon(\sigma_n) = \upsilon(\sigma_{n_0})$ for all $n \geqslant n_0$. We claim that $\rho = \sigma_{n_0}$ is generated by Π.

That $\upsilon(\rho) \subseteq \Upsilon(\Pi)$ is immediate.

For the reverse inclusion, suppose that $(M, N) \in \Upsilon(\Pi)$. Then $MT_0 \ldots T_t = NT_0 \ldots T_t$ for some t. Let $m \geqslant n_0$ be such that the applicative context of x at π_{m-1} consists of more than t arguments (such an m exists if Π is maximal, imperishable). Then for any $\Pi' \supseteq \sigma_m$, $M \sim_{\mathcal{T}(\Pi')} N$ as well. That is, $(M, N) \in \bigcap\{\Upsilon(\Pi') \mid \Pi' \supseteq \sigma_m\}$. Hence $(M, N) \in \upsilon(\sigma_m) = \upsilon(\sigma_{n_0})$. □

Corollary 83. *The set of survivable traces is closed, hence compact.*

Proof. Let Π be a trace which is *not* survivable. Then $\Upsilon(\Pi) = \Lambda^0 \times \Lambda^0$, the trivial equivalence relation. By Proposition 82 there is a $\rho \subseteq \Pi$ with $\upsilon(\rho)$ trivial as well. That is, $M \sim_{\mathcal{T}(\Pi')} \Omega$ for all $\Pi' \supseteq \rho$ and $M \in \Lambda^0$.

So any trace which is not survivable is contained in an open neighborhood of such traces. Hence the set of non-survivable traces is open. Its complement, the set of survivable traces, is closed. □

Remark 84. The proposition fails for general F. Consider, for instance, the term $\mathtt{Y}(\lambda f x.[f(x\Omega), \mathtt{\Sigma}^{\mathtt{I}} x])$, where $\mathtt{\Sigma}^N x = \langle \mathtt{\Sigma}^N (xN) \rangle$. That the trace with tunnel $\langle \Omega, \Omega, \ldots \rangle$ is a limit point of survivable traces allows us to conclude, from what has already been established, that this term has infinite range. It is a refreshing exercise to see this through. (Although the general theorems of this section depend on the truth of the Scope Lemma, particular examples such as this one can always be verified manually.)

Combining the facts above, we get a structure theorem for terms with finite range in $\mathcal{H}$.

Theorem 85. *Suppose that F has finite range in $\mathcal{H}$. Then there is a context with k holes $C[\ ]_1 \dots [\ ]_k$ such that $F = \lambda x.C[x]\dots[x]$ and for each i, $\langle\rangle$ is principal in $C[\Omega]_1 \dots [\Omega]_{i-1}[x][\Omega]_{i+1}\dots[\Omega]_k$.*

Proof. For Π a survivable tunnel in $\mathsf{TS}(F)$, let ρ_Π be the principal node which Π passes through (via Proposition 82). Since the set of survivable tunnels is compact by the corollary above, the collection $\{\rho_\Pi \mid \Pi \text{ is survivable}\}$ has a finite subcollection $\rho_1, \dots, \rho_m$ that covers all survivable tunnels. This gives us what we are looking for.

Let $l_i = |\rho_i| - 1$, and let $l = \max\{l_1, \dots, l_m\}$. For each i, the subterm x occurring at position $\rho_i(l_i)$ in F_{l_i} has finitely many descendants $\pi_{i,1}, \dots, \pi_{i,n_i}$ in F_l which are still maximal. Let $C[\dots]$ be the context obtained from F_l by placing holes at positions $\pi_{1,1}, \dots, \pi_{1,n_1}, \pi_{2,1}, \dots, \pi_{3,1}, \dots, \pi_{m,1}, \dots, \pi_{m,n_m}$. Write $k = n_1 + \dots + n_m$; then for for $1 \leqslant j \leqslant k$, $\langle\rangle$ is principal in $C[\Omega]_1 \dots [x]_j \dots [\Omega]_k$, because all its survivable traces pass through the position $\pi_{i,i'}$, which corresponds with the jth hole of $C[\dots]$ and is a descendant of ρ_i, while the latter node is principal. Also, $Fx \twoheadrightarrow F_l \equiv C[x]\dots[x]$. □

The point of Theorem 85 is that it suffices to study tunnels of traces which are generators.

We write $M \sqsubseteq N$ if for any $C[\ ]$:

$$C[M] \text{ solvable} \Longrightarrow C[N] \text{ solvable}$$

Trivially, $M =_{\mathcal{H}^*} N \iff M \sqsubseteq N$ and $N \sqsubseteq M$.

Proposition 86. *Suppose that Π is a generator for ρ and has a survivable tunnel $\langle T_n\rangle$. Let k be the length of the applicative context of x at $\pi_{|\rho|}$. Then for any survivable $\Pi' \supseteq \rho$ with $\langle T'_n\rangle = \mathcal{T}(\Pi')$ and any $n \geqslant k$ we have*

$$T_n, T'_n \textit{ closed} \Longrightarrow T'_n \sqsubseteq T_n$$

Proof. Let $\Pi, \Pi', \langle T_n\rangle, \langle T'_n\rangle, k$ be as above. Suppose that for some $m \geqslant k$ one has $C[T_m] =_{\mathcal{H}} \Omega \neq_{\mathcal{H}} C[T'_m]$. If T_m, T'_m are closed, this is equivalent to

$$\exists \overrightarrow{P} \quad T_n\overrightarrow{P} =_{\mathcal{H}} \Omega \ \wedge\ T'_n\overrightarrow{P} =_{\mathcal{H}} \mathtt{I}$$

Let $X \in \Lambda^0$ be a survivor of $\langle T'_n\rangle$: $X \not\twoheadrightarrow_{\langle T'_n\rangle} \Omega$.

Put $Q = \lambda t_0 \ldots t_m.t_m \overrightarrow{P} X T'_0 \ldots T'_m$. Then

$$\begin{aligned} QT_0 \ldots T_m &= T_m \overrightarrow{P} X T'_0 \ldots T'_m = \Omega X \overrightarrow{T}' = \Omega \\ QT'_0 \ldots T'_m &= T'_m \overrightarrow{P} X T'_0 \ldots T'_m = \mathtt{I} X \overrightarrow{T}' = X \langle T'_n \rangle^{\sim m+1} \end{aligned}$$

Hence $X \sim_{\langle T_n \rangle} \Omega$, but $(X, \Omega) \notin \Upsilon(\Pi')$. This gives us a contradiction, because $\Upsilon(\Pi) = \upsilon(\rho) = \bigcap_{P \supseteq \rho} \Upsilon(P) \subseteq \Upsilon(\Pi')$. □

If we want to replace $\sqsubseteq$ in the proposition above by actual $\mathcal{H}^*$-equality (recall that $\mathcal{H}^*$ is the theory of contextual equivalence), then we need a stronger hypothesis.

Definition 87. A position $\rho \in \mathsf{TT}(F)$ is *monochrome* if $\Upsilon(\Pi) = \Upsilon(\Pi')$ for all $\Pi, \Pi' \supseteq \rho$. That is, the input stream equivalence operator agrees on the tunnels of all traces through ρ:

$$X \sim_{\langle T_n \rangle} Y \iff X \sim_{\langle T'_n \rangle} Y$$

where $\langle T_n \rangle, \langle T'_n \rangle$ are the tunnels at Π, Π' and $X, Y \in \Lambda^0$ are arbitrary.

Proposition 88. *Suppose that $\rho \in \mathsf{TT}(F)$ is monochrome and the applicative context of x at ρ has length k. If $\Pi \in \mathcal{C}_\rho$ has tunnel $\langle T_n \rangle$, then for all $\Pi' \in \mathcal{C}_\rho$ with $\mathcal{T}(\Pi') = \langle T'_n \rangle$, and all $n \geqslant k$:*

$$T_n, T'_n \textit{ closed} \Longrightarrow T_n =_{\mathcal{H}^*} T'_n$$

Proof. Let ρ, Π,... be as above. If the equivalence relations induced by traces through ρ are all equal, then $\upsilon(\rho) = \Upsilon(\Pi) = \Upsilon(\Pi')$. Hence Π and Π' are both generators for ρ. In particular, the previous proposition applies to Π,Π' in either order. Hence

$$T_n, T'_n \text{ closed} \Longrightarrow T'_n \sqsubseteq T_n \wedge T_n \sqsubseteq T'_n$$

The right side is trivially equivalent to $T_n =_{\mathcal{H}^*} T'_n$. □

The results of this section show that the Scope Lemma would go a long way toward characterizing terms with finite range and their tunnels. However, it also opens up other issues, such as the behavior of tunnels which might have free variables. Can any of the last results above be extended beyond the closed setting?

6.7 Open terms, open problems

In section 6.3 it was proved that any tunnel which has a partial computable subtunnel consisting of fully solvable terms has infinitely many survivors. This condition of full solvability is rather artificial — it was forced upon us by our inability to deal with free variables. In order to have a full account of survivability of input streams (which is necessary if we are to have a full characterization of terms with finite range) we need to address the free variable issue.

Given a sequence of (open) terms $\langle M_n \rangle$, we are interested in the structure of terms X for which $x\overrightarrow{M}[x := X]$ stays solvable i.e. always has a head normal form. The only variables which could play a role in the head reduction of $x\overrightarrow{M} = x\langle M_i \rangle^{\sim n}$ under the substitution $\sigma = \sigma_X = [x := X]$ are those which occur on the Böhm tree of M_n for some n. For $y \in \mathsf{BT}(M_n)$, we have the following possibilities:

- y is bound in M_n. Then y can in principle be instantiated by any term which the abstraction binding y is applied to. This effectively gives X control over y and its arguments, a fact we exploited in the case of fully solvable terms.

- y is equal to x. Then all occurrences of y in M_n are bound by the substitution σ_X, hence X only can control y if some arguments to y are separable from $\overrightarrow{M}$ in the sense of Barendregt [1984] — which would allow X to act differently at the root of $X\overrightarrow{M}$ and where y occurs.

- y is a free variable not equal to x. Then y is unaffected by σ, and, should y ever make it to the head position in the reduction of $X\langle M_i^\sigma \rangle^{\sim n}$, it would "freeze" this reduction, turning $X\langle M_i^\sigma \rangle^{\sim m}$ into $y\overrightarrow{P}\langle M_{i+n}^\sigma \rangle^{\sim m-n}$ for all $m \geqslant n$. If an arbitrary argument could be applied to $y\overrightarrow{P}$ before y becomes the head variable, then $\Lambda^0/\langle M_n \rangle$ would trivially be infinite.

This much seems obvious. In fact, if $\langle M_n \rangle$ is a tunnel inside a lambda term, there is an additional complication with the third case, namely that the free variable might actually be bound, not in M_n but in an abstraction above the scope of $x\overrightarrow{M}$ — in a context strictly containing the trace to $\langle M_n \rangle$. The consequences of this will be analyzed later in this section. It doesn't affect the conclusion that, for a free occurrence of $y \neq x$, it is of utmost importance to decide whether y can be "Böhmed out" to the head position.

If any such y exists, then the survivability and the range questions would have immediate positive answers.

However, the ability to Böhm out subterms of M_n is made complicated by the second case. The usual technique is insufficient, because we can only choose the substitution σ, and cannot subsequently apply the result to any "selector" terms. Yet if we have multiple occurrences of x on the path from the root of $x\overrightarrow{M}$ to y, then the substitution must be carefully chosen to perform the Böhming out simultaneously.

This brings us to an interesting computational problem in the pure lambda calculus, namely the following

Problem. Given: a term $M \equiv xM_0 \dots M_{n-1}$, and $y \neq x$ a variable occurring freely on the Böhm tree of M. Construct a λ-term X, with $y \notin \mathsf{FV}(X)$, such that $M[x := X] = y\overrightarrow{N}$.

We think of y is being "Böhmed out" by the substitution $[x := X]$. Again, the classic technique of Barendregt [1984] does not apply, because there the substitution is followed by a vector of "selectors" which actually do the Böhming out. Here we need the X to supply these vectors by itself. This problem could arise in any situation where we want to perform a Böhm-out action deep inside a term, and only have access to the head variable there; our motivation with open tunnels is just an example.

The problem can be equivalently formulated in terms of positions: if $\pi \in \mathsf{BT}(M)$, find a substitution $[x := X]$ such that the head reduction of $M[x := X]$ eventually brings π into the head position (so that the descendant of π under the head reduction comes to the root).

Clearly, if z is a free variable of M that occurs somewhere along the path π, then the problem has no solution unless $z \equiv x$, so the only free variable which can command the position we want to Böhm out is x.

On the other hand, if all variables occurring along π are bound, then the problem has a trivial solution: $X = \lambda\vec{a}.a_i\overrightarrow{P}$, where $i = \pi(0)$ and $M_i\overrightarrow{P} = y\overrightarrow{N}$. (When M_i is closed, the existence of $\overrightarrow{P}$ follows by the Böhm out theorem, but by inspecting the proof it is manifest that we only need substitution for the free variables along π.)

The case that remains is that the path π leading to $y \in \mathsf{BT}(M)$ passes through one or more occurrences of x, in addition to the one at the root. We are then to decide whether these occurrences can be instantiated by the same term in a way that would allow y to come up to the top.

Here we will state a conjecture that will fully characterize the set of positions π that can be accessed via some substitution σ_X. Its statement most closely resembles Separability Theorem of Coppo et al. [1978] and can be viewed as an extension of that result. (In fact it is pretty obvious that the conjecture must be true, but its proof appears to be very technical.)

Example 89. For each term M below, decide whether there is an $X \in \Lambda^0$ such that $M[x := X] = y\overrightarrow{N}$.

$$\begin{array}{ll}
x\Omega(xy\Omega) & x\Omega\mathtt{I}(x(x\Omega\mathtt{I}y)\mathtt{K}\Omega) \\
x\Omega(xy\Omega)\mathtt{I} & x\Omega(x\mathtt{K}\Omega(x\mathtt{I}y\Omega))\mathtt{I} \\
x\mathtt{I}(x\mathtt{I}(xy\Omega)) & x(x\Omega\mathtt{I})(x\mathtt{K}\Omega(x\mathtt{I}y\Omega))\Omega \\
x\mathtt{I}(x\Omega(xy)) & x\mathtt{I}\Omega(\lambda w.x(w\mathtt{K}\Omega)\Omega(x(w\Omega\overline{\mathtt{K}})y\Omega)) \\
x(x\Omega(xy)\mathtt{K})\Omega\mathtt{I} & x\mathtt{I}\Omega(\lambda w.x(w\mathtt{K}\Omega)\Omega(x(w\overline{\mathtt{K}}\Omega)y\Omega)) \\
x(x\Omega(x\Omega y\mathtt{I})\mathtt{K})\Omega\mathtt{I} & x[\mathtt{K},\Omega]\Omega(x[\overline{\mathtt{K}},\mathtt{K}](x[\overline{\mathtt{K}},\Omega](x[\mathtt{K},\overline{\mathtt{K}}]\Omega y)\Omega)\Omega) \\
x(x\Omega(xy\Omega\mathtt{I})\mathtt{K})\Omega\mathtt{I} & x\mathtt{I}(x(x\Omega y)\Omega\mathtt{I})\mathtt{I}
\end{array}$$

We encourage the reader to try some of these exercises before proceeding to get a feel for the problem. The examples in the first column are easy; those in the second column are harder, and the last one is the hardest.

With the conjecture that we state here, the reader would be able to solve all these problems immediately. Furthermore, when a solution exists, she would be able to quickly write the term X by inspecting the structure of M.

We will assume as known the notions of Böhm trees, head normal forms, etc. If the introduction in the previous section does not suffice, the reader should consult Barendregt [1984]. For $\sigma \in \mathsf{BT}(M)$, M_σ is the subterm of M at position σ. The *Böhm rank* of M is the number $m - l$, where $M = \lambda x_1 \dots x_l.yP_1 \dots P_m$; the Böhm rank of M exists and is well-defined whenever M is solvable. We write $M \sim_\sigma N$ if M_σ and N_σ have the same head variable and the same Böhm rank. The head variable of M_σ is denoted by $y_\sigma(M)$. Finally, if σ and τ are strings and $\sigma \subseteq \tau$, then $\sigma\backslash\tau$ denotes the string $\langle\sigma(|\tau|), \sigma(|\tau|+1), \dots, \sigma(|\sigma|-1)\rangle$.

Definition 90. In what follows, $M \in \Lambda$ and $\mathcal{F} \subseteq \Lambda$.

1. An *M-filtration* is a sequence $\mathcal{F}_0 \supseteq \mathcal{F}_1 \supseteq \cdots$ such that $\forall i \exists \sigma_i \in \mathsf{BT}(M)$:

$$(\forall N \in \mathcal{F}_i : \ \sigma_i \in \mathsf{BT}(N)) \wedge (\mathcal{F}_{i+1} = \{N \in \mathcal{F}_i \mid N \sim_{\sigma_i} M\})$$

   The position σ_i is called the *pore* of the filtration at i.

2. M is *distinct from* $\mathcal{F}$, written $M \mid \mathcal{F}$, if there exists a finite M-filtration $\mathcal{F}_0 \supseteq \cdots \supseteq \mathcal{F}_n$ with $\mathcal{F}_0 = \mathcal{F}$ and $\mathcal{F}_n = \varnothing$.

3. M is *separable from* $\mathcal{F}$, written $M \parallel \mathcal{F}$, if there are $N_1 \cdots N_k \in \Lambda$ with

$$\begin{aligned} M\overrightarrow{N} &= \mathtt{K} \\ F\overrightarrow{N} &= \overline{\mathtt{K}} \qquad \forall F \in \mathcal{F} \end{aligned}$$

Proposition 91. *For closed M, $\mathcal{F}$:*

$$M \mid \mathcal{F} \iff M \parallel \mathcal{F}$$

Proof. By the methods of Coppo et al. [1978]. □

The definition below defines two concepts by mutual induction.

Definition 92. Let $M \in \Lambda(x)$ be given.

- υ is *x-accessible* in M if the set $\{\sigma \subseteq \upsilon \mid y_\sigma(M) = x\}$ can be partitioned as $\mathcal{S} \cup \mathcal{C}$ such that
  1. M is x-distinct from $\{M_\sigma \mid \sigma \in \mathcal{S}\}$.
  2. $\{\upsilon \backslash \sigma \mid \sigma \in \mathcal{C}\}$ forms a $\subseteq$-chain,
- M is *x-distinct* from $\mathcal{F}$ if there exists an M-filtration $\mathcal{F}_0 \supseteq \cdots \supseteq \mathcal{F}_n$ such that $\mathcal{F}_0 = \mathcal{F}$, $\mathcal{F}_n = \varnothing$, and for each i the pore σ_i is x-accessible in M and in N, for each $N \in \mathcal{F}_i$.

Definition 93. A subterm N of M at position σ is *Böhmed out* by an instance ρ of M,written $\sigma \overset{\rho}{\hookleftarrow} M$, if a descendant of (N, σ) occurs in the root position at some point in the head reduction sequence of M^ρ.

Conjecture 94. *Let $M \in \Lambda(x)$ and $\upsilon \in \mathsf{BT}(M)$.*

$$\exists \rho \quad \upsilon \overset{\rho}{\hookleftarrow} M \iff \upsilon \textit{ is } x\textit{-accessible in } M$$

In particular, if $M \parallel \{M_\sigma \mid \langle\rangle \neq \sigma \in \mathcal{P}\}$, then υ can be Böhmed out.

The conjecture was stated for only one free variable; we briefly consider the case when we have move variables. If we wished to simply "block" the free variables from being used, we could replace them by $\perp$ (by applying the substitution $\vec{y} = \overrightarrow{\Omega}$); this will make all nodes passing through them unaccessible. Alternatively, the variables could be used to help Böhm out the x (making more positions accessible), then they should be put into the head position first. In all other cases, the interaction of the variables will be left subject to how the characterization is to be used.

Assuming the conjecture, we can easily solve all of the problems in Example 89. We illustrate the technique with the last one.

Example 95. Find a closed term X such that

$$x\mathtt{I}(x(x\Omega y)\Omega\mathtt{I})\mathtt{I}[x := X] = y\overrightarrow{N}$$

or prove that no such X exists.

Solution. The path $\pi = \langle 1, 0, 1\rangle$ to the unique occurrence of y passes through 3 head occurrences of x, let's call them π_0, π_1, π_2. By the characterization above, in order for X to exist we must at the very least be able to separate π_1 from the others, since $\langle 0, 1\rangle \nsubseteq \langle 1, 0, 1\rangle$. As the first argument of x at π_2 is Ω, the second argument of x at π_1 is Ω, and at the root (at π_0) x has only three arguments in its scope, we see that any possible filtration must begin with the third. That is,

$$X = \lambda abc.c\ldots \tag{6.8}$$

The scope of x at π_2 can be η-expanded to yield $\lambda z.x\Omega yz$. Because we have control over the third argument of x at π_2, we can consider that position to be effectively separated (when we get to π_2, we will simply instantiate z with what is necessary to get y). But we still need to separate π_0 from π_1, because their third argument is the same while $\pi(0) \neq \pi(1)$. The only candidate is the first argument, which is $\mathtt{I}$ at π_0 and $\lambda z.X\Omega yz$ at π_2. By (6.8), the latter β-reduces to $\lambda z.z\ldots$. Because this has the same head variable as $\mathtt{I}$, we can only separate them by exploiting the difference in their Böhm rank, which is zero for $\mathtt{I}$ and $-n$ for $X\Omega y$, where n is yet to be determined. Then the two are separated by the context $[\ \ ]\mathtt{U}^n_n(\mathtt{U}^n_0 B)\Omega^{\sim n-1}A$. Thus we have

$$X = \lambda abc.c(a\mathtt{U}^n_n(\mathtt{U}^n_0 B)\Omega^{\sim n-1}A)\cdots$$

In the case of π_0, we want to pass control to π_1, which is the second argument. Hence $A = b$. In the case of π_1, we want to pass control to π_2, which is the first argument, and needs to be applied to some term C. Hence $B = aC$:

$$X = \lambda abc.c(a\mathtt{U}^n_n(\mathtt{U}^n_0(aC))\Omega^{\sim n-1}b)\cdots \tag{6.9}$$

Now, C needs to extract y when $X\Omega y$ is applied to it, yielding the term $X\Omega yC = C(\Omega(\mathtt{U}_n^n \cdots))\cdots$. Then the $\cdots$ in (6.9) must include b, so that $X\Omega yC = C(\Omega\cdots)b$. Now we can take $C = \mathtt{U}_1^1$, and obtain

$$X = \lambda abc.c(a\mathtt{U}_n^n(\mathtt{U}_0^n(a\mathtt{U}_1^1))\Omega^{\sim n-1}b)b$$

Here the scope of c has two arguments, so $n = 2$, and we finally get

$$X = \lambda abc.c(a\mathtt{U}_2^2(\mathtt{U}_0^2(a\overline{\mathtt{K}}))\Omega b)b \tag{6.10}$$

Let's verify that the term works as intended:

$$\begin{aligned}
X\mathtt{I}(X(X\Omega y)\Omega\mathtt{I})\mathtt{I} &= \mathtt{I}(\mathtt{I}\mathtt{U}_2^2(\mathtt{U}_0^2(\mathtt{I}\overline{\mathtt{K}}))\Omega(X(X\Omega y)\Omega\mathtt{I}))(X(X\Omega y)\Omega\mathtt{I})\\
&= \mathtt{U}_2^2(\cdots)\Omega(X(X\Omega y)\Omega\mathtt{I})(X\cdots)\\
&= X(X\Omega y)\Omega\mathtt{I}(X\cdots)\\
&= \mathtt{I}(X\Omega y\mathtt{U}_2^2(\mathtt{U}_0^2(X\Omega y\overline{\mathtt{K}}))\Omega\Omega)\Omega(X\cdots)\\
&= (X\Omega y\mathtt{U}_2^2)(\mathtt{U}_0^2(X\Omega y\overline{\mathtt{K}}))\Omega\Omega\Omega(X\cdots)\\
&= (\mathtt{U}_2^2(\Omega\cdots)y)(\mathtt{U}_0^2(X\Omega y\overline{\mathtt{K}}))\Omega\Omega\Omega(X\cdots)\\
&= (\mathtt{U}_0^2(X\Omega y\overline{\mathtt{K}}))\Omega\Omega\Omega(X\cdots)\\
&= (X\Omega y\overline{\mathtt{K}})\Omega(X\cdots)\\
&= (\overline{\mathtt{K}}(\Omega\cdots)y)\Omega(X\cdots)\\
&= y\Omega(X\cdots) = yN_0N_1
\end{aligned}$$

The solution (6.10) is not optimally hygienic: there is some junk following y. Sometimes this junk is unavoidable, but in this case it can be completely eliminated either by taking $C = \mathtt{U}_1^3$ or erasing the left b. □

We return to the discussion of the range property.

In section 6.3, we have seen that for a sequence of λ-terms $\langle M_n\rangle$, $\Lambda^0/\langle M_n\rangle$ is infinite if $\langle M_n\rangle$ is computable and M_n are closed. This left open the question of what happens when M_n has free variables. If the conjecture above is verified, we would be able to pursue a more complete answer.

If $y \in \mathsf{FV}(x\overrightarrow{M})\backslash\{x\}$ occurs at a position which is x-accessible, then the range of $\langle M_n\rangle$ is infinite: given $X = \lambda\vec{a}.a_i\overrightarrow{P}$, with $xM_0\ldots M_{n-1}[x := X] = y\overrightarrow{N}$, then $xM_0\ldots M_n[x := \lambda\vec{a}a'.X\vec{a}\mathsf{c}_k] = y\overrightarrow{N}\mathsf{c}_k$. This is not immediate from the conjecture, but we believe that it would follow from the proof. In any case, it will be possible to extend the arity of X sufficiently far out that it would not interfere with the Böhming out.

If for every n, $i \leqslant n$, no free variable of M_i, other than x, is accessible in $xM_0 \cdots M_n$, then these variables cannot take part in the head reduction of any term plugged in for x. This will make the theorems of Section 6.7 valid for open terms as well, modulo occurrences of x. But the conjecture also provides a way to deal with these occurrences. If it was established, it would give us a tool to study input stream equivalence operators induced by sequences consisting of open terms — which is an open problem.

Yet if the sequence in question comes from a trace of x in M, there is an additional complication. Suppose $\langle M_n \rangle$ is the tunnel at Π, and let $y \in \mathsf{FV}(M_n)$. As noted before, y must be either bound, equal to x, or be a free variable not equal to x. However, the last case might not be stable! The variable y could be bound by a λ occurring *above* the scope of x in Π. If this occurrence ever becomes part of a redex, then y could be instantiated by another term sometime later in the Gross–Knuth sequence of M.

In this case, we really should have defined M_n by the *stable* nodes in its Böhm trees. That is, if a descendant of y at v is eventually instantiated by a solvable term whose head variable is never bound again, then we should call it *the* subterm of M_n at position v. However, this is not guaranteed to happen.

Given a trace Π, call a subterm N of M *ethereal* if every descendant of N has a head normal form with the head variable free in N but bound by an abstraction contracted at some later point in Π. Ethereal terms act a lot like unsolvables: they cannot be given an applicative context which would make them $\mathtt{I}$. Perhaps this illustrates a deep flaw of $\mathcal{H}$ as a λ-theory: that Ω-reduction is not sufficiently general for contexts under reduction, despite the appearence.

If $\mathcal{T}(\Pi)$ has an ethereal subterm occurring at some x-accessible position and Π is imperishable, then the range of the term M is infinite, because such terms act just like “fixed” free variables. The only exception is an ethereal $\mathtt{K}^\infty$: an ethereal subterm that develops infinite λ-prefix while its head variable stays free. If this subterm was accessible, it could consume the whole tunnel without ever becoming unsolvable.

Statman [1993] has suggested that a possible counterexample to the range property for $\mathcal{H}$ could be obtained by constructing solvable Plotkin terms. This is not the approach we took: our counterexample does not send all solvable terms to the same point. The existence of solvable Plotkin terms is thus still an open question. We believe that if Conjecture 94 was established this question could be almost settled negatively. The only mechanism which would remain to be refuted is that of ethereal $\mathtt{K}$-infinities.

However, if a counterexample could indeed be constructed that was based on ethereal $\mathtt{K}^\infty$s, then it would show that the use of recursion theory was *not* necessary to refute the range property. If a construction can be done without using reflection, it would be a serious improvement over our result.

It may seem unlikely that a counterexample could be constructed based merely on ethereal $\mathtt{K}^\infty$s. However, we will now show that this mechanism is actually quite potent. Specifically, we define a term F such that

$$\begin{aligned} F(\lambda x_1 \ldots x_l.x_i M_1 \ldots M_m) &=_{\mathcal{H}} F(\lambda x_1 \ldots x_{l'}.x_j M_1 \ldots M_{m'}) \\ \iff \qquad & m - l = m' - l' \end{aligned}$$

Definition 96. For $M \in \Lambda$, let $\overline{M} = \mathtt{C}M = \lambda xy.Myx$. Put

$$\begin{aligned} \mathtt{\Sigma} px &= \lambda z.z(\mathtt{\Sigma} p(xp)z) \\ Gy &= \lambda z.z(G(y \circ \mathtt{K})z) \\ F &= G \circ \overline{\mathtt{\Sigma}} \end{aligned}$$

Now we compute

$$\begin{aligned} Fx &= G(\overline{\mathtt{\Sigma}}x) \\ &= \lambda z.z(G(\overline{\mathtt{\Sigma}}x \circ \mathtt{K})z) \\ &= \lambda z.z(G(\lambda y.\overline{\mathtt{\Sigma}}x(\mathtt{K}y))z) \\ &= \lambda z.z(z(G(\lambda y.\overline{\mathtt{\Sigma}}x(\mathtt{K}y) \circ \mathtt{K})z)) \\ &= \lambda z.z(z(G(\lambda y.\overline{\mathtt{\Sigma}}x(\mathtt{K}(\mathtt{K}y)))z)) \\ &= \cdots \\ &= \lambda z.z^n(G(\lambda y.\overline{\mathtt{\Sigma}}x(\mathtt{K}^n y))z) \\ &= \lambda z.z^n(G(\lambda y.\mathtt{\Sigma}(\mathtt{K}^n y)x)z) \\ &= \lambda z.z^n(G(\lambda y.\lambda w.w(\mathtt{\Sigma}(\mathtt{K}^n y)(x(\mathtt{K}^n y))w))z) \\ &= \lambda z.z^n(G(\lambda y.\lambda w.w(w(\mathtt{\Sigma}(\mathtt{K}^n y)(x(\mathtt{K}^n y)(\mathtt{K}^n y))w)))z) \\ &= \cdots \\ &= \lambda z.z^n(G(\lambda y.\lambda w.w^m(\mathtt{\Sigma}(\mathtt{K}^n y)(x(\mathtt{K}^n y)^{\sim m})w))z) \end{aligned}$$

Notice that all subterms are solvable in the last term. At the same time, the only trace of x receives a tunnel consisting entirely of ethereal $\mathtt{K}$-infinities, all of them identical. It is straightforward to verify that when M and N have the same Böhm rank, then $FM = FN$. It is less straightforward, but possible using the usual tandem of Church–Rosser and Standardization, to verify that the converse holds as well.

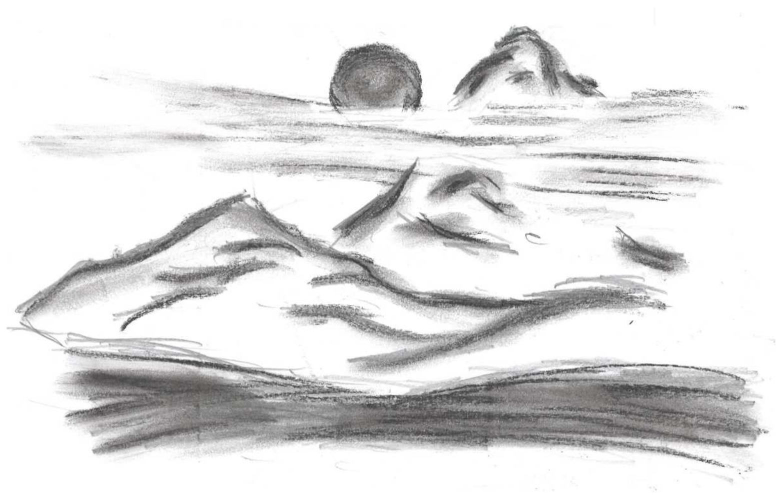
That F equalizes all terms with the same Böhm rank suggests that ethereal terms can have dramatic influence in $\mathcal{H}$. While this does not yet give us a term with finite range, it brings us dangerously close: if we can find a term H such that $x \in_{\mathcal{H}} Hx$ and $Hx = H(\lambda y.x)$, then combining H with the mechanism in F will yield a solvable Plotkin term.

However, we believe that such an H does not exist, for its existence would contradict Conjecture 65. We conjecture that partitioning terms according to their rank is the furthest we can get without using reflection.

In conclusion, we remark that generalizing Corollary 48 to those $\langle M_n \rangle$ which have x as the only free variable is still open.

Problem. Does there exist a computable sequence $\langle M_n \rangle \subseteq \Lambda(x)$ with

$$1 < |\Lambda^0/\{X \sim Y \mid x\overrightarrow{M}[x := X] = x\overrightarrow{M}[x := Y] \text{ for some } \overrightarrow{M} \subseteq \langle M_n \rangle\}| < \omega$$



Chapter 7

Properties of Enumerators

In this chapter we study reflection in the Lambda Calculus from an axiomatic point of view. Specifically, we consider various properties that the quote $\ulcorner\cdot\urcorner$ must satisfy as a function from Λ to Λ. The most important of these is the existence of a definabile left inverse: a term $\mathtt{E}$, called the *evaluator* for $\ulcorner\cdot\urcorner$, that satisfies $\mathtt{E}\ulcorner M\urcorner = M$ for all $M \in \Lambda$.

Usually the quote $\ulcorner M\urcorner$ encodes the syntax of the term M, and the evaluator proceeds by analyzing the syntax and reifying all constructors by their actual meaning in the calculus. Raymond Smullyan [1994] wondered which elements of the syntax must be accessible via the quote in order for an evaluator to exist. He asked three specific questions, to which we provide negative answers.

7.1 Introduction

Reflection is a powerful phenomenon in mathematical logic. Its most dramatic application was by Gödel, who used it in the proof of his famous Incompleteness Theorems, which destroyed Hilbert's formalist program in its original incarnation (one could call the latter *Naïve Formalism.*) Soon after, it was at the heart of the proofs of equivalence between various models of computation that ultimately provided evidence for Church's thesis. In particular, it is the core component of the enumeration theorem, a result used implicitly in virtually every proof of Recursion Theory.

The ability of a computing system to interpret its own syntax also played a significant role in the evolution of functional programming languages. For example, in one of the early reports on the development of Lisp, John McCarthy [1962] introduced the so-called Meta-Circular Evaluator — a Lisp form which can execute an arbitrary list as a Lisp form — a ”universal Lisp form”. Since then, many languages (including Lisp, Prolog, Smalltalk, etc.) have been designed ground-up using a meta-circular implementation. In the other direction, some languages have the “quote” command, which represents expressions of the language within some standard datatype. The presence of an explicit quote operator is one of the reasons why Lisp is not a pure functional language.

The peculiar use of self-reference made Gödel's argument a favorite among philosophers, and inspired a number of publications in popular science, some of which even attribute a certain mystical element to the work of Gödel. For example, in his introduction to *Gödel, Escher, Bach: an Eternal Golden Braid*, Hofstadter [1999] wrote: ”GEB is in essence a long proposal of strange loops as a metaphor for how selfhood originates.” While Hofstadter's thesis is not falsifiable[1], it nevertheless paints a pleasant allegorical picture, and has stimulated much positive interest in Gödel's work.

The questions of Smullyan were brought to our attention by Henk Barendregt. Of course, they are only a sliver in the more global puzzle of understanding reflection as a distinct phenomenon. There is still lacking a general concept, an all-inclusive definition through which the common features of the constructions in Gödel's theorem, computability, number theory (systems of arithmetic), and set theory could be related. Finding such a concept remains a fascinating open problem.

7.2 Coding of lambda terms

The first enumerator for the lambda calculus was constructed by Kleene [1936] in the proof that every lambda-definable function is computable — among the first pieces of evidence for the Church–Turing thesis. Together with the proof that every computable function is lambda-definable, this gave an interpretation of lambda-calculus within itself. Kleene's approach used Gödel's arithmetization of syntax, which codes grammar trees of terms as

[1]If someone proposed a theory of consciousness which fully explained the phenomenon to everyone's satisfaction without a single reference to reflection, we could still say: “But maybe there is *some* model, or theory, which can also explain consciousness, and for which self-reflection is a good metaphor.”

natural numbers. This has the drawback that an evaluator exists only for terms whose free variables come from a finite set which is fixed in advance.

Mogensen [1994] found an elegant self-interpreter which, instead of coding variables by numerals, coded them by themselves. The coding therefore allows an evaluator which is uniform on the set of all (open) lambda terms. Mogensen's construction has a different drawback: it lacks a *discriminator* — a term which can test whether or not two quotes code the same term. However, Barendregt [1994] did find a discriminator for Mogensen coding which works for all closed terms.

A more significant distinction between Kleene's enumerator and Mogensen's is that Kleene actually emulates variable binding within the quotes. This requires a number of auxiliary functions to deal with alpha-conversion, making definitions rather complicated. On the other hand, Mogensen encodes binders by actual "meta-level" lambdas. This technique is known as Higher Order Abstract Syntax (Pfenning and Elliott [1988]), and Mogensen's coding is arguably the most canonical application of it.

Classically, an *enumerator* is a term $\mathtt{E}$ such that every closed lambda term is convertible to $\mathtt{E}\mathtt{c}_n$ for some natural number n. An interesting result on the coding in lambda calculus states that all enumerators are *reducing*: if $\mathtt{E}$ is an enumerator, then $\forall M \in \Lambda^0 \; \exists n \in \mathbb{N} \;$ s.t. $\mathtt{E}\mathtt{c}_n \twoheadrightarrow M$. Richard Statman gave the first proof of this result using computability theory, and Henk Barendregt provided a constructive adaptation, which can be found in the festschrift of Dirk van Dalen (Barendregt [1999]).

To keep matters simple, we will restrict our attention to the coding of *closed* terms, and work in the combinatory version of the lambda calculus with basis $\{\mathtt{K}, \mathtt{S}\}$. This results in no loss of generality, as all closed lambda terms can always be written in this basis. Indeed, our constructions can be translated into Mogensen coding rather explicitly. Furthermore, the choice of basis has no effect on our results.

In what follows, we will need to have a standard, reference coding with which others can be compared. Any of those mentioned above would work; we will use a variant which uses pairing to represent the syntax trees.

Definition 97. (Canonical Quote) For $\overrightarrow{M} \in \Lambda$, write $[M_1, M_2] = \lambda z.zM_1M_2$, $\langle M_1, \cdots, M_n \rangle = \lambda z.z\overrightarrow{M}$, and let M_{CL} denote the combinatory translation of the lambda term M. Now define

- $\underline{M} \equiv (\mathtt{SI}(\mathtt{K}M)) = \langle M \rangle_{CL}$ if $M \in \{\mathtt{K}, \mathtt{S}\}$

- $\underline{MN} \equiv (\lambda xyz.zxy)_{CL}\underline{M}\underline{N} = [\underline{M}, \underline{N}]_{CL}$ for all M, N.

7.3 Axioms for the quote operator

A *coding* is a map from Λ^0_{CL} into itself. Since the primary application of coding is the manipulation of the syntax of terms, most of the properties that we investigate will concern existence of combinators relating the structure of a term to that of its quote. Among these, most attention is given to the Constructor and Destructor axioms. Roughly, the former allows one to obtain the quote of a term from the quotes of its subterms. The latter is dual: it breaks up the term into its subterms (with respect to the quote).

Definition 98. (Coding Axioms) Let $\ulcorner\cdot\urcorner : \Lambda^0_{CL} \to \Lambda^0_{CL}$. We say $\ulcorner\cdot\urcorner$ *satisfies axiom* X from among those below if there exists a combinator X with the stated property.

$$
\begin{array}{lll}
\text{CON } (\textit{constructor}): & \begin{cases} \text{A:} & A\ulcorner M\urcorner\ulcorner N\urcorner = \ulcorner MN\urcorner \\ \text{B:} & B\ulcorner M\urcorner = \ulcorner\ulcorner M\urcorner\urcorner \end{cases} & \\
\text{DES } (\textit{destructor}): & \begin{cases} \text{P:} & P_i\ulcorner M_0M_1\urcorner = \ulcorner M_i\urcorner, \quad i \in \{0,1\} \\ \text{Z:} & Z_b\ulcorner M\urcorner = \begin{cases} \mathtt{K} & M \equiv b \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases} \qquad b \in \{\mathtt{K}, \mathtt{S}\} \end{cases} & \\
\text{CMP } (\textit{complete}): & \begin{cases} \text{U:} & U\ulcorner M\urcorner & = \underline{M} & \text{(uncoding)} \\ \text{U}^{-1}\text{:} & U^{-1}\underline{M} & = \ulcorner M\urcorner & \text{(encoding)} \end{cases} & \\
\text{E } (\textit{evaluator}): & \mathtt{E}\ulcorner M\urcorner = M & \\
\Delta \ (\textit{discriminator}): & \Delta\ulcorner M\urcorner\ulcorner N\urcorner = \begin{cases} \mathtt{K} & M \equiv N \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases} & \\
\text{MON } (\textit{monic}): & \forall M, N \in \Lambda^0_{CL} & \ulcorner M\urcorner = \ulcorner N\urcorner \Longrightarrow M \equiv N \\
\text{SOL } (\textit{solvable}): & \forall M \in \Lambda^0_{CL} & \ulcorner M\urcorner \text{ is solvable}
\end{array}
$$

Remark 99. Smullyan called a coding satisfying CON *admissible*, and a coding satisfying DES *preadmissible*. He asked whether either implies the other, and whether an evaluator can be constructed from CON. All three questions have negative answers.

Remark 100. Axiom B appears to be too strong: if we want to requote M, why should we care about *the particular alpha-equivalence class* of $\ulcorner M \urcorner$? It may be more reasonable to require

$$\mathsf{B}^-: \qquad B^- \ulcorner M \urcorner = \ulcorner N \urcorner, \text{ where } N = \ulcorner M \urcorner$$

Nevertheless, we will proceed with Smullyan's original formulation.

The axioms above are the primary focus of our attention. In studying them, the following auxiliary properties are useful.

Definition 101. We introduce two additional axioms

$$\mathrm{Z}_? \text{ (\textit{leaf test})}: \qquad Z_? \ulcorner M \urcorner = \begin{cases} \mathtt{K} & M \in \{\mathtt{K}, \mathtt{S}\} \\ \overline{\mathtt{K}} & M \equiv M_0 M_1 \end{cases}$$

R_D (*range test*): $\exists$ *c.e.* $D \subseteq \Lambda^0_{CL}$, $\mathsf{Range}(\ulcorner \cdot \urcorner) \subseteq D$, $\exists R_D \in \Lambda^0_{CL}$, $\forall N \in D$:

$$R_D N = \begin{cases} \mathtt{K} & \exists M.\ N = \ulcorner M \urcorner \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases}$$

Proposition 102. *(Elementary properties) Let* $\ulcorner \cdot \urcorner$ *be a coding. Then the following implications hold:*

1. $\ulcorner \cdot \urcorner \equiv \underline{\cdot} \implies \mathrm{CON} \wedge \mathrm{DES} \wedge \mathrm{E} \wedge \Delta$
2. $\mathrm{Z} \implies \mathrm{SOL}, \quad \mathrm{DES} \implies \mathrm{MON}, \quad \Delta \implies \mathrm{MON} \wedge \mathrm{Z}$
3. $\mathrm{CMP} \implies \mathrm{CON} \wedge \mathrm{DES} \wedge \Delta$
4. $\mathrm{DES} \implies \mathrm{U} \implies \mathrm{E}$

Proof. 1. We verify that the standard coding has all of the properties of interest.

- Let $P_0 = \langle \mathtt{K} \rangle_{CL} = \mathtt{SI}(\mathtt{KK})$. Let $P_1 = \langle \overline{\mathtt{K}} \rangle_{CL} = \mathtt{SI}(\mathtt{K}\overline{\mathtt{K}})$. Then

$$\begin{aligned} P_i[M_0, M_1] &= \mathtt{I}[M_0, M_1](\mathtt{K}(\lambda x_0 x_1.x_i)[M_0, M_1]) \\ &= (\lambda x_0 x_1.x_i) M_0 M_1 = M_i \end{aligned}$$

In particular, $P_i \underline{M_0 M_1} = \underline{M_i}$.

- Let $Z_? = (\lambda x.x\mathtt{U}^3_2\mathtt{I})_{CL}$. Then

$$\begin{aligned} Z_?[x] &= \mathtt{K} \\ Z_?[x,y] &= \overline{\mathtt{K}} \end{aligned}$$

  In particular, $Z_?\underline{MN} = Z_?[\underline{M},\underline{N}] = \overline{\mathtt{K}}$, while $Z_?\underline{\mathtt{K}} = Z_?\underline{\mathtt{S}} = \mathtt{K}$.

- With Z$_?$ satisfied, it is trivial to get full Z. Since $\mathtt{K},\mathtt{S}$ are normal forms, by Böhm's theorem (Barendregt [1984]), there exist closed terms $\overrightarrow{Q}$ such that $\mathtt{K}\overrightarrow{Q} = \mathtt{K}$ and $\mathtt{S}\overrightarrow{Q} = \overline{\mathtt{K}}$. Then

$$\begin{aligned} Z_{\mathtt{K}}x &= (\text{IF } Z_?x \text{ THEN } x\overrightarrow{Q} \text{ ELSE } \overline{\mathtt{K}})_{CL} \\ Z_{\mathtt{S}}x &= (\text{IF } Z_?x \text{ THEN } x\overrightarrow{Q}\overline{\mathtt{K}}\mathtt{K} \text{ ELSE } \overline{\mathtt{K}})_{CL} \end{aligned}$$

- So far we have proved that $\underline{\cdot}$ satisfies DES. To show axiom A, let $\mathtt{P} = (\lambda xyz.zxy)_{CL}$ be the pairing function. Explicitly,

$$\mathtt{P} \equiv \mathtt{S(S(KS)(S(KK)(S(KS)(S(K(SI))(S(KK)I)))))(K(S(KK)I)))}$$

  Now $\underline{\cdot}$ satisfies A by taking $A = \mathtt{P}$. Furthermore, this is the representative of the β-equivalence class of $\underline{MN}$ that was chosen by Definition 97, i.e. $\underline{MN} \equiv (\mathtt{P}\,\underline{M})\,\underline{N}$.

- Using a fixed-point combinator, put

$$Bx = (\text{IF } Z_?x \text{ THEN } (Z_{\mathtt{K}}x)\underline{\underline{\mathtt{K}}}\,\underline{\underline{\mathtt{S}}} \text{ ELSE } \mathtt{P}(\mathtt{P}\underline{\mathtt{P}}(B(P_0x)))(B(P_1x)))_{CL}$$

  From now on, we will omit the $(\cdot)_{CL}$ subscript, meaning that any λ-expression is implicitly converted to its combinatory equivalent.

- Evaluator is easy for the standard coding:

$$\mathtt{E}x = \text{IF } Z_?x \text{ THEN } x\mathtt{I} \text{ ELSE } x(\lambda xy.\mathtt{E}x(\mathtt{E}y))$$

- So is the discriminator:

$$\begin{aligned} \Delta xy = &\text{IF } Z_?x \\ &\qquad \text{THEN IF } Z_{\mathtt{K}}x \text{ THEN } Z_{\mathtt{K}}y \text{ ELSE } Z_{\mathtt{S}}y \\ &\qquad \text{ELSE IF } Z_?y \text{ THEN } \overline{\mathtt{K}} \text{ ELSE } (\Delta(P_0x)(P_0y))(\Delta(P_1x)(P_1y))\overline{\mathtt{K}} \end{aligned}$$

2. That Z $\implies$ SOL is an immediate consequence of the Genericity Lemma (Barendregt [1984, 14.3.24]).

   That $\Delta \implies$ MON $\wedge$ Z is also immediate.

   To see that DES $\implies$ MON we proceed by induction on the height of M. The base step is assured by Z. If $M \equiv M_0M_1$, and $N \equiv N_0N_1$, then $(M \not\equiv N) \implies (M_i \not\equiv N_i)$ for some i. If $\ulcorner M \urcorner = \ulcorner N \urcorner$, then applying the i'th projection contradicts the inductive hypothesis.

3. Use translation to $\underline{\cdot}$ and back.

4. Take

$$Ux = \text{IF } Z_?x \text{ THEN } (Z_{\mathtt{K}}x)\underline{\mathtt{K}}\,\underline{\mathtt{S}} \text{ ELSE } A_s(U(P_1x))(U(P_2x))$$

   where A_s is a combinator witnessing axiom A for the standard coding.

   Then take

$$\mathtt{E} = \mathtt{E}_s \circ U$$

   where $\mathtt{E}_s$ is the standard evaluator constructed in part 1. □

Note that if the coding is a constant map, then it satisfies CON but neither Z nor E. Thus, as pointed out by an anonymous referee, two of Smullyan's questions have trivial answers.

A slight modification to the standard coding gives a counterexample that is also monic and solvable.

Theorem 103. *There exists a map* $\ulcorner\cdot\urcorner$ *which satisfies* CON, MON, SOL, *and* P, *yet neither* Z *nor* E. *In particular,* CON $\not\Rightarrow$ DES.

Proof. Define $\ulcorner\cdot\urcorner$ by

- $\ulcorner M \urcorner \equiv [\Omega M]$ if $M \in \{\mathtt{K}, \mathtt{S}\}$
- $\ulcorner MN \urcorner \equiv \mathtt{P}\ulcorner M \urcorner \ulcorner N \urcorner = [\ulcorner M \urcorner, \ulcorner N \urcorner]$

Note that $\ulcorner\cdot\urcorner$ is monic, solvable, and satisfies A, P, and $Z_?$ via the same combinators as the standard coding. The combinator witnessing B must be modified ever so slightly:

$$Bx = \text{IF } Z_?x \text{ THEN } (Z_{\mathtt{K}}x)\ulcorner\ulcorner \mathtt{K} \urcorner\urcorner \ulcorner\ulcorner \mathtt{S} \urcorner\urcorner \text{ ELSE } \mathtt{P}(\mathtt{P}\ulcorner \mathtt{P} \urcorner(B(P_0x)))(B(P_1x))$$

To finish the proof, note that if $Z(\lambda x.x(\Omega M)) = \mathtt{K}$, then by Genericity Lemma (Barendregt [1984, 13.3.24]) we have $Z(\lambda x.x(\Omega M)) = Z(\lambda x.x(\Omega N))$. Therefore, no term can satisfy axiom Z. By the same token, no evaluator can exist, for its value on $\ulcorner \mathtt{K} \urcorner$ would necessarily agree with that on $\ulcorner \mathtt{S} \urcorner$. □

Theorem 104. *There exists a map* $\ulcorner \cdot \urcorner$ *satisfying* Z *and* P *which does not satisfy* A*. Thus* DES $\not\Rightarrow$ CON*. Furthermore,* $\ulcorner \cdot \urcorner$ *is monic and solvable.*

Proof. For $M \in \Lambda^0_{CL}$, let $s(M)$ denote the size of the syntax tree of M, defined inductively by $s(\mathtt{K}) = s(\mathtt{S}) = 1,\ s(MN) = s(M) + s(N) + 1$. Certainly, $s(M)$ can be easily computed from $\underline{M}$:

$$\tilde{s}x = \text{IF } Z_?x \text{ THEN } \mathtt{c}_1 \text{ ELSE } \mathtt{c}_+(\tilde{s}(P_0x))(\tilde{s}(P_1x))$$

where $\mathtt{c}_+$ is the addition on the Church numerals.

For $n \in \mathbb{N}$, let H_n be a lambda term encoding the first n values of the characteristic function of the halting problem. Specifically, we put $H_n = \langle h_0, h_1, \ldots, h_{n-1} \rangle = \lambda z.z\vec{h}$, where

$$h_i = \begin{cases} \mathtt{K} & \varphi_i(i)\downarrow \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases}$$

For $0 \leqslant k \leqslant n$, let Π^n_k be such that $\Pi^n_k\langle M_1, \ldots, M_n\rangle = \langle M_1, \ldots, M_k\rangle$. For example, Π^n_k could be obtained by taking $\Pi^n_k = \Pi \mathtt{c}_n \mathtt{c}_k$, where

$$\Pi nk = \lambda xz.x(k\mathtt{B}(\lambda s.k\langle \mathtt{I}\rangle(n\mathtt{K}s))z)$$

(here $\mathtt{B}$ is the composition combinator $\lambda xyz.x(yz)$.)

Finally, put

$$\ulcorner M \urcorner = [\underline{M}, H_{s(M)}] \tag{7.1}$$

Trivially, MON and SOL are satisfied. To see that this coding satisfies axiom Z, we simply compose the combinator Z_b for the standard coding with the first pair-projection:

$$(\lambda x.Z_b(x\mathtt{K}))\ulcorner M \urcorner = \begin{cases} \mathtt{K} & M \equiv b \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases} \qquad b \in \{\mathtt{K}, \mathtt{S}\}$$

For the ith projection, we use the standard combinator P_i with the auxiliary combinators we defined above:

$$\begin{aligned}
&(\lambda x.(\lambda y.[y, \Pi(\tilde{s}(x\mathtt{K}))(\tilde{s}y)(x\overline{\mathtt{K}})])(P_i(x\mathtt{K})))\ulcorner M_0M_1\urcorner\\
=&(\lambda y.[y, \Pi(\tilde{s}(\ulcorner M_0M_1\urcorner\mathtt{K}))(\tilde{s}y)(\ulcorner M_0M_1\urcorner\overline{\mathtt{K}})])(P_i(\ulcorner M_0M_1\urcorner\mathtt{K}))\\
=&(\lambda y.[y, \Pi(\tilde{s}(\ulcorner M_0M_1\urcorner\mathtt{K}))(\tilde{s}y)(\ulcorner M_0M_1\urcorner\overline{\mathtt{K}})])(P_i([\underline{M_0M_1}, H_{s(M_0M_1)}]\mathtt{K}))\\
=&(\lambda y.[y, \Pi(\tilde{s}(\ulcorner M_0M_1\urcorner\mathtt{K}))(\tilde{s}y)(\ulcorner M_0M_1\urcorner\overline{\mathtt{K}})])(P_i\underline{M_0M_1})\\
=&(\lambda y.[y, \Pi(\tilde{s}\underline{M_0M_1})(\tilde{s}y)(H_{s(M_0M_1)})])\underline{M_i}\\
=&[\underline{M_i}, \Pi(\tilde{s}\underline{M_0M_1})(\tilde{s}\underline{M_i})\langle h_0, \ldots, h_{s(M_0M_1)-1}\rangle]\\
=&[\underline{M_i}, \langle h_0, \ldots, h_{s(M_i)-1}\rangle]\\
=&\ulcorner M_i\urcorner
\end{aligned}$$

Hence (7.1) satisfies DES. But notice that

$$\varphi_e(e)\downarrow \iff \ulcorner\mathtt{K}^e\mathtt{I}\urcorner\overline{\mathtt{K}}\mathtt{U}^e_e = \mathtt{K}$$

Therefore, if there was a combinator for axiom A, we could decide the halting problem by checking whether $\mathtt{c}_e(A\ulcorner\mathtt{K}\urcorner)\ulcorner\mathtt{I}\urcorner\overline{\mathtt{K}}\mathtt{U}^e_e$ equals $\mathtt{K}$ or $\overline{\mathtt{K}}$. Such an A cannot exist. Thus $\ulcorner\cdot\urcorner$ does not satisfy CON. □

Notice that non-computability of the coding $\ulcorner\cdot\urcorner$ was essential in the proof above. Indeed, if the coding was computable, then axiom U^{-1} would be satisfied. By Proposition 102, part 4, the destructor axiom would make the coding complete. Then by part 3, it would satisfy CON.

7.4 Positive results

The next natural question is what additional property could be sufficient for the equivalence CON $\iff$ DES to hold. It turns out that existence of a discriminator goes quite far in this direction.

Theorem 105. $\Delta \wedge \mathrm{U}^{-1} \Longrightarrow \mathrm{U}$.

Proof. To construct U, we need to uniformly enumerate all combinators built up from `K` and `S`. Recall that $[M_n]$ is a *uniform enumeration* of $\{M_n\}$ if for each k, there is some X_k such that

$$[M_n] = [M_0, [M_1, [M_2, \ldots [M_k, X_k] \ldots]$$

That is, $[M_n]$ is an infinite stream whose elements form the sequence $\{M_n\}$. The following functions operate on streams:

$$\begin{aligned} \mathtt{Map} f m &= [f(m\mathtt{K}), \mathtt{Map} f(m\overline{\mathtt{K}})] \\ \mathtt{Fold} f m &= f(m\mathtt{K})(\mathtt{Fold} f(m\overline{\mathtt{K}})) \\ \mathtt{Merge} mn &= [m\mathtt{K}, [n\mathtt{K}, \mathtt{Merge}(m\overline{\mathtt{K}})(n\overline{\mathtt{K}})]] \end{aligned}$$

(These definitions implicitly make use of fixed-point combinators.)

Now we define the *standard enumeration* of CL terms to be

$$\mathscr{C} = [\mathtt{K}, [\mathtt{S}, \mathtt{Fold}\ \mathtt{Merge}\ (\mathtt{Map}\ (\lambda s.\mathtt{Map}\ s\ \mathscr{C})\ \mathscr{C})]]$$

It is straightforward to verify that for each $M \in \Lambda^0_{CL}$ there is a unique n such that $M \equiv C_n$, where $\mathscr{C} = [C_0, [C_1, \ldots]]$. But here we need the combinators to be quoted, hence we define

$$\underline{\mathscr{C}} = [\underline{\mathtt{K}}, [\underline{\mathtt{S}}, \mathtt{Fold}\ \mathtt{Merge}\ (\mathtt{Map}\ (\lambda s.\mathtt{Map}\ (\mathtt{P}s)\ \underline{\mathscr{C}})\ \underline{\mathscr{C}})]]$$

where `P` is the pairing combinator for Proposition 102. (Note that this notation is overloaded; we don't mean that $\underline{\mathscr{C}}$ is the standard quote of $\mathscr{C}$.)

Define

$$U_0 s x = \text{IF } \Delta x(U^{-1}(s\mathtt{K})) \text{ THEN } s\mathtt{K} \text{ ELSE } U_0(s\overline{\mathtt{K}})x$$

$$U = U_0 \underline{\mathscr{C}}$$

Note that $U_0 \underline{\mathscr{C}} \ulcorner M \urcorner = \underline{M}_n$, where $n = (\mu k)(M \equiv M_k \in \mathscr{C})$. Thus

$$U \ulcorner M \urcorner \equiv \underline{M}$$

□

Theorem 106. *Suppose* $\ulcorner\cdot\urcorner$ *is a coding which satisfies* Δ*. Then* $\mathrm{U}^{-1} \iff \mathrm{A}$*. In particular,* $\mathrm{CON} \wedge \Delta \implies \mathrm{DES}$*.*

Proof. ($\Rightarrow$) By the theorem above and Proposition 102.3,
$\Delta \wedge \mathrm{U}^{-1} \implies \mathrm{CMP} \implies \mathrm{CON} \wedge \mathrm{DES} \implies \mathrm{A}$.

($\Leftarrow$) Suppose Δ and A are satisfied. Put

$$\begin{aligned} U^{-1}x = \;& \textsc{if } Z_?x \\ & \textsc{then if } Z_{\mathtt{K}}x \textsc{ then } \ulcorner\mathtt{K}\urcorner \textsc{ else } \ulcorner\mathtt{S}\urcorner \\ & \textsc{else } A(U^{-1}(P_0x))(U^{-1}(P_1x)) \end{aligned}$$

By induction, $U^{-1}\underline{M} = \ulcorner M\urcorner$, hence U^{-1} is satisfied. □

It remains to consider the question of reconstructing the quote from the Destructor axioms. The problem with using the approach of Theorem 105, where we try to "guess" the quote of a term by comparing every possibility to the input, is that we have no information on the space of these possibilities. This is where the range test comes in. Recall that the statement of this axiom is

$$\exists \text{ c.e. } D \subseteq \Lambda^0_{CL},\ \mathsf{Range}(\ulcorner\cdot\urcorner) \subseteq D,\ \exists R_D \in \Lambda^0_{CL},\ \forall N \in D :$$
$$R_DN = \begin{cases} \mathtt{K} & \exists M.\ N = \ulcorner M\urcorner \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases}$$

With the axiom above, we state the final theorem.

Theorem 107. $\mathrm{DES} \wedge \mathrm{R}_D \implies \mathrm{U}^{-1}$.

Proof. As in the proof of Theorem 105, we construct $U^{-1}x$ by looking at all possibilities until we find one that matches x, according to the standard discriminator. Let D enumerate a superset of $\mathsf{Range}(\ulcorner\cdot\urcorner)$. We receive this fact as a uniform enumeration $D = [M_n]$, such that for each n one has $R_DM_n = \mathtt{K}$ if $M_n = \ulcorner N\urcorner$ and $R_DM_n = \overline{\mathtt{K}}$ otherwise. Furthermore, every quote $\ulcorner N\urcorner$ appears in the list D: $\forall N \exists n\ \ulcorner N\urcorner = M_n$.

As per Proposition 102.d, let U witness axiom U for $\ulcorner\cdot\urcorner$. Now put

$$U^{-1}x = \mathtt{Fold}\ (\lambda ht.\ \textsc{if } (R_Dh)(\Delta x(Uh))\overline{\mathtt{K}} \textsc{ then } h \textsc{ else } t)\ D$$

It is routine to verify that $U^{-1}\underline{M} = \ulcorner M\urcorner$ for each M. □

Corollary 108. *For a complete coding, one of the following suffices:*

$$\begin{aligned} \mathrm{A} &\wedge \Delta \\ \mathrm{U}^{-1} &\wedge \Delta \\ \mathrm{U}^{-1} &\wedge \mathrm{DES} \\ \mathrm{R}_D &\wedge \mathrm{DES} \end{aligned}$$

Proof. By the theorems 102 through 107. □

Chapter 8

Principal Subtyping of Lambda Terms

The *Principal Type Theorem*, usually attributed to Curry, Hindley, and Milner (Barendregt et al. [2011]) is a fundamental result in the simple theory of types. It states that every lambda term which can be given a simple type has a so-called *principal* type, which is the most general type this term can have. The Hindley–Milner type inference algorithm reduces the problem of finding this type to first-order unification.

By adding equations between simple types one obtains the *recursive types*. While they are more complicated than their simple counterparts, there is still a notion of a most general recursive type, the *principal recursive type* of a term M. Furthermore, this principal recursive type can be found by turning off the "occurs check" in the unification subprocedure of the Hindley–Milner algorithm. (The occurs check is a condition which prevents the solution from having cyclical terms.) Thus, ironically, it is more simple to find a principal recursive type than to find a principal simple type!

Because the simple types are generated by a single binary constructor (the arrow symbol), any set of unification constraints has a solution if fixed-points are allowed. Hence, every lambda term can be given a recursive type. Nevertheless, recursive types can be very useful in characterizing untyped lambda terms. As we will see, they naturally induce a partial order on Λ, which stays non-trivial even on the set of unsolvable terms.

Here we will extend the above results to theories of subtyping. Like simple types, subtyping theories characterize computational properties of lambda terms. The recursive types extend the characterization via simple types, while subtyping theories further refine it. These results were included in the new book by Barendregt et al. [2011], with simpler proofs due to the authors.

8.1 Types, recursive types, subtyping theories

This section defines the basic concepts which we work with. Our definitions closely follow Barendregt et al. [2011].

8.1.1 Simple types

Let a set $\mathbb{A}$ be fixed.

Definition 109. The set of *types over* $\mathbb{A}$ is given by the grammar

$$\mathbb{T} ::= \mathbb{A} \mid \mathbb{T} \to \mathbb{T}$$

$$\frac{}{\Gamma \vdash x : A \qquad (x : A) \in \Gamma}$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$$

Figure 8.1: The system $\boldsymbol{\lambda}_{\to}$

Definition 110. (Type assignment)

1. A (typing) *judgement* is an expression of the form $M : A$, where $M \in \Lambda$ and $A \in \mathbb{T}$. M is the *subject* of the judgement, and A is the *predicate*.

2. A *declaration* is a judgement whose subject is a variable. A *basis* is a finite set of declarations with distinct subjects.

3. The relation $\vdash \subseteq \{\Gamma \mid \Gamma \text{ is a basis}\} \times \Lambda \times \mathbb{T}$ is defined by induction according to the rules in Figure 8.1. When $(\Gamma, M, A) \in \vdash$, we say that *the statement* $M : A$ *is derivable from the basis* Γ, and write $\Gamma \vdash M : A$.

The most trivial semantics for simple types is obtained by interpreting the atomic types $\alpha \in \mathbb{A}$ as some arbitrary sets and the arrow type $A \to B$ by the full function space between the sets denoted by A and B.

Definition 111. Let $\{X_\alpha \mid \alpha \in \mathbb{A}\}$ be a family of sets. The *full type structure* over $\{X_\alpha\}$, written $\{\mathscr{X}(A)\}_{A\in\mathbb{T}}$ is defined inductively as follows:

1. $\mathscr{X}(\alpha) = X_\alpha$ for $\alpha \in \mathbb{A}$,
2. $\mathscr{X}(A \to B) = \{f \mid f : \mathscr{X}(A) \to \mathscr{X}(B)\}$.

With the above definition, all λ-terms which can be assigned type A by the system $\boldsymbol{\lambda}_\to$ can be interpreted as elements of $\mathscr{X}(A)$. If A is a function type $A_0 \to A_1$, and $M : A$, then $[\![M]\!]$ is a function from $\mathscr{X}(A_0)$ to $\mathscr{X}(A_1)$.

$$A = A \quad \text{(Refl)}$$

$$\frac{A = B}{B = A} \quad \text{(Sym)}$$

$$\frac{A = B \qquad B = C}{A = C} \quad \text{(Trans)}$$

$$\frac{A = A' \qquad B = B'}{A \to B = A' \to B'} \quad \text{(Arrow-=)}$$

Figure 8.2: Closure laws for type theories

8.1.2 Recursive types

Definition 112. Let $\mathbb{T}$ be fixed as above.

1. A *type equation* is an expression of the form $A = B$, with $A, B \in \mathbb{T}$.
2. A *simultaneous recursion* (s.r.) is a finite set of type equations.

3. A *type theory* is a set of type equations which is closed under the rules in Figure 8.2. (In other words, T is a type theory if and only if T is a congruence on $(\mathbb{T}, \to)$.)

4. If E is a set of type equations, then *the type theory generated by* E is

$$T(E) = \bigcap\{T \mid T \text{ is a type theory and } E \subseteq T\}$$

Equations of the type theory generated by an s.r. can be used in deriving a type for a lambda term.

$$\frac{}{E; \Gamma \vdash x : A \qquad (x : A) \in \Gamma}$$

$$\frac{E; \Gamma \vdash M : A \to B \qquad E; \Gamma \vdash N : A}{E; \Gamma \vdash MN : B}$$

$$\frac{E; \Gamma, x : A \vdash M : B}{E; \Gamma \vdash \lambda x.M : A \to B}$$

$$\frac{E; \Gamma \vdash M : A \qquad (A = B) \in T(E)}{E; \Gamma \vdash M : B}$$

Figure 8.3: The system $\boldsymbol{\lambda}^{E}_{=}$

Definition 113. Let E be a set of type equations, and Γ a basis. The set of *lambda terms typable in the context* Γ *using equations from* E is defined by induction via the deduction system given in Figure 8.3.

The system $\boldsymbol{\lambda}^{E}_{=}$ can be much stronger than $\boldsymbol{\lambda}_{\to}$ given sufficiently rich E. For example, if there are types A, B with $(A = A \to B) \in T(E)$, then $E; \varnothing \vdash \Omega : B$, where $\Omega = (\lambda x.xx)(\lambda x.xx)$. In contrast, Ω cannot be assigned any type in $\boldsymbol{\lambda}_{\to}$, because this system has the strong normalization property, while Ω has no normal form.

It is more convenient to talk about type theories using the language of type algebras.

Definition 114. A *type algebra* $\mathcal{A}$ is a pair $(A, \to)$ where A is a set and $\to$ is a binary operation on A.

An example of a type algebra is $(\mathbb{T}, \to)$; this is called the *free type algebra over* $\mathbb{A}$. Another example is $(\mathbb{T}/{\approx}, \to_{\approx})$, where $\approx$ is a congruence on $\mathbb{T}$ with respect to $\to$; this is called the *syntactic type algebra over* $\mathbb{T}$ *with congruence* $\approx$, or the *quotient algebra* $(\mathbb{T}, \to)$ *modulo* $\approx$. Barendregt et al. [2011] show that every type algebra is isomorphic to a syntactic type algebra over $\mathbb{T}$ for some $\mathbb{A}$ and an appropriate type theory $\approx$. (Basically, this is just the First Isomorphism Theorem for binary operator algebras.)

$$\frac{}{\Gamma \vdash_{\mathcal{A}} x : a \qquad a \in A, (x : a) \in \Gamma}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} M : a \to_{\mathcal{A}} b \qquad \Gamma \vdash_{\mathcal{A}} N : a}{\Gamma \vdash_{\mathcal{A}} MN : b}$$

$$\frac{\Gamma, x : a \vdash_{\mathcal{A}} M : a}{\Gamma \vdash_{\mathcal{A}} \lambda x.M : a \to_{\mathcal{A}} b}$$

Figure 8.4: The system $\boldsymbol{\lambda}_{=}^{\mathcal{A}}$

Definition 115. Let $\mathcal{A}$, $\mathcal{B}$ be type algebras.

1. A *homomorhpism* from $\mathcal{A}$ to $\mathcal{B}$ is a function $h : A \to B$ such that

$$\forall a, a' : A \quad h(a \to_{\mathcal{A}} a') = h(a) \to_{\mathcal{B}} h(a')$$

2. Let h be a morphism from $\mathcal{A}$ to $\mathcal{B}$. The *kernel* of h is the set $\ker h = \{(a, a') \in A \times A \mid h(a) = h(a')\}$. It is routine to verify that the kernel of a homomorphism is always a congruence. (Hence $\mathcal{A}/\ker h$ is a type algebra, and if h is surjective it is in fact isomorphic to $\mathcal{B}$.)

3. Let $\rho : \mathbb{A} \to A$ be any map. The *homomorphism* $\tilde{\rho}$ *induced by* ρ is defined in the obvious way by taking $\tilde{\rho}(\alpha) = \rho(\alpha)$ for $\alpha \in \mathbb{A}$ and $\tilde{\rho}(a \to b) = \tilde{\rho}(a) \to_{\mathcal{A}} \tilde{\rho}(b)$. Then $\tilde{\rho}$ is a morphism from $(\mathbb{T}, \to)$ to $\mathcal{A}$.

Obviously, the point of type algebras is that they allow us to interpret type theories.

Definition 116. Let T be a type theory. The *type algebra generated by* T is the quotient algebra $\mathcal{A}(T)$ of $(\mathbb{T}, \to)$ modulo T.

Definition 117. Let $\mathcal{A}$ be a type algebra, and let T be a type theory.

1. A *type environment in* $\mathcal{A}$ is a map $\rho : \mathbb{A} \to A$.
2. ρ *justifies* T *in* $\mathcal{A}$ if $\forall (a = b) \in T \quad \tilde{\rho}(a) = \tilde{\rho}(b)$.
3. $\mathcal{A}$ *satisfies* T if there exists a ρ that justifies T in $\mathcal{A}$.

Fact. *$\mathcal{A}$ satisfies T $\iff$ $\exists h : \mathcal{A}(T) \to \mathcal{A}$.*

The system $\boldsymbol{\lambda}^{E}_{=}$ can be translated into the language of type algebras.

Definition 118. Let $\mathcal{A}$ be a type algebra. The set of terms typable by elements of $\mathcal{A}$ is given by the deduction system in Figure 8.4.

Type algebras make it easy to talk about recursive types.

Definition 119. Let E be a s.r. A *recursive type* (with respect to E) is an element of the type algebra $\mathbb{T}/T(E)$.

$$A \leqslant A \quad \text{(Refl)}$$

$$\frac{A \leqslant B \qquad B \leqslant C}{A \leqslant C} \quad \text{(Trans)}$$

$$\frac{A' \leqslant A \qquad B \leqslant B'}{A \to B \leqslant A' \to B'} \quad \text{(Arrow-}\leqslant\text{)}$$

Figure 8.5: Closure laws for subtyping theories

8.1.3 Subtyping

Definition 120. Let $\mathbb{A}$, $\mathbb{T}$ be as before.

1. A *type inequality* is an expression of the form $A \leqslant B$, where $A, B \in \mathbb{T}$.
2. A *subtyping theory* is a set of type inequalities closed under the rules in Figure 8.5.

3. If I is a set of type inequalities, then *the subtyping theory generated by* I is the set

$$T(I) = \bigcap \{T \mid T \text{ is a subtyping theory and } I \subseteq T\}$$

Definition 121. A *type structure* is a triple $\mathcal{S} = (S, \to, \leqslant)$, where $(S, \to)$ is a type algebra, $(S, \leqslant)$ is a poset, and the following axiom is satisfied:

$$\forall s, s', t, t' \in S \quad s' \leqslant s \wedge t \leqslant t' \Longrightarrow s \to t \leqslant s' \to t'$$

An example of a type structure is $\mathcal{S}_0 = (\mathbb{T}, \to, =)$, which is trivial because the poset $(\mathbb{T}, =)$ is trivial. Nevertheless, we still call it *the free type structure over* $\mathbb{A}$. A better example is the following:

Definition 122. Let T be a subtyping theory. Write $a \sim_T b$ if both $(a \leqslant b)$ and $(b \leqslant a)$ are elements of T. Then $\sim_T$ is a type theory. The *type structure generated by* T is the triple $\mathcal{S}(T) = (S, \to_{\mathcal{S}}, \leqslant_{\mathcal{S}})$, where $(S, \to_{\mathcal{S}}) = (\mathbb{T}, \to)/{\sim_T}$, and $[a]_{\sim_T} \leqslant_{\mathcal{S}} [b]_{\sim_T}$ if and only if $(a \leqslant b) \in T$. (It is easy to check that this is well defined.)

$$\frac{}{\Gamma \vdash_{\mathcal{S}} x : a} \quad a \in S, (x : a) \in \Gamma$$

$$\frac{\Gamma \vdash_{\mathcal{S}} M : a \to_{\mathcal{S}} b \qquad \Gamma \vdash_{\mathcal{S}} N : a}{\Gamma \vdash_{\mathcal{S}} MN : b}$$

$$\frac{\Gamma, x : a \vdash_{\mathcal{S}} M : a}{\Gamma \vdash_{\mathcal{S}} \lambda x.M : a \to_{\mathcal{S}} b}$$

$$\frac{\Gamma \vdash_{\mathcal{S}} M : a \qquad (a \leqslant_{\mathcal{S}} b)}{\Gamma \vdash_{\mathcal{S}} M : b}$$

Figure 8.6: The system $\boldsymbol{\lambda}^{\mathcal{S}}_{\leqslant}$

The partial order of type structures refines the typability hierarchy of its underlying type algebra.

Definition 123. Let $\mathcal{S}$ be a type structure. The set of λ-terms *typable by elements of* $\mathcal{S}$ is defined inductively by the system $\boldsymbol{\lambda}^{\mathcal{S}}_{\leqslant}$ in Figure 8.6.

The system could also be presented in terms of a set of type inequalities I, then one obtains the system shown in Figure 8.7 below. However, we prefer the approach via type structures, since it is more flexible.

$$\frac{}{I;\Gamma \vdash x : A \qquad (x : A) \in \Gamma}$$

$$\frac{I;\Gamma \vdash M : A \to B \qquad I;\Gamma \vdash N : A}{I;\Gamma \vdash MN : B}$$

$$\frac{I;\Gamma, x : A \vdash M : B}{I;\Gamma \vdash \lambda x.M : A \to B}$$

$$\frac{I;\Gamma \vdash M : A \qquad (A \leqslant B) \in T(I)}{I;\Gamma \vdash M : B}$$

Figure 8.7: The system $\boldsymbol{\lambda}^{I}_{\leqslant}$

Definition 124. A *homomorphism* between type structures $\mathcal{S}$ and $\mathcal{S}'$ is a map $h : S \to S'$ which is monotone with respect to $\leqslant$ and is a homomorphism of the underlying type algebras.

Definition 125. Let T be a subtyping theory, and $\mathcal{S}$ a type structure.

1. A *type environment in* $\mathcal{S}$ is any map $\rho : \mathbb{A} \to S$. As before, this map trivially extends to a type structure morphism $\tilde{\rho} : (\mathbb{T}, \to, =) \to \mathcal{S}$.

2. ρ *justifies* T *in* $\mathcal{S}$ if $\forall (A \leqslant B) \in T \quad \tilde{\rho}(A) \leqslant_{\mathcal{S}} \tilde{\rho}(B)$.

3. $\mathcal{S}$ *satisfies* T if there exists a ρ that justifies T in $\mathcal{S}$.

Fact. *$\mathcal{S}$ satisfies $T \iff \exists h : \mathcal{S}(T) \to \mathcal{S}$.*

To show the reader how routine the proofs of these facts really are, we give the proof of the one above.

Proof. ($\Leftarrow$) Let $h : \mathcal{S}(T) \to \mathcal{S}$ be a morphism. Then $(\lambda a : \mathbb{A}.h([a]_T))$ is a type environment justifying T in $\mathcal{S}$.

($\Rightarrow$) Suppose that $\mathcal{S}$ satisfies T via ρ. We will show that $\tilde{\rho}$ is constant on the $\sim_T$-classes of $\mathbb{T}$. It will then be well-defined as a morphism from $\mathcal{S}(T)$.

Suppose that $(A \leqslant B) \in T$ and $(B \leqslant A) \in T$. Since ρ justifies T in $\mathcal{S}$, $\tilde{\rho}(A) \leqslant_{\mathcal{S}} \tilde{\rho}(B)$, and $\tilde{\rho}(B) \leqslant_{\mathcal{S}} \tilde{\rho}(A)$. Since $\mathcal{S}$ is a poset, $\tilde{\rho}(A) = \tilde{\rho}(B)$. Thus $\tilde{\rho}$ is a type algebra morphism from $(\mathbb{T}, \to)/{\sim_T}$ to $(S, \to_{\mathcal{S}})$, and is also monotone with respect to $\leqslant_{\mathcal{S}}$ because ρ justifies T. $\square$

8.2 Principal types

We recall the principal type theorem for simple and recursive types.

Definition 126. Let $\mathbb{A}$ and $\mathbb{T}$ be fixed as before.

1. A *substitution* is a partial map from $\mathbb{A}$ to $\mathbb{T}$.
2. If σ is a substitution, we write $\sigma(A)$ for the type obtained from A by replacing every $\alpha \in \mathsf{dom}(\sigma)$ by $\sigma(\alpha)$. (Basically, we just extend σ to a homomorphism from $\mathbb{T}$ to $\mathbb{T}$ by making it act as identity on atoms outside its domain.)
3. A type A is *more general* than B if there exists a substitution σ such that $B = \sigma(A)$.

Theorem 127. *(Principal Type Theorem, Curry–Hindley) For each $M \in \Lambda$ there exists a basis Γ and a type A such that*

1. *$\Gamma \vdash M : A$*
2. *If $\Gamma' \vdash M : A'$, then there exists a substitution σ such that $\Gamma' \supseteq \sigma(\Gamma)$, and $A' = \sigma(A)$.*

Furthermore, the pair Γ, A can be found in linear time.

The type A in the theorem above is called *the principal type of M*. It is unique modulo renaming of type atoms. The theorem states that the principal type of M is more general than any other type A' with $\Gamma' \vdash M : A'$.

Definition 128. A type algebra $\mathcal{A}$ is *weaker* than $\mathcal{A}'$, writing $\mathcal{A} \preccurlyeq \mathcal{A}'$, if for any type theory T

$$\mathcal{A} \text{ satisfies } T \Longrightarrow \mathcal{A}' \text{ satisfies } T$$

Fact. *For type algebras $\mathcal{A}$, $\mathcal{A}'$ we have*

$$\mathcal{A} \preccurlyeq \mathcal{A}' \iff \exists h : \mathcal{A} \to \mathcal{A}'$$

Theorem 129. *(Principal Recursive Type Theorem) For each $M \in \Lambda$ there exists a type algebra $\mathcal{A}$, a basis Γ, and a type $a \in \mathcal{A}$ such that*

1. $\Gamma \vdash_{\mathcal{A}} M : a$
2. *If $\Gamma' \vdash_{\mathcal{A}'} M : a'$, then $\mathcal{A} \preccurlyeq \mathcal{A}'$. Furthermore, there exists a morphism $h : \mathcal{A} \to \mathcal{A}'$ such that $\Gamma' \supseteq h(\Gamma)$, and $a' = h(a)$.*

The triple $(\mathcal{A}, \Gamma, a)$ is called *the principal triple of M*. For the proof of this theorem, see Barendregt et al. [2011].

In the following section, we provide an analogue of Theorem 129 for subtyping theories.

8.3 The principal subtyping theorem

Definition 130. Suppose $\mathcal{S}$, $\mathcal{S}'$ are type structures. We say that $\mathcal{S}$ is *weaker* than $\mathcal{S}'$, writing $\mathcal{S} \preccurlyeq \mathcal{S}'$, if, for any subtyping theory T,

$$\mathcal{S} \text{ satisfies } T \Longrightarrow \mathcal{S}' \text{ satisfies } T.$$

Proposition 131. $\mathcal{S} \leqslant \mathcal{S}' \iff \exists h : \mathcal{S} \to \mathcal{S}'$.

Proof. ($\Rightarrow$) Let $h : \mathcal{S} \to \mathcal{S}'$ be a morphism. Let T be a subtyping theory and $\rho : \mathbb{A} \to S$ an environment justifying T in $\mathcal{S}$. Then $h \circ \rho$ justifies T in $\mathcal{S}'$.

($\Leftarrow$) Suppose $\mathcal{S} \leqslant \mathcal{S}'$.

Consider $\mathbb{T}^S$ — the set of types over atoms from S. Define $[\![\ ]\!] : \mathbb{T}^S \to \mathcal{S}$ by

- $[\![s]\!] = s$
- $[\![A \to B]\!] = [\![A]\!] \to_{\mathcal{S}} [\![B]\!]$

Now put $T = \{A \leqslant B \mid A, B \in \mathbb{T}^S \text{ and } [\![A]\!] \leqslant_{\mathcal{S}} [\![B]\!]\}$ Clearly, T is satisfiable in $\mathcal{S}$ via $[\![\ ]\!]$. Since $\mathcal{S}$ is weaker than $\mathcal{S}'$, let $\rho : S \to S'$ be an environment justifying T in $\mathcal{S}'$. ρ is the required homomorphism. □

Definition 132. Let T, T' be subtyping theories. Say that T is *weaker* than T', writing $T \leqslant T'$, whenever $\mathcal{S}(T)$ is weaker than $\mathcal{S}(T')$.

Proposition 133. *Let σ be a substitution. Then T is weaker than $\sigma(T)$.*

Proof. Let ρ justify a subtyping theory T' in $\mathcal{S}(T)$. Then $\tilde{\sigma} \circ \rho$ justifies it in $\mathcal{S}(\sigma(T))$. □

Recall that a lambda term M is typable in a type structure $\mathcal{S}$ if there exists a context Γ and a type A such that $\Gamma \vdash_{\mathcal{S}} M : A$ according to the rules given in Figure 8.6.

Proposition 134. *Suppose that $\mathcal{S} \leqslant \mathcal{S}'$ and M is typable in $\mathcal{S}$. Then M is typable in $\mathcal{S}'$.*

Proof. Let $h : \mathcal{S} \to \mathcal{S}'$ be a homomorphism. By induction on the derivation of $\Gamma \vdash_{\mathcal{S}} M : A$, we see that $h(\Gamma) \vdash_{\mathcal{S}'} M : h(A)$. □

Theorem 135. *For every lambda term M there exists a subtyping theory T such that*

$$M \text{ is typable in } \mathcal{S} \iff \mathcal{S}(T) \leqslant \mathcal{S}$$

Proof. We define the subtyping theory T_M, context Γ_M, and type A_M by induction on the structure of M:

- $M = x$. Then
$$(T_M, \Gamma_M, A_M) = (\varnothing, \{x : \alpha_x\}, \alpha_x)$$

- $M = M_0 M_1$. Given $(T_{M_0}, \Gamma_{M_0}, A_{M_0})$ and $(T_{M_1}, \Gamma_{M_1}, A_{M_1})$, put
$$(T_M, \Gamma_M, A_M) = (T_{M_0} \cup T_{M_1} \cup \{A_{M_0} \leqslant (A_{M_1} \to \beta_M)\}, \Gamma_{M_0} \cup \Gamma_{M_1}, \beta_M)$$

- $M = \lambda x.N$. Given (T_N, Γ_N, A_N), put
$$(T_M, \Gamma_M, A_M) = (T_N, \Gamma_N \backslash \{x : \alpha_x\}, \alpha_x \to A_N)$$

Note that the declarations $x : A$ are only introduced into Γ_M when M is a variable. Thus we have the following consequence of the above definition:

$$(x : A) \in \Gamma_M \Longrightarrow x \in M \Longrightarrow A = \alpha_x \tag{8.1}$$

We also assume that different bound variables have distinct names and that the Variable Convention is observed (otherwise, we first α-convert M).

Now we show that $T = T_M$ has the required property.

Suppose $\mathcal{S}(T) \leqslant \mathcal{S}$. By the previous proposition, it suffices to show that M is typable in $\mathcal{S}(T)$. We show by induction on M that

$$T_M; \Gamma_M \vdash M : A_M \tag{8.2}$$

- $M = x$. By a single application of the variable rule,
$$\varnothing; x : \alpha_x \vdash x : \alpha_x$$

- $M = M_0 M_1$. By the inductive hypothesis, we can complete the following derivation:

$$\dfrac{\dfrac{\dfrac{\vdots}{T_{M_0}; \Gamma_{M_0} \vdash M_0 : A_{M_0}}}{T_{M_0}, A_{M_0} \leqslant (A_{M_1} \to \beta_M); \Gamma_{M_0} \vdash M_0 : A_{M_1} \to \beta_M} \qquad \dfrac{\vdots}{T_{M_1}; \Gamma_{M_1} \vdash M_1 : A_{M_1}}}{T_{M_0} \cup \{A_{M_0} \leqslant (A_{M_1} \to \beta_M)\} \cup T_{M_1}; \Gamma_{M_0} \cup \Gamma_{M_1} \vdash M_0 M_1 : \beta_M}$$

- $M = \lambda x.N$. By the inductive hypothesis, we have $T_N; \Gamma_N \vdash N : A_N$. Taking note of (8.1), we write

  $$T_N; \Gamma_N, x : \alpha_x \vdash N : A_N$$

  By rule ($\to$I), we get

  $$T_N; \Gamma_N \backslash \{x : \alpha_x\} \vdash \lambda x.N : \alpha_x \to A_N$$

In all cases above, (8.2) is verified. It follows that $\Gamma_M \vdash_{\mathcal{S}(T)} M : A_M$.

Conversely, suppose that $\Delta \vdash_{\mathcal{S}} M : B$. We are to show that $\mathcal{S}(T) \leqslant \mathcal{S}$. By Barendregt et al. [2011, 11A.18] we may assume that $\mathsf{FV}(\Delta) = \mathsf{FV}(M) = \mathsf{FV}(\Gamma)$.

By Proposition 131, it suffices to find a type environment $\rho : \mathbb{A} \to S$ which justifies T in $\mathcal{S}$. We will extract ρ from the derivation of $\Delta \vdash_{\mathcal{S}} M : B$. This too will be done by induction.

Let π be a derivation of $\Delta \vdash_{\mathcal{S}} M : B$. For $N \subseteq M$, let π_N be the smallest subderivation of π whose root (last typing judgement) contains N as the subject. (Even if N occurs several times in M as a subterm, each such occurrence has a unique corresponding subderivation.) Since the premise of the ($\leqslant$) rule has the same subject as its conclusion, the minimality of π_N implies that its last rule is not ($\leqslant$), but is uniquely determined by the top-level term constructor of N. Let $\sigma(N) \in S$ denote the type that is assigned to N at the root of π_N.

We define a partial map $\rho_N : \mathbb{A} \to S$ by induction on N:

- $N = x$. We have two possibilities.

  1. $x \in \mathsf{FV}(M)$. Then $\rho_N = (\alpha_x, \Delta(x))$.
  2. x is bound by the top λ in $(\lambda x.N') \subseteq M$. Then $\rho_N = (\alpha_x, A)$, where $\sigma(\lambda x.N') = A \to B$.

- $N = N_0 N_1$. Then $\rho_N = \rho_{N_0} \cup \rho_{N_1} \cup \{(\beta_N, \sigma(N))\}$

- $N = \lambda x.N'$. Then $\rho_N = \rho_{N'}$.

Finally, put

$$\rho = \bigcup_{N \subseteq M} \rho_N = \rho_M$$

An important consequence of the definition above is that for $N \subseteq M$,

$$\tilde{\rho}(A_N) \leqslant \sigma(N) \tag{8.3}$$

(Notice that we actually have equality in all cases except abstraction.)

It remains to show that ρ justifies T in $\mathcal{S}$. Since $N \subseteq M \implies T_N \subseteq T_M$, this too can be done by induction on N:

- $N = x$. Then $T_N = \varnothing$, so certainly ρ justifies T_N in $\mathcal{S}$.
- $N = \lambda x.N'$. By IH, ρ justifies $T_{N'}$. Then certainly ρ justifies $T_N = T_{N'}$.
- $N = N_0N_1$. This is the interesting case. By induction, we know that ρ justifies both T_{N_0} and T_{N_1}. To see that ρ justifies T_N, it remains to show that $\tilde{\rho}(A_{N_0}) \leqslant_{\mathcal{S}} \tilde{\rho}(A_{N_1} \to \beta_N)$

  By definition of $\sigma(N)$, we know that π_N must have the form

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \hline T_N;\Gamma_N \vdash N_0 : \sigma(N_0)\\ \hline \vdots\end{array}\leqslant}{T_N;\Gamma_N \vdash N_0 : B \to \sigma(N)}\leqslant \qquad \dfrac{\begin{array}{c}\vdots\\ \hline T_N;\Gamma_N \vdash N_1 : \sigma(N_1)\\ \hline \vdots\end{array}\leqslant}{T_N;\Gamma_N \vdash N_1 : B}\leqslant}{T_N;\Gamma_N \vdash N_0N_1 : \sigma(N)}\to\mathrm{E}$$

  Since the only inferences occurring in π_N between its last inference ($\to$E) and the root of π_{N_0} are ($\leqslant$)-rules, we have $\sigma(N_0) \leqslant (B \to \sigma(N))$, and similarly $\sigma(N_1) \leqslant B$. Hence we have

$$\begin{aligned}\tilde{\rho}(A_{N_0}) &\leqslant_{(8.3)} \sigma(N_0)\\ &\leqslant B \to \sigma(N) =_{(\mathrm{def})} B \to \rho(\beta_N)\\ &\leqslant_{(\mathrm{Arrow}-\leqslant)} \sigma(N_1) \to \rho(\beta_N)\\ &\leqslant_{(8.3)} \tilde{\rho}(A_{N_1}) \to \tilde{\rho}(\beta_N) = \tilde{\rho}(A_{N_1} \to \beta_N)\end{aligned}$$

We have that $\tilde{\rho}(A_{N_0}) \leqslant_{\mathcal{S}} \tilde{\rho}(A_{N_1} \to \beta_N)$, finishing the induction. Indeed, ρ justifies $T = T_M$ and $\mathcal{S}(T) \leqslant \mathcal{S}$, being what was required to show. $\square$

8.4 Discussion

The existence of principal recursive types and principal subtyping theories suggests a new approach to measuring the computational content of untyped lambda terms. The weaker-than relation $\preccurlyeq$ between type algebras applies in particular to the principal recursive types of lambda terms. Thus this relation can be extended to Λ as follows. For $M, N \in \Lambda$, write $\mathcal{A}_M$, $\mathcal{A}_N$ for their corresponding principal type algebras. Put

$$M \preccurlyeq N \iff \mathcal{A}_M \preccurlyeq \mathcal{A}_N \tag{8.4}$$

This definition introduces a new non-trivial partial order relation on the set of pure λ-terms. In contrast to the more classical notions like the Böhm tree order corresponding to the partial order in the **CPO** semantics, or the relative solvability order of Statman [1986], the relation (8.4) reflects the relative difficulty in assigning a type to the lambda terms M and N. On the semantic side, this addresses the following question: if we know that N can be realized as a morphism between some collection of structures, does it follow that M can also be realized as such?

For example, the principal type algebra $\mathcal{A}_{\omega\mathtt{I}}$ of the lambda term $\omega\mathtt{I} = (\lambda x.xx)(\lambda y.y)$ has an element $a \in A$ equal to $a \to a$ (Barendregt et al. [2011]). This makes $\mathcal{A}_{\omega\mathtt{I}}$ a weakly terminal object in the category of type algebras, since every other algebra can be mapped to it by the constant morphism that sends everything to a. Hence for every other algebra $\mathcal{A}$ we have $\mathcal{A} \preccurlyeq \mathcal{A}_{\omega\mathtt{I}}$, so that $\mathcal{A}_{\omega\mathtt{I}}$ is the "strongest" type algebra of all.

If we now work in the categorical semantics of simple types, then in place of the element $a = a \to a$ of $\mathcal{A}_{\omega\mathtt{I}}$ we will find an object A isomorphic to the exponential A^A. The construction of such an object is the crux of Scott's domain semantics for lambda calculus, since a solution to the recursive domain equation $A = A^A$ allows one to interpret all lambda terms both as maps over A and as elements of A. It is in this sense that being able to interpret $\omega\mathtt{I}$ entails the ability to interpret all other lambda terms.

In contrast, the term Ω requires only having a type B equal to $B \to C$, in which case it can be given type C. This recursive type does not suffice to type all lambda terms: it is not universal like the principal type of $\omega\mathtt{I}$. Hence Ω is strictly weaker than $\omega\mathtt{I}$: it can be realized as a morphism on the type $A = A^A$, but $\omega\mathtt{I}$ cannot be realized using only an object B isomorphic to $B \Rightarrow C$. This is the semantic meaning of $\Omega \prec \omega\mathtt{I}$.

Such a relationship might seem counterintuitive: $\omega\mathtt{I}$ is a strongly normalizing term with the unique reduction to its normal form $\mathtt{I}$ consisting of only two steps, while Ω is not even solvable. Thus the weaker-than order measures not so much the computational behavior of terms, but how demanding is the task of assigning functionality to them. The term $\omega\mathtt{I}$ applies identity to itself while requiring that its type be the same as its domain, necessarily equal to the codomain as well. But Ω only requires a structure isomorphic to the set of morphisms from it to some other space.

Furthermore, the subtyping order refines the above ordering, in the sense we will now make precise.

Let **TA** and **TS** be the categories of type algebras and type structures, respectively. In both cases, the weaker-than relation $\preccurlyeq$ is captured by morphisms of the category. There's a functor $E : \mathbf{TA} \to \mathbf{TS}$ which assigns to every type algebra the equality between its elements as a partial order. The functoriality immediately gives $\mathcal{A} \preccurlyeq \mathcal{B} \Longrightarrow E\mathcal{A} \preccurlyeq E\mathcal{B}$.

The functor E has a right adjoint U — the forgetful functor that, given any type structure, returns its underlying type algebra.

E also has a a left adjoint S, which, given $(S, \to, \preccurlyeq)$ returns the quotient of $(S, \to)$ modulo $\approx$, where $\approx$ is the symmetric closure of $\preccurlyeq$.

Hence we have the following picture:

$$\begin{tikzcd}
(A/{\approx}, \to_{\approx}) \arrow[d, "f"'] & (A, \to, \preccurlyeq) \arrow[l, "S"'] \arrow[d, "\tilde{f}"'] \\
(B, \rightsquigarrow) \arrow[r, "E"] \arrow[d, "g"'] & (B, \rightsquigarrow, =) \arrow[d, "\tilde{g}"'] \\
(C, \rightarrowtail) & (C, \rightarrowtail, \sqsubseteq) \arrow[l, "U"']
\end{tikzcd}$$

where the maps f and g represent generic type algebra morphisms, and $\tilde{f}$ and $\tilde{g}$ their transposes.

The above facts demonstrate precisely that subtyping refines the weaker-than relation on Λ induced by type algebras. First of all, notice that E is an embedding of **TA** into **TS** (in fact $U \circ E$ is the identity on **TA**) — thus the entire "$\preccurlyeq$-preorder" of type algebras is fully represented by type structures.

More importantly, the fact that E has a left adjoint means that the principal subtyping theorem is a strict generalization of the principal type theorem for recursive types. Specifically, let $\mathcal{S}_M$ be the principal type structure for

M. By definition, there's a morphism from $\mathcal{S}_M$ to any $E(\mathcal{A})$ with M typable in the algebra $\mathcal{A}$. Since these morphisms are in one-to-one correspondence with maps from $S(\mathcal{S}_M)$ to $\mathcal{A}$, we see that $S(\mathcal{S}_M)$ is weaker than any other type algebra that can type M. That is, $S(\mathcal{S}_M)$ is equivalent to the principal type algebra $\mathcal{A}_M$. Thus the principal type algebra of M can be recovered from the principal type structure of M.

The converse does not hold. Taking again the example $\omega\mathtt{I}$, we find that its principal type structure, even augmented with invertibility axioms, only yields the inequality $(A \to A) \leqslant A$ (the symmetric reflection of which does yield the reflexive type). This does not suffice to type every lambda term. (For example, the closure of T_{Ω_3} under invertibility axioms contains a type A with inequalities $(A \to A) \leqslant A$ and $A \leqslant (A \to A)$, giving it the power to type all lambda terms.)

This fact may seem puzzling, because in Scott's costruction, it actually suffices to have the function space $A \to A$ be a retract of A. Thus we might expect that such inequality would yield the strongest "universal" subtyping theory. So why does the categorical correspondence break down?

The answer is that the embedding–projection (EP) pairs that serve as the retraction "preorder" in Scott's construction do not reflect the intended semantics of subtyping: with respect to the EP pairs, the function space constructor is covariant in both arguments, while our type system assumes the arrow to be contravariant in the first argument. Hence EP pairs do not represent subtyping as we understand it. On the other hand, this "escape from contraviance" makes the function space operator monotone, allowing its iteration to inductively define the D_∞ model in the limit.

We conclude that the theory of subtyping yields a new partial-order structure on the lambda terms, which stays non-trivial even on the set of unsolvables. This opens a new avenue for studying relations between these terms, a topic about which very little is known. It is not immediately obvious whether the order described above is decidable, because type theories may be equivalent without being the same (for example, $\{\alpha \leqslant \beta \to \beta\}$ is equivalent to $\{\alpha \leqslant \alpha\}$). We conjecture that the $\leqslant$-order on Λ is decidable.

Bibliography

Barendregt, Henk [1984]. *The Lambda Calculus: Its Syntax and Semantics*, Studies in Logic and the Foundation of Mathematics 103, 2nd edition, North-Holland, Amsterdam.

Barendregt, Henk [1993]. Constructive proofs of the range property in Lambda Calculus, *Theoretical Computer Science* **121**(1-2), pp. 59 – 69. Corrado Böhm festschrift.

Barendregt, Henk [1994]. Discriminating Coded Lambda Terms.

Barendregt, Henk [1999]. Enumerators of Lambda Terms Are Reducing Constructively.

Barendregt, Henk [2008]. Towards the range property for the lambda theory H, *Theoretical Computer Science* **398**(1-3), pp. 12 – 15. Calculi, Types and Applications: Essays in honour of M. Coppo, M. Dezani-Ciancaglini and S. Ronchi Della Rocca.

Barendregt, Henk, Jan Bergstra, Jan Willem Klop and Henri Volken [1976]. Degrees, reductions and representability in the Lambda Calculus, *Preprint no. 22*, University of Utrecht, Department of Mathematics.

Barendregt, Henk, Wil Dekkers and Richard Statman [2011]. *Lambda Calculus with Types*, North-Holland?

Baumgartner, Peter, Ulrich Furbach and Ilkka Niemelä [1996]. Hyper tableaux, *in:* José Alferes, Luís Pereira and Ewa Orlowska (eds.), *Logics in Artificial Intelligence*, Lecture Notes in Computer Science 1126, Springer Berlin / Heidelberg, pp. 1–17.

Bezem, Marc and Thierry Coquand [2005]. Automating Coherent Logic, *in:* Geoff Sutcliffe and Andrei Voronkov (eds.), *Logic for Programming, Artificial Intelligence, and Reasoning*, Lecture Notes in Computer Science 3835, Springer Berlin / Heidelberg, Berlin, Heidelberg, chapter 18, pp. 246–260.

Bezem, Marc and Dimitri Hendriks [2008]. On the Mechanization of the Proof of Hessenberg's Theorem in Coherent Logic, *Journal of Automated Reasoning* **40**(1), pp. 61–85.

Bezem, Marc, Dimitri Hendriks and Hans de Nivelle [2000]. Automated Proof Construction in Type Theory Using Resolution, *in:* David McAllester (ed.), *Automated Deduction - CADE-17*, Lecture Notes in Computer Science 1831, Springer Berlin / Heidelberg, Berlin, Heidelberg, chapter 10, pp. 148–163.

Böhm, Corrado [1968]. Alcune proprieta delle forme $\beta\eta$-normali nel λK-calculus, *Pubblicazioni 696*, Instituto per le Applicazioni del Calcolo, Roma.

Buss, Samuel R. [1998]. An introduction to proof theory, *in:* Samuel R. Buss (ed.), *Handbook of Proof Theory, Studies in Logic and the Foundations of Mathematics*, Elsevier, Amsterdam, p. 811.

Coppo, M., M. Dezani-Ciancaglini and S. Ronchi della Rocca [1978]. (Semi)-separability of finite sets of terms in Scott's D-infinity models of the lambda calculus, *in:* Giorgio Ausiello and Corrado Böhm (eds.), *Automata, Languages and Programming*, Lecture Notes in Computer Science 62, Springer Berlin / Heidelberg, pp. 142–164.

Coquand, Thierry [2003]. A completeness proof for geometrical logic, pp. 79–90.

David, René and Karim Nour [2010]. Personal communication.

De Nivelle, Hans and Jia Meng [2006]. Geometric resolution: A proof procedure based on finite model search, *In Proc. International Joint Conference on Automated Reasoning (IJCAR*, pp. 303–317.

Endrullis, Jörg, Dimitri Hendriks and Jan Willem Klop [2010]. Modular construction of fixed point combinators and clocked Böhm trees, *Logic* *in Computer Science (LICS)* **2010.8**, pp. 111 – 119. IEEE Federated Logic Conference.

Fisher, John [2008]. Personal communication.

Fisher, John [2009]. Concurrent coherent logic, preprint.

Fisher, John and Marc Bezem [2007]. Skolem Machines and Geometric Logic, *in:* Cliff Jones, Zhiming Liu and Jim Woodcock (eds.), *Theoretical Aspects of Computing – ICTAC 2007*, Lecture Notes in Computer Science 4711, Springer Berlin / Heidelberg, Berlin, Heidelberg, chapter 14, pp. 201–215.

Gentzen, Gerhard [1935]. Untersuchungen über das logische Schließen. II, *Mathematische Zeitschrift* **39**(1), pp. 405–431.

Girard, Jean Y., Paul Taylor and Yves Lafont [1989]. *Proofs and types*, Cambridge University Press, New York, NY, USA.

Hofstadter, Douglas R. [1999]. *Gödel, Escher, Bach: An Eternal Golden Braid*, 20 anv edition, Basic Books.

Intrigila, Benedetto and Richard Statman [2007]. On Henk Barendregt's favorite open problem, *Reflections on Type Theory, Lambda Calculus, and the Mind.* Henk Barendregt festschrift.

Janicic, Predrag and Stevan Kordic [1995]. Euclid - the geometry theorem prover, *FILOMAT* **9:3**, pp. 723–732.

Kleene, S. C. [1936]. Lambda-definability and recursiveness, *Duke Mathematical Journal* **2**, pp. 340–353.

McCarthy, John [1962]. *LISP 1.5 Programmer's Manual*, The MIT Press.

Mogensen, Torben [1994]. Efficient Self-Interpretation in Lambda Calculus, *Journal of Functional Programming* **2**, pp. 345–364.

Mulvey, J [2003]. On the geometry of choice, *Topological and algebraic structures in fuzzy sets*, Trends Log. Stud. Log. Lib. 20, Kluwer Acad. Publ., Dordrecht, pp. 309–336.

de Nivelle, Hans [2003]. Translation of Resolution Proofs into Short First-Order Proofs without Choice Axioms, *in:* Franz Baader (ed.), *Automated Deduction – CADE-19*, Lecture Notes in Computer Science 2741, Springer Berlin / Heidelberg, Berlin, Heidelberg, chapter 33, pp. 365–379.

Otten, Jens [2009]. Personal communication.

Pfenning, Frank and Conal Elliott [1988]. Higher-order abstract syntax, *PLDI*, pp. 199–208.

Smullyan, R [1994]. Diagonalization and self-reference.

Smullyan, Raymond M. [1995]. *First-Order Logic*, Dover Publications.

Statman, Richard [1993]. Does the range property hold for the λ-theory H?,TLCA list of open problems, http://tlca.di.unito.it/opltlca/.

Statman, Rick [1986]. Every countable poset is embeddable in the poset of unsolvable terms, *Theoretical Computer Science* **48**, pp. 95 – 100.

de Vrijer, Roel [2007]. Barendregt's lemma, *Reflections on Type Theory, Lambda Calculus, and the Mind.* Henk Barendregt festschrift.