Constructive Completeness Proofs and Delimited Control

Danko Ilik

Thèse

présentée pour obtenir le grade de

Docteur de l'Ecole Polytechnique

Spécialité :

Informatique

par

Danko ILIĆ

Titre de la thèse :

Preuves constructives de complétude et contrôle délimité

Soutenue le 22 octobre 2010 devant le jury composé de :

M.	Hugo HERBELIN	Directeur de thèse
M.	Ulrich BERGER	Rapporteur
M.	Thierry COQUAND	Rapporteur
M.	Olivier DANVY	Rapporteur
M.	Gilles DOWEK	Examinateur
M.	Paul-André MELLIÈS	Examinateur
M.	Alexandre MIQUEL	Examinateur
M.	Wim VELDMAN	Examinateur

Constructive Completeness Proofs and Delimited Control

Danko ILIĆ

ABSTRACT. Motivated by facilitating reasoning with logical meta-theory inside the Coq proof assistant, we investigate the constructive versions of some completeness theorems.

We start by analysing the proofs of Krivine and Berardi-Valentini, that classical logic is constructively complete with respect to (relaxed) Boolean models, and the algorithm behind the proof.

In an effort to make a more canonical proof of completeness for classical logic, inspired by the normalisation-by-evaluation (NBE) methodology of Berger and Schwichtenberg, we design a completeness proof for classical logic by introducing a notion of model in the style of Kripke models.

We then turn our attention to NBE for full intuitionistic predicate logic, that is, to its completeness with respect to Kripke models. Inspired by the computer program of Danvy for normalising terms of λ-calculus with sums, which makes use of delimited control operators, we develop a notion of model, again similar to Kripke models, which is sound and complete for full intuitionistic predicate logic, and is coincidentally very similar to the notion of Kripke-style model introduced for classical logic.

Finally, based on observations of Herbelin, we show that one can have an intuitionistic logic extended with delimited control operators which is equiconsistent with intuitionistic logic, preserves the disjunction and existence properties, and is able to derive the Double Negation Shift schema and Markov's Principle.

RÉSUMÉ. Motivés par la facilitation du raisonnement sur des méta-théories logiques à l'intérieur de l'assistant de preuve Coq, nous étudions les versions constructives de certains théorèmes de complétude.

Nous commençons par l'analyse des preuves de Krivine et Berardi-Valentini qui énoncent que la logique classique est constructivement complète au regard des modèles booléens relaxés, ainsi que l'analyse de l'algorithme de cette preuve.

En essayant d'élaborer une preuve de complétude plus canonique pour la logique classique, inspirés par la méthode de la normalisation-par-évaluation (NPE) de Berger et Schwichtenberg, nous concevons une preuve de complétude pour la logique classique en introduisant une notion de modèle dans le style des modèles de Kripke, dont le contenu calculatoire est l'élimination des coupures, ou la normalisation.

Nous nous tournons ensuite vers la NPE pour une logique de prédicats intuitionniste (en considérant tous les connecteurs logiques), c'est-à-dire, vers sa complétude par rapport aux modèles de Kripke. Inspirés par le programme informatique de Danvy pour la normalisation des termes du λ-calcul avec sommes, lequel utilise des opérateurs de contrôle délimité, nous développons une notion d'un modèle, encore une fois semblable aux modèles de Kripke, qui est correct et complet pour la logique de prédicats intuitionniste, et qui est, par coïncidence, très similaire à la notion de modèle de Kripke introduit pour la logique classique.

Finalement, en se fondant sur des observations de Herbelin, nous montrons que l'on peut avoir une logique intuitionniste étendue avec des opérateurs de contrôle délimité qui est equiconsistante avec la logique intuitionniste, qui préserve les propriétés de disjonction et d'existence, et qui est capable de dériver le schéma « Double Negation Shift » et le principe de Markov.

Contents

Acknowledgements

This thesis is not individual work, I just happen to write it. Any credit it possibly gets is to be shared between all those who have been around in the past, teachers, family, and friends.

My director Hugo Herbelin was a guide as one can only wish to have. I am deeply grateful to him for his trust in me and for his evergreen enthusiasm. I will never forget the many afternoons of delirious discussions about Logic and control operators.

Behind the scenes was the support of my family. My *belle-mère* Jeanne Delcroix Angelovski spent many months in Paris taking care of our son Gricha. My mother in Skopje spent all of her free time helping out. The love of Despina and Gricha was a constant catalyst and a well of inspiration. Among friends which have been encouraging me and keeping contact are Dragan Bocevski, Georgi Stojanov, Andrea Maura Castilla, Irfan Junedi Khaja Mohammad, Takako Nemoto, and Monica and François.

Paris was the best place I could have been in to do a thesis. I thank *all* of my colleagues and friends from the laboratories LIX and PPS. It was very inspiring to be among them. I would also like to thank everyone from the Logic Masterclass in Utrecht and Nijmegen; it is in a sense where my thesis started and it was certainly the single one year I have learnt the most in my entire life. Andrej Bauer and Giovanni Sambin have been my hosts for the research visits to Ljubljana and Padova, I thank them for their hospitality.

Special thanks go to the rapporteurs and the members of jury. They were prepared to read my thesis, give valuable comments, come to Paris, and responded on short notice. I deeply thank them.

Finally, I would like to thank the founding agencies without which I would not have been writing these lines. First, The Swedish Institute for granting me a MSc studies scholarship in Sweden; I do not believe that I would have had the opportunity to do scientific research without them. Second, the Netherlands' Mathematical Research Institute for financing my Masterclass studies. Finally, the École Polytechnique provided me with a generous grant for PhD studies which gave me the best possible working conditions.

It has been three years. I am happy to be finished and looking forward to continuing.

Introduction

The Curry-Howard correspondence has provided a major impulse for cross-fertilisation of the fields of Theoretical Computer Science (Programming Languages) and Mathematical Logic (Proof Theory). It consists of the observation that to each proof in a system of intuitionistic logic, there corresponds a program in a statically typed effect-free functional programming language, and vice versa.

This allows, on one hand, to get a direct computational interpretation of Intuitionistic Mathematics[1], and, on the other hand, to use Mathematics as a language for specifying the desired behaviour of programs. A well-typed program is a correct program, since it represents itself a proof of its own correctness with respect to the specification.

In the past three decades, special purpose software called "proof assistants" have emerged allowing formal computer-aided development of intuitionistic proofs and programs. They, on one hand, present environments in which a mathematician can automatically check whether his proofs can be traced back to the axioms, and, on the other hand, they present programming languages in which a computer scientist can write programs and formally verify whether they are correct, that is, conform to a specification.

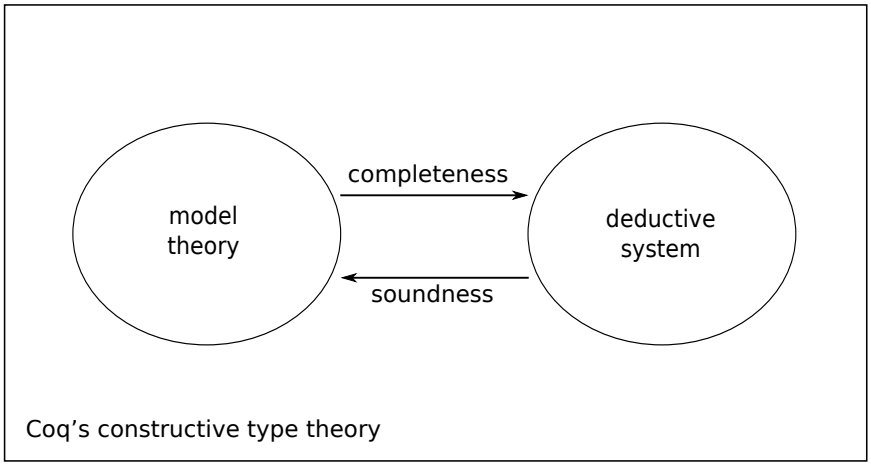
In this thesis, we *apply* the Curry-Howard correspondence both ways. We first study the computational content of a completeness theorem from Mathematical Logic, and then, starting from certain normalisation algorithms, we develop new completeness theorems. We also propose an *extension* of the Curry-Howard correspondence to account for not widely accepted intuitionistic principles, and to account for certain "delimited" control operators that allow to go beyond effect-free programming languages.

The work of this thesis was started under the title "Applied Formal Metatheory in Coq". Coq is a proof assistant, originally based on Coquand's Calculus of Constructions. The purpose of our work was to make reasoning about logical systems inside Coq easier for its user. The idea, due to Herbelin, was that if we formalise a *constructive* version of some completeness theorem, for example Gödel's completeness theorem for classical predicate calculus, we would have a certified procedure allowing us to switch back-and-forth between proof theoretic and model theoretic reasoning about classical predicate logic. This would have the following practical advantages for users of the Coq proof assistant:

- when reasoning "model theoretically", the usual tactics and automation of Coq could be used, instead of having to manipulate directly the data encoding the proof theoretic side;

[1] The Curry-Howard correspondence has been also extended to Classical Mathematics.

- the semantic characterisation of the model theory would represent just a fragment of the quite strong type theory of Coq, a fragment which would exactly capture, for example, first-order classical validity;
- finally, users would be spared from the problems connected to α-conversion when reasoning with binders (quantifiers) formally: the constructive proof of completeness would take care of that automatically.



Soon after starting the project, we saw that there are many interesting theoretical questions still asking for attention, from a constructive (intuitionistic) point of view, therefore, although the completeness proofs of this thesis were formalised in Coq, we did not pursue the goal of their practical "real life" applicability.

The work is divided into four chapters, each of which has a proper introduction, here we just list their contributions.

Chapter 1 investigates the constructive versions of Gödel's proof of completeness for classical predicate logic, following the works of Krivine, and of Berardi and Valentini.

- We fill-in the details of how to do a constructive Henkin-style proof, avoiding the explicit building of an infinite sequence of language extensions, supplementing the work of Berardi and Valentini, and of Henkin;
- We formalise a larger part of the argument in Coq, and, based on that, give a characterisation of its computational behaviour;
- In the historic review, we give an English introduction to Krivine's article written in French.

Chapter 2 introduces a notion of model similar to Kripke models for which classical predicate logic is sound and complete. A part of the results of this chapter has been published as article [**109**].

- We give a notion of model which relies on a dual to the "forcing" relation, and therefore more directly captures classical validity than what would be possible with usual Kripke models and a double-negation translation to intuitionistic logic;
- The notion of model is at the same time a kind of continuations monad, a structure well understood in computer science; in particular, it has a

direct computational interpretation, unlike semantic cut-elimination proofs that rely on set comprehension;
- We give a computational characterisation of completeness via normalisation-by-evaluation-like algorithms, based on a reify-reflect pair;
- We fully formalise the arguments in Coq;
- We show that a dual notion of model is possible, the original one giving rise to a call-by-name evaluation strategy, the dual one giving rise to a call-by-value strategy.

Chapter 3 develops a completeness proof for intuitionistic predicate logic, with $\vee$ and $\exists$, based on a new notion of model similar to the one of Chapter 2.

- Using the proposed models, we are able to prove completeness without the Fan theorem, unlike the case for Kripke and Beth models;
- Also, we are able to avoid the use of delimited control operators that Danvy uses in an algorithm for partial evaluation of λ-calculus with sum, that is at the origin of our work;
- We fully formalise the arguments in Coq;
- We make a connection with the models of Chapter 2, that relies solely on the continuations (not) being able to change their answer types.

Chapter 4 proposes an extension of intuitionistic predicate logic to a system that has the disjunction and existence properties, but can also derive predicate-logic versions of the Double Negation Shift schema and Markov's Principle.

- We provide a typing system for delimited control operators that is close to already proposed systems, but crucially relies on certain types representing Σ^0_1-formulae;
- We connect delimited control operators to bar recursive realisability;
- We partially extend the work of Herbelin on an intuitionistic logic that can prove Markov's Principle;
- We obtain the extension of Glivenko's theorem for our system.

All proofs in this thesis are constructive, except for the proof sketches from the historical overview of Chapter 1.

CHAPTER 1

Constructive completeness for Boolean models

In Foundations of Mathematics, the completeness problem appears as a natural question as soon as we start to consider formal systems for giving a rigorous treatment of mathematical arguments. The problem, for classical predicate logic apparently first posed in print by Hilbert and Ackermann in [**104**], is the problem of adequacy of the formal system under consideration, that is, the question: Can every true statement be derived in finitely many steps by means of the axioms and rules of inference of the formal system?

While from the work of Bernays [**34**] it was known that the answer is positive for the propositional fragment, due to the existence of the decision procedure based on truth tables, for predicate logic the existence of such a procedure was unknown, and was actually a major research problem known as the Entscheidungsproblem. (German for 'decision problem')

Not long after Hilbert and Ackermann, the completeness problem was reposed and answered positively for predicate logic by Gödel in his doctoral dissertation [**82**], published as article in [**83**]. His solution circumvented the Entscheidungsproblem, which remained unsolved until the results of Church [**46**] and Turing [**167**].

In this chapter we will take a fresh look at the completeness theorem for classical predicate logic. We will start with a historic overview of the versions of the proof from the one of Gödel, through the one of Henkin [**94**], to the one of Krivine [**125**], which represents the first constructive proof. Then, we will go on, following Berardi and Valentini [**29**], to develop a detailed constructive argument (Sections 1.2 and 1.3), which was a subject of partial formalisation in the Coq proof assistant [**62**] (Section 1.5), and to discuss the computational content (Section 1.4) and the remaining related works (Section 1.6).

1.1. Historical overview

1.1.1. Basic definitions. The completeness theorem connects an intuitive notion of truth of a mathematical statement to that of its provability in a formal system. At the time of Gödel's early works, and up to the 1950s, around was a notion of truth that was an intuitive extension, to deal with quantifiers, of the truth-table validity of propositional logic. Today, the intuitive notion that Gödel and others worked with can be recognised as an instance of the standard Tarski truth definition of Model Theory.[1]

[1]However, Tarski's truth definition only took its definite model-theoretic form in [**159**], while the first publish theory of truth of Tarski [**158**] had a more generic approach. In [**158**] Tarski proposed a syntactic notion of truth, based on two separate formal languages: an object language L (the language whose truth is to be defined), and a meta-language M (the language used to define truth of sentences of L). In the spirit of his time, Tarski assumed [**105**] that L and M would be based on some kind of higher-order logic, but he was aware of the fact that M has to be stronger than L

1.1.1. Definition. A *signature K* is a collection of individual constants $c_0, c_1, \ldots$ (finitely or infinitely many), and predicate and function symbols with finite arities, $P_0, P_1, \ldots P_n$ and $f_0, f_1, \ldots f_m$.

1.1.2. Definition. The *language of signature K* consists of *formulae*, which are built up inductively, using individual variables, standard logical constants, and the symbols of K:

- $\bot$ is a formula
- $P_i(t_1, \ldots, t_n)$ is a formula, if $t_1, \ldots, t_n$ are *terms*;
  - an individual variable x is a term;
  - an individual constant c_j of K is a term;
  - if $t_1, \ldots, t_m$ are terms, so is $f_l(t_1, \ldots, t_m)$.
- if A, B are formulae, then $A \wedge B$, $A \vee B$, and $A \to B$ are formulae;
- if A is a formula, possibly containing x as a free individual variable, then $\exists x A$ and $\forall x A$ are formulae.

The formulae of the first two forms are called *prime* or *atomic*. The formula $\neg A$ is an abbreviation for $A \to \bot$. We call *sentence* a formula in which all individual variables are bound by a quantifier.

1.1.3. Remark. Often, a special predicate symbol "=" is taken as a default member of every signature. For simplicity, similarly to Gödel, Henkin, and Krivine, we do not treat "=" as special.

1.1.4. Definition. A *structure of signature K*,

$$\mathscr{M} = (M, P_0^{\mathscr{M}}, \ldots, P_n^{\mathscr{M}}, f_0^{\mathscr{M}}, \ldots, f_m^{\mathscr{M}}, c_0^{\mathscr{M}}, c_1^{\mathscr{M}}, \ldots),$$

consists of:

- a domain of individuals M, typically a set;
- for each predicate symbol P of arity k of K, a k-ary relation $P^{\mathscr{M}}$ on the domain M, that is, a subset of M^k;
- for each function symbol f of arity k of K, a k-ary function $f^{\mathscr{M}}$ from M^k into M;
- for each constant symbol c of K, an element $c^{\mathscr{M}}$ of M to denote c.

1.1.5. Definition. We say that a structure $\mathscr{M}$ is a *model* of a formula A, that $\mathscr{M}$ *satisfies* A, or that $\mathscr{M}$ *realises* A, and write $\mathscr{M} \vDash A$, if the following primitive-recursive meta-language interpretation of A holds:

- for $A \equiv \bot$, $\mathscr{M} \nvDash \bot$;
- for $A \equiv P(a_1, \ldots, a_k)$, $\mathscr{M} \vDash A$ if $(a_1, \ldots, a_k) \in P^{\mathscr{M}}$;
- for $A \equiv A_1 \wedge A_2$, $\mathscr{M} \vDash A$ if $\mathscr{M} \vDash A_1$ and $\mathscr{M} \vDash A_2$;
- for $A \equiv A_1 \vee A_2$, $\mathscr{M} \vDash A$ if $\mathscr{M} \vDash A_1$ or $\mathscr{M} \vDash A_2$;
- for $A \equiv A_1 \to A_2$, $\mathscr{M} \vDash A$ if $\mathscr{M} \vDash A_1$ implies $\mathscr{M} \vDash A_2$;
- for $A \equiv \forall x B$, $\mathscr{M} \vDash A$ if $\mathscr{M} \vDash B[a/x]$ for all $a \in M$;
- for $A \equiv \exists x B$, $\mathscr{M} \vDash A$ if $\mathscr{M} \vDash B[a/x]$ for some $a \in M$.

We say that A is *satisfiable*, or *realisable*, or *has a model*, if there exists a structure $\mathscr{M}$ which is a model of A.

---

in order for the Liar paradox [**85**] to be avoided. Nowadays, in Model Theory, a full set theory with an axiom of choice is standardly assumed for M.

1.1.6. Definition. Let K be a signature. We say that a formula A, written in the language of signature K, is *true*, or *valid*, if any structure $\mathscr{M}$ of signature K is a model of A.

We now have set up the basic framework for looking at the major instances of the theorem which appeared through history. In this chapter $\vdash A$ will stand for derivability of A inside a system for classical predicate logic. In Subsections 1.1.2, 1.1.3, and 1.1.4, we assume also a classical metalanguage.

1.1.2. Gödel's proof. We give a sketch of Gödel's original proof, based on [**84, 83, 82**]. The key role is played by the following lemma, today known as Model Existence Lemma.

1.1.7. Lemma. *For every formula A of a language with signature K, either there exists a structure $\mathscr{M}$ of signature K such that $\mathscr{M} \vDash A$, or $\vdash \neg A$.*

1.1.8. Theorem (Completeness)**.** *If A is true, then A is derivable.*

PROOF. We apply Lemma 1.1.7 on the formula $\neg A$. Then, if there is a model $\mathscr{M}$ of $\neg A$, we obtain contradiction, since $\mathscr{M}$ is by hypothesis also a model of A. Otherwise, the formula $\neg\neg A$ is derivable, hence A is itself derivable . □

PROOF SKETCH OF LEMMA 1.1.7. We say that a formula A is *refutable* if $\neg A$ is derivable. Gödel shows that every formula is either satisfiable or refutable, by showing that every formula in prenex normal form is either satisfiable or refutable, since we know that the equivalence between a formula and its prenex normal form is derivable. Actually, it is enough to consider prenex normal forms where the left-most quantifier is universal, because each prefixing existential quantifier can be immediately eliminated by replacing the variable it binds with a constant. Let the *degree* of such a prenex normal formula be the number of blocks of universal quantifiers separated by existential ones. The proof is by induction on the degree:

(1) *If every formula of degree k is either satisfiable or refutable, then so is every formula of degree $k+1$.* This is proved by Skolemisation. A formula of degree $k+1$ is transformed into one of degree k which is equisatisfiable with the first one. New function symbols are introduced in the process.

(2) *Every formula of degree* 1 *is either satisfiable or refutable.* A formula P of degree 1 is of the form $\forall\vec{r}\exists\vec{n}A(\vec{r};\vec{n})$, where $\vec{r}$ denotes a q-tuple of variables, and $\vec{n}$ denotes an s-tuple of variables. Let $(\vec{r}_n)$ be an infinite sequence of q-tuples of variables $x_0, x_1, x_2 \ldots$ generated in (some) lexicographical order:

$$\begin{aligned}\vec{r}_1 &= (x_0, x_0, \ldots, x_0)\\ \vec{r}_2 &= (x_1, x_0, \ldots, x_0)\\ \vec{r}_3 &= (x_0, x_1, \ldots, x_0)\\ &\vdots\end{aligned}$$

We define the sequence of formulae A_n by

$$\begin{aligned} A_1 &= A(\vec{r}_1; x_1, x_2, \dots x_s) \\ A_2 &= A(\vec{r}_2; x_{1+s}, x_{2+s}, \dots x_{2s}) \wedge A_1 \\ &\vdots \\ A_n &= A(\vec{r}_n; x_{(n-1)s+1}, x_{(n-1)s+2}, \dots x_{ns}) \wedge A_{n-1} \\ &\vdots \end{aligned}$$

where, for each A_n, the variables put in the places bound by the existential quantifier do not appear in the formulae A_m for $m < n$. We denote by P_n the formula $\exists x_0 \cdots \exists x_{ns} A_n$. It is easy to show that $P \Rightarrow P_n$ is derivable. Now, because each of A_n is a formula of propositional logic,

(a) either some A_n is refutable, and hence P is refutable because P_n is refutable;
(b) or, no A_n is refutable, that is, all A_n are satisfiable, and hence we get an infinite sequence of models

$$\mathscr{M}_1 \subseteq \mathscr{M}_2 \subseteq \cdots \subseteq \mathscr{M}_n \subseteq \cdots .$$

Then $\mathscr{M} := \cup_{i \in \mathbb{N}} \mathscr{M}_i$ is a model of the formula P.

□

The introduction notes to [**82, 83, 84**] see in the last step an application of Kőnig's lemma, although Gödel himself justifies the step by "familiar arguments".

1.1.3. Henkin's proof. It was Henkin who apparently first remarked the slight imprecision in the Skolemisation step of Gödel's proof. Namely, in order to eliminate existential quantifiers that follow universal ones, Skolemisation introduces new function symbols which are not interpreted in the models $\mathscr{M}_1 \subseteq \mathscr{M}_2 \subseteq \cdots$, because they come from an extended language.

It is Henkin's proof which is standard in today's textbooks on logic. It was carried out in his PhD thesis and published in article form as [**94**]. Henkin's thesis goes beyond the article [**94**], because it also discusses completeness in the context of higher-order logic and proves, using completeness, Stone's representation theorem for Boolean algebras.

The key role is played by the following Model Existence Lemma.

1.1.9. Lemma. *Let S_0 be a signature. If Λ is a set of sentences of signature S_0, which is consistent ($\Lambda \nvdash \bot$), then Λ has a model.*

PROOF SKETCH. Let S_{i+1} be a signature that extends S_i with countably many new constant symbols $u_0^{i+1}, u_1^{i+1}, \dots$. A set of sentences Γ of signature S will be called *maximal consistent* if, for any sentence A of signature S, $A \notin \Gamma \to \Gamma, A \vdash \bot \,\&\, \Gamma \nvdash \bot$, that is, if $(\Gamma, A \vdash \bot \to \Gamma \vdash \bot)$ implies $A \in \Gamma$.

We will now construct Γ_0, a maximal consistent set of sentences of S_0, that contains the given set Λ. Let $\Gamma_{0,0} := \Lambda$. We fix an enumeration of formulae of signature S_0. Let $\Gamma_{0,1} := \Gamma_{0,0} \cup \{B_1\}$, where B_1 is the first formula from the enumeration such that $\Gamma_{0,0} \cup \{B_1\}$ is consistent. In general, let $\Gamma_{0,i+1} := \Gamma_{0,i} \cup \{B_{i+1}\}$, where B_{i+1} is the $(i+1)$-th formula from the enumeration such that $\Gamma_{0,i} \cup \{B_{i+1}\}$ is consistent. We set $\Gamma_0 := \cup_{i \in \mathbb{N}} \Gamma_{0,i}$, for which we have that:

- $\Lambda \subseteq \Gamma_0$;
- Γ_0 is consistent, because each one of $\Gamma_{0,i}$ is consistent by definition;
- Γ_0 is maximal consistent: given A such that $\Gamma_0, A \vdash \bot \rightarrow \Gamma_0 \vdash \bot$, that is, given $\Gamma_0, A \nvdash \bot$, that is, given Γ_0, A is consistent, we have that each $\Gamma_{0,i}, A$ is consistent. Since A also appears in the enumeration of formulae, for some j, $A \in \Gamma_{0,j}$, hence $A \in \Gamma_0$.

We have thus built a maximally consistent set of sentences of S_0, Γ_0.

We will now proceed to build a maximally consistent set of sentences of signature S_1, Γ_1.

Fix an enumeration of the sentences of Γ_0. Select the first sentence of form $\exists x A$ from the enumeration, and let $A' := A\{u_1^1/x\}$. We have replaced the free variable x of A with the first new constant from S_1. The set Γ_0, A' is consistent: if $\Gamma_0, A' \vdash \bot$, then $\Gamma_0 \vdash \neg A'$, hence $\Gamma_0 \vdash \forall x \neg A$, and $\Gamma_0 \vdash \neg \exists x A$, which contradicts the fact that Γ_0 is consistent, since $A \in \Gamma_0$.

Hence we can add to Γ_0 all such A', keeping it consistent. We now construct Γ_1 in the same way we constructed Γ_0, but starting from the consistent set of sentences $\Gamma_0, A'_1, A'_2, \ldots$.

We can iterate this procedure constructing a maximally consistent set Γ_i of sentences of signature S_i, by starting from the consistent set $\Gamma_{i-1}, A_1^{i\prime}, A_2^{i\prime}, \ldots$.

We can now define the set of sentences $\Gamma_\omega := \cup_{i \in \mathbb{N}} \Gamma_i$, and easily see that it satisfies two properties:

(1) Γ_ω is a maximally consistent set of sentences of signature S_ω;
(2) if $(\exists x A) \in \Gamma_\omega$, then $A' \in \Gamma_\omega$.

Actually, as Henkin remarks, the entire construction was just in order to obtain these two properties.

We can now define the model promised by the statement of the Lemma. Let $\mathscr{I}$ be a structure of signature S_ω in which the domain of individuals consists of all individual constants (the old constants of S_0 and all the new ones). For an atomic formula A, we define the truth of A in $\mathscr{I}$, by the derivability of A from Γ_ω. The use of properties (1) and (2) is in showing that the extension to composite formulae A of the property,

$$\mathscr{I} \vDash A \text{ iff } \Gamma_\omega \vdash A,$$

holds, that is, that validity in $\mathscr{I}$ is well defined. The proof is by induction on the complexity of A. Property (1) is used to handle implication (and negation), and property (2) is used to handle the quantifiers. □

1.1.10. Theorem (Completeness)**.** *If A is a valid sentence of S_0, then $\vdash A$.*

PROOF. If A is valid, then $\neg A$ has no model, which, by contraposition of Lemma 1.1.9, means that $\neg A \vdash \bot$. Therefore $\vdash A$. □

In the version of proof that we will give in this chapter, we will avoid the explicit infinite sequence of language extensions. That this can be done, is also a subsequent realisation of Henkin [**95**, p.156].

1.1.4. Krivine's proof. Krivine was the first to give a constructive proof [**125**] of Gödel's completeness theorem.[2] He shows that the statement of completeness for first-order logic can be formalised as a true formula TC of classical second-order logic, with the axiom schema of comprehension and the axiom of full second-order induction. The formula TC uses five function symbols $\{0, s, \hookrightarrow, \sigma, @\}$, but no axioms are supposed for the last three of those.

A *concrete* proof of completeness of first-order logic can be obtained by building a concrete second-order model $\mathscr{M}_0$, which interprets the five function symbols in the intended way:

- the domain of individuals of $\mathscr{M}_0$ is the set of first-order sentences of a signature L which, besides the five function symbols, contains also countably many constant symbols;
- for a fixed enumeration of the sentences, the nullary function symbol 0 is interpreted as the first formula in the enumeration, and s is a unary function symbol that, given a formula, returns the next one according to the enumeration;
- the binary function symbol $\hookrightarrow$ is interpreted as implication between sentences;
- for a fixed bijection $G \mapsto t_G$ between sentences and closed terms of signature L, the function symbol σ is interpreted as substitution:

  $$\sigma(\forall x A', B) := A'\{t_B/x\},$$
  $$\sigma(A, B) := A, \qquad \text{if } A \text{ is not a universal formula.}$$

  and @ is interpreted as a generator of fresh terms:

  $$@(A, B) := t_C$$

  where C is a sentence such that the term t_C does not appear in A and B.
- the second order variables of arity n are interpreted, as usually for second-order models, by n-ary relations on the domain of $\mathscr{M}_0$.

Krivine's proof is constructive because the formula TC has a form such that its double-negation translation is equivalent to TC inside intuitionistic second-order logic.

Since the article [**125**] contains a very detailed formal argument, we will here content ourselves to just describing the formula TC and giving a sketch of the proof.

Let M and J be unary second-order variables. In the model $\mathscr{M}_0$, such an entity is a collection of formulae. Let $\forall x\,\mathrm{Ent}(x)$ denote the axiom of second-order induction, that is, let $\mathrm{Ent}(x)$ be the formula:

$$\forall X \big(\forall y \big(Xy \to X(sy)\big) \to X0 \to Xx\big).$$

[2] N.B. Observations that Gödel's proof is essentially constructive appear already, in a couple of places, in the papers [**119, 116, 120**] of Kreisel.

We define the predicate $\mathrm{Mod}(M)$, to be read as "M is a model", by the conjunction of the following formulae:

$$\begin{aligned}
&\forall x\, y\, \big(M(x \hookrightarrow y) \to Mx \to My\big)\\
&\forall x\, y\, \big(\mathrm{Ent}(x) \to (Mx \to My) \to M(x \hookrightarrow y)\big)\\
&\forall x\, y\, \big(Mx \to M(\sigma(x,y))\big)\\
&\forall x\, \big(\mathrm{Ent}(x) \to (\forall y M(\sigma(x,y))) \to Mx\big)
\end{aligned}$$

We also define a predicate $\mathrm{Ded}(J)$, to be read as "J is closed by deduction", by the conjunction of the following formulae, which, in the model $\mathscr{M}_0$, express that J is a collection of formulae closed under deduction from the rules of Hilbert's system for propositional calculus plus the axioms for introduction and elimination of the universal quantifier.

$$\begin{aligned}
&\forall x\, y\; J(y \hookrightarrow x \hookrightarrow y)\\
&\forall x\, y\, z\; J((x \hookrightarrow y) \hookrightarrow (x \hookrightarrow y \hookrightarrow z) \hookrightarrow x \hookrightarrow z)\\
&\forall x\, y\; J(((x \hookrightarrow y) \hookrightarrow x) \hookrightarrow x) && \text{Peirce's law}\\
&\forall x\, y\; J(x \hookrightarrow y) \to Jx \to Jy && \text{modus ponens}\\
&\forall x\, y\; J(x \hookrightarrow \sigma(x,y))\\
&\forall x\, y\; J\big(\big(\sigma\,(x, @\,(x,y)\big) \hookrightarrow y\big) \hookrightarrow y\big) \to J\big(\big(x \hookrightarrow y\big) \hookrightarrow y\big)
\end{aligned}$$

Now, starting from the simple version of completeness specified by the formula

$$(\mathrm{TC}_0) \qquad \forall x\,(\forall M\,(\mathrm{Mod}(M) \to Mx) \to \forall J\,(\mathrm{Ded}(J) \to Jx))\,,$$

we generalise to the full statement of completeness, where the formula x is valid and derivable modulo a collection of formulae P (that is, a collection of axioms),

$$(\mathrm{TC}_0(P)) \qquad \forall x\,(\forall M\,(\mathrm{Mod}(M) \to P \subseteq M \to Mx) \to \forall J\,(\mathrm{Ded}(J) \to P \subseteq J \to Jx))\,,$$

and we finally arrive at

$$(\mathrm{TC}) \qquad \forall x \forall J\,(\forall M\,(\mathrm{Mod}(M) \to J \subseteq M \to Mx) \to \mathrm{Ded}(J) \to Jx)$$

which is equivalent to $\forall P.\mathrm{TC}_0(P)$.

1.1.11. Theorem. *The formula TC is a valid formula of both intuitionistic and of classical second-order logic, with as axioms the comprehension schema and full second-order induction, in the language* $\{0, s, \hookrightarrow, \sigma, @\}$.

PROOF SKETCH. The proof is carried out in classical second-order logic, and is afterwards translated by a double-negation interpretation into intuitionistic second-order logic.

Although the full proof from [**125**] works independently of interpretation, this sketch works in the intended model $\mathscr{M}_0$.

Let $G_0, G_1, \ldots$ be an enumeration of sentences, and let a be a given sentence. We define by recursion a sequence of sentences $\phi_0, \phi_1, \ldots$ by:

$$\begin{aligned}
\phi_0 &:= a\\
\phi_{n+1} &:= \phi_n && \text{if } (G_n \hookrightarrow \phi_n) \hookrightarrow \phi_n \in J\\
\phi_{n+1} &:= (G'_n\{c/x\} \hookrightarrow \phi_n) \hookrightarrow \phi_n && \text{otherwise, if } G_n \text{ is of form } \forall x G'_n\\
\phi_{n+1} &:= (G_n \hookrightarrow \phi_n) \hookrightarrow \phi_n && \text{otherwise}
\end{aligned}$$

where c is a constant symbol not appearing in ϕ_n, G_n. Now, define

$$\widetilde{M} := \{\phi \mid \exists n.((\phi \hookrightarrow \phi_n) \hookrightarrow \phi_n) \in J\}.$$

It rests to show that $\mathrm{Mod}(\widetilde{M})$ and $a \notin \widetilde{M}$. □

In his article, Krivine also attempts to solve the "specification problem" for TC, that is, to determine the common operational behaviour of *all* different programs that correspond to proofs of TC. He claims that the specification of TC is the one of an "interactive disassembler equipped with protection for system calls".

1.1.5. The proof of Berardi and Valentini. Berardi and Valentini "reverse engineered" Krivine's proof into a more conventional and less formal one, at the same time isolating what they see as the main principle behind, a constructive version of the Ultra-filter Theorem for countable Boolean algebras.

Krivine uses a notion of truth which is not the standard one, namely, there is no requirement that $\bot$ be not true in a model. This, as he himself remarks, means that, classically, there is exactly one model which is not a standard Tarski model, the all-true model. Additionally, Berardi and Valentini remark that, due to a result of McCarthy [**135**], if completeness of classical predicate logic with respect to standard Tarski models was provable intuitionistically, then there would be an intuitionistic proof of Markov's Principle; however, Markov's Principle is independent of intuitionistic logic (Heyting Arithmetic) [**119**].

A similar phenomenon happens with intuitionistic completeness of intuitionistic logic (that will be treated in Chapter 3): in order to avoid the metamathematical results of Gödel and Kreisel [**120**], Veldman [**175**] has to give special treatment to $\bot$ in the semantics by allowing "exploding" nodes that can validate $\bot$.

1.1.12. Definition. A *minimal model* is a set of sentences M which is:

- *implication-faithful*: $A \Rightarrow B \in M \leftrightarrow (A \in M \to B \in M)$;
- *for-all-faithful*: for every formula A with at most one free variable x, $\forall x.A(x) \in M \leftrightarrow$ for any closed term t, $A(t) \in M$;
- *meta-DN*: $(\neg A \in M \to \bot \in M) \to A \in M$

1.1.13. Definition. A *standard model* is a minimal model M with the additional property that $\bot \notin M$.

1.1.14. Remark. For any standard model M there corresponds a model $\mathscr{M}$ in the sense of Definition 1.1.5, and vice versa. What we call "standard model" is known as "the theory" of a Tarski model, that is the set of all sentences true in the Tarski model.

In [**29**], Berardi and Valentini prove in detail their constructive version of the Ultra-filter theorem, and outline how a Henkin-style proof based on it should look like. In the following sections we give a detailed proof of both the Ultra-filter Theorem and the completeness theorem, generalising slightly the Ultra-filter Theorem to setoids (sets equipped with an equality relation which is not necessarily substitutive).

1.2. Constructive ultra-filter theorem

1.2.1. Definition. A *Countable Boolean Algebra over a setoid* $(B, \doteq)$, $\mathscr{B}$, consists of an interpretation of the constants $\{\dot\wedge, \dot\vee, \dot\bot, \dot\top, \dot\neg, \ulcorner\cdot\urcorner\}$ which satisfies the following axioms.

$$
\begin{array}{rr}
x\dot\wedge x \doteq x & (x\dot\vee y)\dot\wedge z \doteq (x\dot\wedge z)\dot\vee(x\dot\wedge z) \\
x\dot\vee x \doteq x & (x\dot\wedge y)\dot\vee z \doteq (x\dot\vee z)\dot\wedge(x\dot\vee z) \\
x\dot\wedge y \doteq y\dot\wedge x & \dot\bot\dot\wedge x \doteq \dot\bot \\
x\dot\vee y \doteq y\dot\vee x & \dot\bot\dot\vee x \doteq x \\
x\dot\wedge(y\dot\wedge z) \doteq (z\dot\wedge y)\dot\wedge z & \dot\top\dot\wedge x \doteq x \\
x\dot\vee(y\dot\vee z) \doteq (z\dot\vee y)\dot\vee z & \dot\top\dot\vee x \doteq \dot\top \\
x\dot\wedge(x\dot\vee y) \doteq x & x\dot\wedge\dot\neg x \doteq \dot\bot \\
x\dot\vee(x\dot\wedge y) \doteq x & x\dot\vee\dot\neg x \doteq \dot\top \\
\ulcorner x\urcorner = \ulcorner y\urcorner \to x = y &
\end{array}
$$

1.2.2. Fact. The following defines a partial order on $\mathscr{B}$:

$$x \dot\leq y := (x\dot\wedge y)\doteq x.$$

We will now need to talk about a collection F of elements of B. Although we think of it as a predicate over B, we will use the notation $F \subseteq B$ and say that F is a subset. We will denote interchangeably by Fx and $x \in F$ membership in F. No use of a power-set axiom is made.

1.2.3. Definition. A subset $F \subseteq B$ is called a *filter* if it is:

- *inhabited*: $\exists x : B, Fx$
- *upwards closed*: $\forall x\, y : B, Fx \to x \dot\leq y \to Fy$
- *meet-closed*: $\forall x\, y : B, Fx \to Fy \to F(x\dot\wedge y)$

1.2.4. Definition. If $X \subseteq B$, the *closure of* X, $\uparrow X$, is the set of all elements of B which are greater than some finite meet of elements of X i.e.

$$\uparrow X := \lambda b.\exists y_1, \ldots, y_n \in X.\ y_1\dot\wedge\cdots\dot\wedge y_n \dot\leq b$$

1.2.5. Definition. $X \subseteq F$ is *inconsistent* if $X\dot\bot$, and $X, Y \subseteq B$ are *equiconsistent* $(X \sim Y)$ if $X\dot\bot \leftrightarrow Y\dot\bot$.

1.2.6. Definition. $X \subseteq B$ is *element-complete for* $b \in B$ if

$$(X \sim \uparrow(X \cup \{b\})) \to b \in X.$$

X is *complete* if it is element-complete for all $b \in B$.

1.2.7. Remark. Note that this definition of "complete", classically equivalent to the more usual one (for all b, either $b \in F$ or $\dot\neg b \in F$), is key to having a constructive proof of the Ultra-filter Theorem.

1.2.8. Fact. We list without proof some easy properties of filters. These are proved in the Coq formalisation.

(1) For every $X \subseteq B$, the closure $\uparrow X$ is a filter.
(2) If F is a filter, then $\dot\top \in F$.
(3) For any $X \subseteq B$, $X \subseteq \uparrow X$.

(4) If $X \subseteq Y \subseteq B$, then $\uparrow X \subseteq \uparrow Y$.
(5) If F is a filter, then $F = \uparrow F$.

1.2.9. Proposition. *If $F \subseteq B$ is a filter, then $x \doteq y$ and $x \in F$ imply $y \in F$.*

PROOF. Immediate from F being upwards closed. □

We will need the following definition and properties for the proof of the Ultra-filter Theorem.

1.2.10. Definition. Let F be a filter. Using the enumeration $\ulcorner\cdot\urcorner : B \to \mathbb{N}$, define the primitive-recursive fixed point $F_n \subseteq B$ by

$$F_0 := F$$
$$F_{n+1} := \lambda b.\, \uparrow(F_n b \vee (\ulcorner b \urcorner = n \wedge F_n \sim \uparrow(F_n \cup \{b\}))) .$$

1.2.11. Lemma. *For every n, F_n is a filter.*

PROOF. A simple induction on n using Fact 1.2.8(1). □

1.2.12. Lemma. *If $n \leq m$, then $F_n \subseteq F_m$.*

PROOF. By induction on the generation of the relation $\leq$ (because $\leq$ has an inductive definition), and using Fact 1.2.8(1). □

1.2.13. Lemma. *For every n, $F_0 \sim F_n$. For every n, m, $F_n \sim F_m$. For any k, and Z as defined in the next theorem, $Z \sim F_k$.*

PROOF. By induction on n, we prove the first part, the other two are easy consequences of it.

The base case is immediate. Let $F \sim F_n$ and let $\dot\bot \in F_{n+1}$. We have to prove that $\dot\bot \in F$. By definition, $\dot\bot \in F_{n+1}$ means that

$$\exists y_1, \ldots, y_l \in \{F_n \cup \{b \mid \ulcorner b \urcorner = n \wedge F_n \sim \uparrow(F_n \cup \{b\})\}\} .\ y_1 \dot\wedge \cdots \dot\wedge y_l \dot\leq \dot\bot .$$

Now, either all of y_i belong to F_n, in which case $\dot\bot \in F_n \sim F$; or, some of them are equal to a b such that $\ulcorner b \urcorner = n$, but, in that case, $\dot\bot \in \uparrow(F_n \cup \{b\}) \sim F_n \sim F$. □

1.2.14. Theorem. *If F is a filter, then $Z := \lambda b.\exists n.F_n b = \cup_{n\in\mathbb{N}} F_n$ is a complete filter extending F, that is equiconsistent with F.*

PROOF. Z is a filter: it is inhabited because F_0 is inhabited, it is upwards closed because each one of F_n is (Lemma 1.2.11), and Z is meet-closed because of Lemma 1.2.12.

$Z \sim F$ because of Lemma 1.2.13.

To show that Z is complete, let $x : B$ with $Z \sim \uparrow(Z \cup \{x\})$ be given. We show that $x \in Z$, by showing that $F_n \sim \uparrow(F_n \cup \{x\})$, when $n = \ulcorner x \urcorner$. Direction $F_n \dot\bot \to \uparrow(F_n \cup \{x\}) \dot\bot$ follows from Fact 1.2.8(3). Let $\uparrow(F_n \cup \{x\}) \dot\bot$. Then $\uparrow(Z \cup \{x\}) \dot\bot$, by Fact 1.2.8(4). By equiconsistency with Z, we have $Z \dot\bot$. By Lemma 1.2.13, we get $F_n \dot\bot$. □

The theorem we just proved is the one that is used for proving the completeness theorem of the next section. We now proceed to give a more familiar form of the Ultra-filter Theorem as a corollary.

1.2.15. Definition. A filter H is an *ultra-filter* if, whenever G is a filter such that $H \sim G$ and $H \subseteq G$, then also $G \subseteq H$.

1.2.16. Corollary. *For any starting filter F, $Z(F)$ is an ultra-filter.*

In particular, if F is consistent ($\dot{\bot} \notin F$), we get a proof of a standard formulation of the Ultra-filter Theorem.

PROOF. Let G be a filter such that $Z(F) \subseteq G$ and $Z(F) \sim G$. To prove that $G \subseteq Z(F)$, let $a \in G$ and use the completeness of $Z(F)$. We have to show that $Z(F) \sim \uparrow(Z(F) \cup \{a\})$. One direction is obvious, for the other, let $\dot{\bot} \in \uparrow(Z(F) \cup \{a\})$. From the hypotheses and Fact 1.2.8, we get that $\dot{\bot} \in \uparrow G = G \sim Z(F)$. □

1.3. Constructive Henkin-style proof

We now proceed to the constructive completeness proof *à la* Henkin. The main difference with other such proofs is that we do not build an infinite extension of signatures $S_0, S_1, \ldots$ explicitly, as Henkin does, but instead extend the grammar of formulae with a separate class of constants, Henkin constants. In the end, when completeness is stated, we require that the input formula contains no Henkin constants, hence the completeness theorem works only for standard formulae.

1.3.1. Definition. The *extended language of signature K* consists of *extended formulae*, which are built up inductively, using individual variables, standard logical constants, the symbols of K, and a special constant symbol c_A for each formula A:

- $\bot$ is a formula
- $P_i(t_1, \ldots, t_n)$ is a formula, if $t_1, \ldots, t_n$ are *terms*;
  - an individual variable x is a term;
  - an individual constant c_j of K is a term;
  - a *Henkin constant* c_A, for A-formula, is a term;
  - if $t_1, \ldots, t_m$ are terms, so is $f_l(t_1, \ldots, t_m)$.
- if A, B are formulae, then $A \wedge B$, $A \vee B$, and $A \to B$ are formulae;
- if A is a formula, possibly containing x as a free individual variable, then $\exists x A$ and $\forall x A$ are formulae.

In other words, the extended formulae are built from three kinds of expressions: formulae can be constructed from terms, terms can be constructed from constants, and constants can be constructed from formulae.

For a derivation system we take the one of Table 1.

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ \text{Ax}$$

$$\frac{\Gamma, A_1 \vdash A_2}{\Gamma \vdash A_1 \Rightarrow A_2}\ \Rightarrow_I \qquad \frac{\Gamma \vdash A_1 \Rightarrow A_2 \qquad \Gamma \vdash A_1}{\Gamma \vdash A_2}\ \Rightarrow_E$$

$$\frac{\Gamma \vdash A \qquad x\text{-fresh}}{\Gamma \vdash \forall x A}\ \forall_I \qquad \frac{\Gamma \vdash \forall x A}{\Gamma \vdash A\{t/x\}}\ \forall_E$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot_E \qquad \frac{\Gamma \vdash (A \Rightarrow \bot) \Rightarrow \bot}{\Gamma \vdash A}\ \neg\neg_E$$

Table 1: Classical natural deduction with $\{\Rightarrow, \forall, \bot\}$

1.3.2. Fact. Given a derivation $\Gamma \vdash A$, we can replace a constant c with a fresh variable x, obtaining a derivation $\Gamma\{x/c\} \vdash A\{x/c\}$.

The proof of this fact is very technical, but since it is well known and we did prove it formally in Coq, we leave it out.

1.3.3. Definition. Given a set of sentences (axioms) $\mathscr{A}$, the *theory of* $\mathscr{A}$, $\mathrm{Th}(\mathscr{A})$, is the set of formulae which are derivable from the axioms.

1.3.4. Definition. The set of *Henkin-axioms* $\mathscr{H}$ is the set of formulae of form $A(c_{\forall xA}) \Rightarrow \forall xA$ for A such that $\forall xA$ is closed.

1.3.5. Definition. A set of formulae $\mathscr{T}$ is *Henkin-complete* if $\mathscr{H} \subseteq \mathscr{T}$.

1.3.6. Lemma. *If $\mathscr{A}$ is a set of sentences (axioms) which contains no Henkin constants, then the theories* $\mathrm{Th}(\mathscr{A} \cup \mathscr{H})$ *and* $\mathrm{Th}(\mathscr{A})$ *are equiconsistent.*

PROOF. That $\bot \in \mathrm{Th}(\mathscr{A}) \to \bot \in \mathrm{Th}(\mathscr{A} \cup \mathscr{H})$ is clear. Let $\bot \in \mathrm{Th}(\mathscr{A} \cup \mathscr{H})$ i.e. $\Gamma \vdash \bot$ for some $\Gamma \in \mathscr{A} \cup \mathscr{H}$. We show that we can eliminate all Henkin axioms from Γ, by showing that we can eliminate one at a time, when the Henkin constant the axiom is based on does not appear in the other formulae of the context. Once we show that, we just need to reorder[3] the Henkin axioms from Γ so that the most complex ones are eliminated first.

Let $(A(c_{\forall xA}) \Rightarrow \forall xA), \Delta \vdash \bot$, where the constant $c_{\forall xA}$ does not appear in Δ. We want to show $\Delta \vdash \bot$. It will be enough to show that $\Delta \vdash \forall y \neg (A \Rightarrow \forall xA)$, because this produces a contradiction with a tautology of classical predicate logic, the Drinker paradox:[4]

$$\vdash \neg \forall y \neg (A \Rightarrow \forall xA)$$

Let y be a fresh variable. We need to show that $\Delta \vdash \neg(A \Rightarrow \forall xA)(y)$. Since $\forall xA$ is closed, we can rewrite the hypothesis $(A(c_{\forall xA}) \Rightarrow \forall xA), \Delta \vdash \bot$ as $\Delta \vdash \neg(A \Rightarrow \forall xA)(c_{\forall xA})$. Now, since $c_{\forall xA}$ does not appear in Δ, it is enough to apply Fact 1.3.2. □

1.3.7. Definition. The Lindenbaum Boolean algebra $\{\mathrm{B}, \doteq, \dot\wedge, \dot\vee, \dot\bot, \dot\top, \dot\neg, \ulcorner\cdot\urcorner\}$ is defined by:

$$\begin{aligned}
&\mathrm{B} \text{ is the set of closed formulae}\\
A_1 \doteq A_2 &\text{ iff } \vdash A_1 \Rightarrow A_2 \text{ and } \vdash A_2 \Rightarrow A_1\\
A_1 \dot\wedge A_2 &:= \neg(A_1 \Rightarrow \neg A_2)\\
A_1 \dot\vee A_2 &:= \neg A_1 \Rightarrow A_2\\
\dot\bot &:= \bot\\
\dot\top &:= \bot \Rightarrow \bot\\
\dot\neg A &:= \neg A
\end{aligned}$$

The enumeration $\ulcorner\cdot\urcorner$ is defined via the Cantor pairing function in Section 1.5.2.

1.3.8. Lemma. *Every theory is a filter in the Lindenbaum Boolean algebra.*

PROOF. This is easy to show. For details have a look at the formal proof. □

[3] Actually, the correctness of the reordering(sorting) algorithm is the only part of the proof which was not fully formalised in Coq. See Section 1.5 for more details.

[4] Which can be phrased in natural language by: "There is someone in the pub such that, if he is drinking, everyone in the pub is drinking".

Let $\mathscr{F}_n$ be defined by setting $F_0 := \mathrm{Th}(\mathscr{A} \cup \mathscr{H})$ in the fixpoint of Definition 1.2.10, and let $\mathscr{Z}$ be defined, like in Theorem 1.2.14, as $B \in \mathscr{Z} \leftrightarrow \exists n. B \in \mathscr{F}_n$.

1.3.9. Lemma. *For every n, the filter $\mathscr{F}_n$ is the theory with axiom set $\mathscr{G}_n$, where*

$$\mathscr{G}_0 := \mathscr{A} \cup \mathscr{H}$$
$$\mathscr{G}_{n+1} := \lambda b. (\mathscr{G}_n b \vee (\ulcorner b \urcorner = n \wedge \mathscr{F}_n \sim\!\uparrow(\mathscr{F}_n \cup \{b\}))) .$$

The complete filter $\mathscr{Z}$ is the theory with axiom set $\cup_{n\in\mathbb{N}}\mathscr{G}_n$.

PROOF. Direction "$\rightarrow$" of the first part is by induction on n. $\mathscr{F}_0$ is a theory by definition. Let $\mathscr{F}_n = \mathrm{Th}(\mathscr{G}_n)$ and $A \in \mathscr{F}_{n+1}$. We have to show that there is a derivation $\Gamma \vdash A$ such that $\Gamma \subseteq \mathscr{G}_{n+1}$. By the definition of the fixpoint,

$$A \in \uparrow(\mathscr{F}_n \cup \{b \mid \ulcorner b \urcorner = n \,\&\, \mathscr{F}_n \sim\!\uparrow(\mathscr{F}_n \cup \{b\})\}) ,$$

that is,

$$\exists y_1, \ldots, y_m \in \mathscr{F}_n \cup \{b \mid \ulcorner b \urcorner = n \,\&\, \mathscr{F}_n \sim\!\uparrow(\mathscr{F}_n \cup \{b\})\}.\ y_1 \dot\wedge \cdots \dot\wedge y_m \dot\leq A,$$

and, by the induction hypothesis,

$$\exists z_1^1, \ldots, z_1^{k_1}, z_2^1, \ldots, z_2^{k_2}, \ldots, z_m^1, \ldots, z_m^{k_m} \in \mathscr{G}_{n+1}.\ z_1^1 \dot\wedge \cdots \dot\wedge z_m^{k_m} \dot\leq A,$$

from which the goal follows by the interpretation of $\dot\wedge$ and $\dot\leq$ in the Lindenbaum algebra.

Direction "$\leftarrow$" is immediate because $\mathscr{G}_n \subseteq \mathscr{F}_n$ for every n.

The statement $\mathscr{Z} \subseteq \mathrm{Th}(\cup_{n\in\mathbb{N}}\mathscr{G}_n)$ follows directly from the first part.

To show $\mathrm{Th}(\cup_{n\in\mathbb{N}}\mathscr{G}_n) \subseteq \mathscr{Z}$, let $\Gamma \vdash A$ and $\Gamma \subseteq \cup_{n\in\mathbb{N}}\mathscr{G}_n$. Note that $\Gamma \subseteq \cup_{n\in\mathbb{N}}\mathscr{G}_n$ is just a shortcut for $\forall x A \in \Gamma, A \in \cup_{n\in\mathbb{N}}\mathscr{G}_n$. To show that $A \in \mathscr{Z}$ means to find m such that $A \in \mathscr{F}_m$. We simply need to take for m the maximum n such that all formulae of Γ belong to $\mathscr{F}_n$ (by Lemma 1.2.12). Now, from $\Gamma \vdash A$, we have $\dot\bigwedge\Gamma \dot\leq A$, therefore, since $\mathscr{F}_m$ is a filter, $A \in \mathscr{Z}$.

□

We are now ready to prove the main theorem.

1.3.10. Theorem (Model Existence)**.** *Let $\mathscr{A}$ be a set of axioms that does not contain any Henkin constants. Then $\mathscr{Z}$ is an equiconsistent extension of* $\mathrm{Th}(\mathscr{A})$ *that is implication-faithful, for-all-faithful, and meta-DN.*

PROOF. That $\mathscr{Z}$ is an equiconsistent extension of $\mathrm{Th}(\mathscr{A})$ follows from Lemma 1.3.6 and Theorem 1.2.14. We need to show that $\mathscr{Z}$ is a minimal model.

- *$\mathscr{Z}$ is meta-DN.* Let A be a sentence such that $\neg A \in \mathscr{Z} \rightarrow \bot \in \mathscr{Z}$. We need to show that $A \in \mathscr{Z}$. We will use the fact that $\mathscr{Z}$ is complete, that is,

  $$\mathscr{Z} \sim\!\uparrow(\mathscr{Z} \cup \{A\}) \rightarrow A \in Z.$$

  The direction "$\rightarrow$" of $\mathscr{Z} \sim\!\uparrow(\mathscr{Z} \cup \{A\})$ is trivial. Let $\bot \in \uparrow(\mathscr{Z} \cup \{A\})$. We show that $\bot \in \mathscr{Z}$ by applying the first hypothesis, but then we have to show that $\neg A \in \mathscr{Z}$. By the definition of $\uparrow$ we have that

  $$\exists y_1, y_2, \ldots, y_m \in \mathscr{Z} \cup \{A\}. y_1 \dot\wedge y_2 \dot\wedge \cdots \dot\wedge y_m \dot\leq \bot.$$

  Now, we look at two cases:

  (1) Either all y_i are in $\mathscr{Z}$, and then, by $\mathscr{Z}$ being a filter, $\neg A \in \mathscr{Z}$, because

  $$y_1 \dot\wedge y_2 \dot\wedge \cdots \dot\wedge y_m \dot\leq \bot \dot\leq \neg A.$$

(2) Or, $y_{\rho(1)}, y_{\rho(2)}, \ldots, y_{\rho(m-1)} \in \mathscr{Z}$ and $y_{\rho(m)} = A$, and then, since

$$y_{\rho(1)} \dot{\wedge} y_{\rho(2)} \dot{\wedge} \cdots \dot{\wedge} y_{\rho(m-1)} \dot{\wedge} A \dot{\leq} \bot,$$

we have that

$$y_{\rho(1)} \dot{\wedge} y_{\rho(2)} \dot{\wedge} \cdots \dot{\wedge} y_{\rho(m-1)} \leq \neg A.$$

Since $\mathscr{Z}$ is a filter, $\neg A \in \mathscr{Z}$.

- *$\mathscr{Z}$ is implication-faithful.* We have to show that

$$(A \Rightarrow B) \in \mathscr{Z} \leftrightarrow (A \in \mathscr{Z} \rightarrow B \in \mathscr{Z}).$$

Direction "$\rightarrow$" follows directly by the $\Rightarrow_E$ rule, because $\mathscr{Z}$ is a theory by Lemma 1.3.9.

Let $A \in \mathscr{Z} \rightarrow B \in \mathscr{Z}$. We show $(A \Rightarrow B) \in \mathscr{Z}$ by applying the fact we proved above, that $\mathscr{Z}$ is meta-DN, hence we have to show that

$$\neg(A \Rightarrow B) \in \mathscr{Z} \rightarrow \bot \in \mathscr{Z}.$$

Let $\neg(A \Rightarrow B) \in \mathscr{Z}$. Since $\mathscr{Z}$ is a (classical) theory, we have that $A \in \mathscr{Z}$ and $\neg B \in \mathscr{Z}$. Now, from the hypothesis and $A \in \mathscr{Z}$ we get $B \in \mathscr{Z}$, hence $\bot \in \mathscr{Z}$.

- *$\mathscr{Z}$ is for-all-faithful.* We have to show that, for any formula A such that $\forall x A$ is closed,

$$(\forall x A) \in \mathscr{Z} \leftrightarrow \text{ for any closed term } t, A\{t/x\} \in \mathscr{Z}.$$

Direction "$\rightarrow$" is by the $\forall_E$ rule, because $\mathscr{Z}$ is a theory.

Let for any closed term $t, A(t) \in \mathscr{Z}$. By definition, $\mathscr{Z}$ is Henkin-complete. Then, we can show that $\forall x A \in \mathscr{Z}$ by using implication-fatefulness on the Henkin axiom,

$$A(c_{\forall x A}) \Rightarrow \forall x A,$$

because from the hypothesis we can conclude that $A(c_{\forall x A}) \in \mathscr{Z}$.

□

Let $\Gamma \vDash A$ denote that, for every minimal model $\mathscr{M}$, $\Gamma \subseteq \mathscr{M}$ implies $A \in \mathscr{M}$. We have the following corollary.

1.3.11. Corollary (Completeness)**.** *For any Γ and A that do not contain Henkin constants, if $\Gamma \vDash A$, then $\Gamma \vdash A$.*

PROOF. Let the set of axioms $\mathscr{C}$ be the finite set $\Gamma \cup \{\neg A\}$. By Theorem 1.3.10, there is a minimal model $\mathscr{M} := \mathscr{Z}(\mathscr{C})$ that extends $\mathrm{Th}(\mathscr{C})$ and is equiconsistent with it. Because A is true in any model in which Γ is true, and because $\Gamma \subseteq \mathrm{Th}(\mathscr{C}) \subseteq \mathscr{M}$, we have that $A \in \mathscr{M}$. But, because also $\neg A \in \mathrm{Th}(\mathscr{C}) \subseteq \mathscr{M}$, we get that $\bot \in \mathscr{M}$. Since $\mathscr{M}$ and $\mathrm{Th}(\mathscr{C})$ are equiconsistent, also $\bot \in \mathrm{Th}(\mathscr{C})$, which, by definition, means that $\Gamma, \neg A \vdash \bot$. Hence, $\Gamma \vdash A$. □

1.4. Computational content

The computational content of the completeness proof is to be read off Theorem 1.3.10. We do not give a succinct characterisation of it like the one of Krivine, mentioned at end of section 1.1.4, but instead discuss multiple aspect.

Theorem 1.3.10 builds the model $\mathscr{Z}$ using a fixed-point, starting from $\mathrm{Th}(\mathscr{A} \cup \mathscr{H})$. The built model $\mathscr{Z}$ is equiconsistent with $\mathrm{Th}(\mathscr{A})$, so the Henkin axioms

somehow "disappear". The procedure behind this is described in the proof of Lemma 1.3.6: we take a derivation $\Gamma \vdash A$ that uses Henkin axioms, we order the axioms according to the depth of the formula which is annotating the Henkin constant, and then we eliminate them one by one by using the Drinker's paradox and Fact 1.3.2, which replaces all occurrences of a constant inside a derivation by a free variable.

Let us now pose the question: how does Theorem 1.3.10 "normalise" proofs involving implication, that is, what is the procedure to construct a derivation $\Gamma \vdash B$ from derivations of $\Gamma \vdash A \Rightarrow B$ and $\Gamma \vdash A$. When we think of calculi that satisfy the Brouwer-Heyting-Kolmogorov interpretation [**161**, p.10] of logical connectives, we think of this proof transformation as the β-reduction relation on proof terms.

The statement $A \Rightarrow B \in \mathscr{Z}$ determines an "approximation" of $\mathscr{Z}$, a number n such that $A \Rightarrow B \in \mathscr{F}_n$, and $A \in \mathscr{Z}$ determines a number m such that $A \in \mathscr{F}_m$. Then, the proof that $\mathscr{Z}$ is a theory from Lemma 1.3.9, shows the way to prove that $B \in \mathscr{Z}$: we take the common approximation of $\mathscr{Z}$ of all formulae used in a derivation of B, and return that number. In the simplest case we have above, we return $\max(n, m)$.

In the next chapters we will refer to the statement

$$(A \Rightarrow B) \in \mathscr{Z} \to A \in \mathscr{Z} \to B \in \mathscr{Z}$$

as *reflection*. The converse,

$$(A \in \mathscr{Z} \to B \in \mathscr{Z}) \to (A \Rightarrow B) \in \mathscr{Z},$$

will be referred to as *reification*. In Theorem 1.3.10, reification is proved via the meta-DN property of $\mathscr{Z}$,

$$\text{for any formula } C, (\neg C \in \mathscr{Z} \to \bot \in \mathscr{Z}) \to C \in \mathscr{Z}.$$

Using reflection, we can transform this to

$$\text{for any formula } C, ((C \in \mathscr{Z} \to \bot \in \mathscr{Z}) \to \bot \in \mathscr{Z}) \to C \in \mathscr{Z},$$

from which we see that meta-DN transforms a higher-order proof, written in a kind of continuation-passing style [**143**], into a flat classical proof.

1.5. Aspects of the Coq formalisation

The source code of the formalisation is available at the address `http://www.lix.polytechnique.fr/~danko/code`.

As we mentioned in the introduction, the formalisation is not complete. The missing part is the correctness of the sorting algorithm needed in Lemma 1.3.6. The reason why this part remained unfinished is that in the formal proof of the lemma, we used a sorting specification which was convenient for the proof, but was an *ad hoc* specification as far as general list sorting is concerned, thus we could not reuse the already proven results about general list sorting in Coq. For lack of time, we contented ourselves by testing in Coq that the sorting algorithm computes as expected.

1.5.1. Formal syntax of formulae and co-finite rule for $\forall_I$. The syntax of formulae is defined using the following inductive datatype.

```
Parameters function predicate constant0 : Set.
Inductive formula : Set :=
  | bot : formula
  | imp : formula → formula → formula
  | all : formula → formula
  | atom : predicate → term → formula
  with term : Set :=
  | bvar : nat → term
  | fvar : nat → term
  | cnst : constant → term
  | func : function → term → term
  with constant : Set :=
  | original : constant0 → constant
  | added : formula → constant.
```

One thing to notice is that we have a truly mutually inductive data-type, the clause for constants depending on the one for formulae, because of Henkin constants (the ones with marker added).

Another thing to notice is that the predicate and function constructors, atom and func, take only a single term as an argument. For the completeness proof to be practically useful in Coq, however, an extension to multi-argument constructors would be necessary. Our goal has been a more theoretical one, to see the details and computational contents of a formal type-theoretic completeness proof.

A third thing to comment about is the formal handling of variables. This is an often neglected aspect of informal proofs, but quite an important one when formalisation is concerned. In the theory of programming languages there is a community effort in progress, for definite settling on good formalisation practises connected to variable binding, known as the POPLMark challenge [**183**]. Following one of the most successful approaches to binders, the "locally nameless" representation [**18**], we represent bound variables and free variables separately, via two separate constructors, bvar and fvar. bvar-s range over numbers, deBruijn indices, while fvar-s can range over any set with decidable equality, for example the set of character strings. Thus, formally, a formula $\forall x \forall y (P(x) \Rightarrow Q(x))$ is represented by `all (imp (P (bvar 0)) (Q (bvar 1)))`, and a formula $\forall x (P(x) \Rightarrow \forall y Q(y))$ is represented by `all (imp (P (bvar 0)) (all (Q (bvar 0))))`. Substitutions are defined via the following fixpoint.

```
Fixpoint
  open_rec (k : nat) (u : term) (t : formula) {struct t} : formula :=
  match t with
    | bot ⇒ bot
    | imp t1 t2 ⇒ imp (open_rec k u t1) (open_rec k u t2)
    | all t1 ⇒ all (open_rec (S k) u t1)
    | atom p t1 ⇒ atom p (open_rec_term k u t1)
  end
with
```

```
  open_rec_term (k : nat) (u : term) (t : term) {struct t} : term :=
  match t with
    | bvar i ⇒ if beq_nat k i then u else (bvar i)
    | fvar x ⇒ fvar x
    | cnst c ⇒ cnst c
    | func f t1 ⇒ func f (open_rec_term k u t1)
  end.

Definition open t u := open_rec 0 u t.
Notation "t ^^ u" := (open t u) (at level 67).
Notation "t ^ x" := (open t (fvar x)).
```

The important point with this representation of variables is that we are able to state the $\forall_I$ rule in a co-finite way. Instead of saying that $\forall x A$ is derivable when $A(x)$ is derivable for some fresh x, we say that $\forall x A$ is derivable when there exists a finite list of variables L, such that for every $x \notin L$, $A(x)$ is derivable. This later form is more convenient than the former one when doing formal proofs by induction on the derivation, because the induction hypothesis is of a more flexible form.

Outside of programming language research, this rule has been used in the context of traditional logic at least by Krivine in [**124**] to characterise α-conversion of System F types (formulae).

1.5.2. Enumeration of formulae. The enumeration of formulae needed for defining the Lindenbaum algebra of Lemma 1.3.7 was defined using the Cantor pairing function [**182**]. In particular, we were able to use an already existing implementation and correctness proof of the pairing function in Coq, coming from the formalisation of Gödel's incompleteness theorem of O'Connor [**146**].

The definition of the enumeration goes via a mutually recursive fixpoint on formulae, terms and constants.

```
Section Enumeration.
  Add LoadPath "pairing".
  Require Import cPair.

  Definition enump := fun p ⇒ cPair 11 (enum_predicate p).
  Definition enumc0 := fun c ⇒ cPair 12 (enum_constant0 c).
  Definition enumfunc := fun f ⇒ cPair 13 (enum_function f).

  Fixpoint enumf (f:formula) : nat :=
    match f with
      | (atom p t) ⇒ cPair 1 (cPair (enump p) (enumt t))
      | (all g) ⇒ cPair 2 (enumf g)
      | (imp g h) ⇒ cPair 3 (cPair (enumf g) (enumf h))
      | bot ⇒ 4
    end
  with enumt (t:term) : nat :=
    match t with
      | (func phi t') ⇒ cPair 5 (cPair (enumfunc phi) (enumt t'))
      | (cnst c) ⇒ cPair 6 (enumc c)
      | (fvar x) ⇒ cPair 7 x
      | (bvar x) ⇒ cPair 8 x
    end
```

```
with enumc (c:constant) : nat :=
  match c with
    | (added x) ⇒ cPair 9 (enumf x)
    | (original x) ⇒ cPair 10 (enumc0 x)
  end.

Eval compute in (enumf (imp bot bot)).

Scheme Induction for formula Sort Prop
with Induction for term Sort Prop
with Induction for constant Sort Prop.

Theorem countable_ftc :
  (∀ f g, enumf f = enumf g → f = g)
  ∧ (∀ t s, enumt t = enumt s → t = s)
  ∧ (∀ c k, enumc c = enumc k → c = k).

Definition enum := enumf.

Definition countable : ∀ x y, enum x = enum y → x = y
  := proj1 countable_ftc.

End Enumeration.
```

1.5.3. Representation of finite quantifications. The constructions of form

$$\exists y_1, \ldots, y_n \in X.\ y_1 \wedge \cdots \wedge y_n \leq z$$

were represented using finite lists and the standard fold-left function from functional programming, that takes a list and a function, and applies cumulatively the function to all members of the list from left to right. For example, the closure of X, $\uparrow X$, was encoded by:

```
Definition up (X:B→Prop) := fun z:B ⇒
  ∃ n:nat, ∃ ys:list B, length ys = n ∧
    fold_left (fun (a:Prop)(b:B) ⇒ and a (X b)) ys True ∧
    leq (fold_left meet ys top) z.
```

Choosing this representation required a few technical lemmas which were tricky to prove, but overall we are satisfied with the choice, because it gave us the possibility to discharge parts of the proof by pure computation, a technique in the type-theoretic jargon known as "proof by reflection".

1.5.4. Setoids and Prop versus Set; the Ring tactic. Subsets of the abstract Boolean algebra B are represented as predicates, that is propositional functions $B \to$ Prop. This decision propagates to the definition of model, where sets of formulae are also represented as propositional functions. The choice of Prop instead of Set, is motivated by practical rather than mathematical arguments (no use of impredicativity is being made). The reason is simply that, at the time when the formalisation was being carried out, the Coq proof assistant supported only setoids where the equivalence relation is in Prop, not in Set.

We were able to instantiate the `Ring` tactic of Coq with the semi-ring structure of B, and to use it to automatically resolve a couple of complex equations in B.

1.5.5. Other aspects of the formalisation. In retrospective, we do not think that many of the components of the formalisation can be implemented in a substantially better way. One exception is the handling of theories related to the Model Existence Lemma. There, we formally manipulated simultaneously both an axiom set and a theory over that axiom set, while the later is predetermined by the former, which means that the corresponding formal proofs can probably be cut in size. This problem is not noticeable in the informal version given in section 1.3.

1.6. Related and future work

In section 1.1 we reviewed the origins of our work. There is a number of other related works, which we mention here.

The proof of Krivine, which is actually a complete formalisation on paper, was checked by Raffalli in the PhoX proof assistant. [**152**]

In a series of articles [**42, 37, 43, 41, 38, 39, 40**], Braselmann and Koepke describe their formalisation of Gödel's completeness theorem in the proof assistant Mizar [**165**], which is based on Tarski-Grothendieck set theory, an extension of the Zermelo-Fraenkel set theory.

Russell O'Connor has formalised the incompleteness theorem of Gödel inside the Coq proof assistant [**146**]. We reused from his formalisation a part of the source code which defines the Cantor pairing function and proves it bijective.

In future, we would like to finish the correctness proof of the sorting algorithm, and to allow multi-argument predicate and function symbols in the syntax of formulae, so that the Coq formalisation becomes practically usable.

CHAPTER 2

Kripke-style models for classical logic

We saw that the computational content of the completeness theorem presented in Chapter 1 reduces to finding a sufficiently large approximation to the ultrafilter $\mathcal{Z}$, that is enough to contain all relevant formulae (in a derivation we can only use finitely many formulae).

From today's perspective, when we know that proof terms for classical logic also have a computational behaviour similar to λ-calculus, the computational contents presented in Section 1.4 seems rather ad hoc: instead of β-reduction for the modus ponens rule, we have the $\max(m,n)$ operation, whose outcome depends on the particular way one defines the enumeration of formulae.

In this chapter we present work that was started as a general framework for more canonical treatment of completeness proofs, based on the observation of Herbelin that completeness is just one aspect of a normalisation-by-evaluation (NBE) proof.

In Section 2.1, we will review NBE and explain what kind of completeness is hidden behind it. In Section 2.2, we will introduce the $\mathrm{LK}_{\mu\tilde{\mu}}$ classical sequent calculus. In Section 2.3, we will introduce, by analogy with NBE for intuitionistic logic, a notion of model, similar to Kripke models, with respect to which we can prove soundness and completeness of $\mathrm{LK}_{\mu\tilde{\mu}}$. In Section 2.4, we will present a dual notion of model, which is also proved sound and complete. In Section 2.5, we will discuss the computational content of the two completeness proofs; it is that of a cut-elimination procedure, each of the two models defining a different cut-elimination strategy. In Section 2.6, we discuss some aspects of the Coq formalisation of the proofs of this chapter, and, finally, we conclude in Section 2.7, by a discussion of related and future work.

2.1. Normalisation-by-evaluation as completeness

In the proof theoretic study of λ-calculi and Semantics of programming languages, normalisation is a property of abstract rewrite systems [**180**]. An abstract rewrite system is normalising if every sequence of rewrite steps described by it is finite. An instance is the simply typed λ-calculus with rewriting defined by β-reduction. Another case that can be fit into this framework are the proofs of normalisation of natural deduction calculi, thanks to the Curry-Howard correspondence with λ-calculi.

Normalisation-by-evaluation (NBE) is a technique, introduced by Berger and Schwichtenberg in [**33**], for proving that a calculus of proof terms is normalising without working directly with the reduction/rewrite relation of an abstract rewrite system. Instead, the proof terms are interpreted in the ambient metalanguage, and from that interpretation a proof term in normal form is extracted. The desired equality between the starting and the ending proof term is proved,

typically $\beta\eta$-equality. The trick is to avoid reasoning with the reduction relation of the object language by relying on reduction provided by the ambient language.

NBE was first used in [**33**] to show normalisation of simply typed lambda calculi, by giving an "inverse" to the evaluation function,

$$[\![-]\!] : \Lambda \to D,$$

from Church-style simply typed lambda terms into some ambient language, or denotational model. The inverse function $\downarrow$, called *reification*, is defined by recursion on the type τ of the term, at the same time defining an auxiliary function $\uparrow$, called *reflection*:

$$\begin{aligned}
\downarrow^{\tau} &: D \to \Lambda\text{-nf} && \\
\downarrow^{\tau} &:= a \mapsto a && \tau\text{-atomic} \\
\downarrow^{\tau\to\sigma} &:= S \mapsto \lambda a.\, \downarrow^{\sigma} (S \cdot \uparrow^{\tau} a) && a\text{-fresh}
\end{aligned}$$

$$\begin{aligned}
\uparrow^{\tau} &: \Lambda\text{-ne} \to D && \\
\uparrow^{\tau} &:= a \mapsto a && \tau\text{-atomic} \\
\uparrow^{\tau\to\sigma} &:= e \mapsto S \mapsto \uparrow^{\sigma} e(\downarrow^{\tau} S) &&
\end{aligned}$$

The kinds of λ-terms that appear as a (co)domain can be sorted out according to the following inductive definition:

$$\begin{aligned}
\Lambda \ni p, q \quad &:= a^{\tau} \mid \lambda a^{\tau}.p^{\sigma} \mid p^{\tau\to\sigma} q^{\tau} && \lambda\text{-terms} \\
\Lambda\text{-nf} \ni r \quad &:= \lambda a^{\tau}.r^{\sigma} \mid e^{\tau} && \lambda\text{-terms in normal form} \\
\Lambda\text{-ne} \ni e \quad &:= a^{\tau} \mid e^{\tau\to\sigma} r^{\tau} && \text{neutral } \lambda\text{-terms}
\end{aligned}$$

Obviously, $\Lambda\text{-ne} \subseteq \Lambda\text{-nf} \subseteq \Lambda$.

In the above definition we used S to range over members of D, and we used $\mapsto$ and $\cdot$ for abstraction and application at the meta-level.

A term in normal form p' is then computed from any term p with type τ, by first evaluating p, and then extracting p' directly from the denotation of the evaluation, that is, by setting

$$p' := \downarrow^{\tau} [\![p]\!].$$

Berger and Schwichtenberg proved that the result is correct i.e. that $p' =_{\beta\eta} p$.

It was a subsequent realisation, which we trace back to Catarina Coquand [**48, 49**], and Coquand and Dybjer [**51**], and which is explicitly present especially in [**49**], that an NBE algorithm can be seen as a composition of a soundness theorem with a completeness theorem for Kripke semantics [**123, 117**].

2.1.1. Definition. A *Kripke model* is given by a preorder $(K, \leq)$ of possible worlds, a binary relation of *forcing* $(-) \Vdash (-)$ between worlds and atomic formulae, and a *family of domains of quantification* $D(-)$, such that,

$$\begin{aligned}
&\text{for all } w' \geq w, w \Vdash X \to w' \Vdash X, \text{ and} \\
&\text{for all } w' \geq w, D(w) \subseteq D(w').
\end{aligned}$$

The relation of forcing is then extended from atomic to composite formulae by the clauses:

$$\begin{aligned} w \Vdash A \wedge B &:= w \Vdash A \text{ and } w \Vdash B \\ w \Vdash A \vee B &:= w \Vdash A \text{ or } w \Vdash B \\ w \Vdash A \Rightarrow B &:= \text{for all } w' \geq w, w' \Vdash A \Rightarrow w' \Vdash B \\ w \Vdash \forall x.A(x) &:= \text{for all } w' \geq w \text{ and } t \in D(w'), w' \Vdash A(t) \\ w \Vdash \exists x.A(x) &:= \text{for some } t \in D(w), w \Vdash A(t) \\ w \Vdash \bot &:= \text{false} \\ w \Vdash \top &:= \text{true} \end{aligned}$$

Of course, for NBE of simply typed λ-calculus, only the part for implication is used from the above definition in the connection to completeness. Conjunction and $\forall$ can be easily accounted for, while the connection between NBE and completeness for full intuitionistic predicate logic, with $\vee$ and $\exists$, is a subject of Chapter 3.

We denote by $\Gamma \vdash p : A$ the typability of the term p with type A and open variables listed in Γ, and, by the Curry-Howard correspondence, $\Gamma \vdash p : A$ also denotes the derivability of the formula A from the hypotheses in Γ. Let $w \Vdash \Gamma$ mean that $w \Vdash B$ for any $B \in \Gamma$.

The evaluation function $[\![-]\!] : \Lambda \to D$ can be written in the form of a Soundness theorem, as follows.

2.1.2. Theorem (Soundness)**.** *If $\Gamma \vdash p : A$ then, in any Kripke model, for any world w, if $w \Vdash \Gamma$ then $w \Vdash A$.*

PROOF. By a simple induction on the derivation. □

The reification function takes the form of Theorem 2.1.4. The proof goes via a model constructed from components of a derivation system, as described in the following lemma.

2.1.3. Lemma (Model Existence)**.** *There is a model $\mathscr{U}$ (the "universal model") such that, if at every world w of $\mathscr{U}$, $w \Vdash \Gamma$ implies $w \Vdash A$, then there exists a term p and a derivation $\Gamma \vdash p : A$.*

PROOF. The universal model $\mathscr{U}$ is built by setting:

- K to be the set of contexts Γ, that is finite sets of variable declarations $(a : A)$;
- "$\leq$" to be the subset relation of contexts;
- "$\Gamma \Vdash X$" to be $\Gamma \vdash X$, for X an atomic formula.

We than prove simultaneously, by induction on the complexity of A, that the two functions defined above, reify ($\downarrow$) and reflect ($\uparrow$), are correct, that is, that $\downarrow$ maps a member of $\Gamma \Vdash A$ to a normal proof term (derivation) $\Gamma \vdash p : A$, and $\uparrow$ maps a neutral term (derivation) $\Gamma \vdash e : A$ to a member of $\Gamma \Vdash A$.

The proof is easy and has been formalised in Alf [**49**] and Coq [**102, 156**]. □

2.1.4. Theorem (Completeness)**.** *If in any Kripke model, at any world w, $w \Vdash \Gamma$ implies $w \Vdash A$, then there exists a term p and a derivation $\Gamma \vdash p : A$.*

PROOF. If $w \Vdash B$ in any Kripke model, then also $w \Vdash B$ in the model $\mathscr{U}$ above, hence there exists a term p such that $\Gamma \vdash p : A$. □

$$\frac{}{\Gamma|A\vdash A,\Delta}\ (\mathrm{Ax}_L) \qquad \frac{}{A,\Gamma\vdash A|\Delta}\ (\mathrm{Ax}_R)$$

$$\frac{\Gamma,A\vdash\Delta}{\Gamma|A\vdash\Delta}\ (\tilde{\mu}) \qquad \frac{\Gamma\vdash A,\Delta}{\Gamma\vdash A|\Delta}\ (\mu)$$

$$\frac{\Gamma\vdash A|\Delta \qquad \Gamma|B\vdash\Delta}{\Gamma|A\Rightarrow B\vdash\Delta}\ (\Rightarrow_L) \qquad \frac{\Gamma,A\vdash B|\Delta}{\Gamma\vdash A\Rightarrow B|\Delta}\ (\Rightarrow_R)$$

$$\frac{\Gamma|A\vdash\Delta \qquad \Gamma|B\vdash\Delta}{\Gamma|A\vee B\vdash\Delta}\ (\vee_L) \qquad \frac{\Gamma\vdash A_i|\Delta}{\Gamma\vdash A_1\vee A_2|\Delta}\ (\vee^i_R)$$

$$\frac{\Gamma|A\vdash\Delta}{\Gamma|A_1\wedge A_2\vdash\Delta}\ (\wedge^i_L) \qquad \frac{\Gamma\vdash A|\Delta \qquad \Gamma\vdash B|\Delta}{\Gamma\vdash A\wedge B|\Delta}\ (\wedge_R)$$

$$\frac{\Gamma|A(x)\vdash\Delta \qquad x\text{ fresh}}{\Gamma|\exists xA(x)\vdash\Delta}\ (\exists_L) \qquad \frac{\Gamma\vdash A(t)|\Delta}{\Gamma\vdash\exists xA(x)|\Delta}\ (\exists_R)$$

$$\frac{\Gamma|A(t)\vdash\Delta}{\Gamma|\forall xA(x)\vdash\Delta}\ (\forall_L) \qquad \frac{\Gamma\vdash A(x)|\Delta \qquad x\text{ fresh}}{\Gamma\vdash\forall xA(x)|\Delta}\ (\forall_R)$$

$$\frac{}{\Gamma|\bot\vdash\Delta}\ (\bot_L) \qquad \frac{}{\Gamma\vdash\top|\Delta}\ (\top_R)$$

$$\frac{\Gamma\vdash A|\Delta \qquad \Gamma|A\vdash\Delta}{\Gamma\vdash\Delta}\ (\mathrm{Cut})$$

Table 1: The sequent calculus $\mathrm{LK}_{\mu\tilde{\mu}}$

2.2. Sequent calculus $\mathbf{LK}_{\mu\tilde{\mu}}$

All the models presented so far (Sections 1.1, 1.3, 2.1) were built from components of a derivation system. We will use the sequent calculus $\mathrm{LK}_{\mu\tilde{\mu}}$of Curien and Herbelin [**53**] (Table 1) in the rest of this chapter. It is a variant of Gentzen's LK sequent calculus, with the following differences.

- Sequents come with an explicitly distinguished formula on the right or on the left, or no distinguished formula at all, resulting in three kinds of sequents: "$\Gamma\vdash\Delta$", "$\Gamma|A\vdash\Delta$" and "$\Gamma\vdash A|\Delta$". In particular, the distinguished formula plays an "active" rôle in the rules;
- Accordingly, the axiom rule splits into two variants, $(\mathrm{Ax}_L)$and (Ax_R), depending on whether the left active formula or the right active formula is distinguished. There are also two new rules, (μ) and $(\tilde{\mu})$, for making a formula active;
- There are no explicit contraction rules: contractions are derivable from a cut against an axiom as follows.
  - Left contraction:

$$\frac{\dfrac{}{\Gamma,A\vdash A\mid\Delta}\ (\mathrm{Ax}_R) \qquad \Gamma,A\mid A\vdash\Delta}{\Gamma,A\vdash\Delta}\ (\mathrm{Cut})$$

– Right contraction:

$$\dfrac{\dfrac{\Gamma \vdash A \mid A, \Delta \qquad \overline{\Gamma \mid A \vdash A, \Delta}\ (\mathrm{Ax}_L)}{\Gamma \vdash A, \Delta}\ (\mathrm{Cut})}$$

- Consequently, the notion of normal proof, or cut-freeness, is slightly different from the notion of cut-freeness in LK: a *normal proof* is a proof whose only cuts are of the form of a cut between an axiom and an introduction rule[1]. This is the notion that we refer to when below, very often, we say *cut-free* or *provable without a cut.*

Derivations in $\mathrm{LK}_{\mu\tilde{\mu}}$ can be written as proof terms, that is, $\mathrm{LK}_{\mu\tilde{\mu}}$ is a typing system for a calculus of proof terms, similar to λ-calculus, the $\bar{\lambda}\mu\tilde{\mu}$-calculus.

2.2.1. Definition. The proof "terms" of $\bar{\lambda}\mu\tilde{\mu}$ are defined by simultaneously defining three categories of expressions:

$$\begin{array}{rr}
c := \langle p \| e \rangle & \text{commands} \\
p, q := a \mid \lambda a.p \mid \iota_1 p \mid \iota_2 p \mid (p, q) \mid \lambda x.p \mid (t, p) \mid \mu\alpha.c \mid \mathsf{tt} & \text{terms} \\
e, f := \alpha \mid p \cdot e \mid [e, f] \mid \pi_1 e \mid \pi_2 e \mid t \cdot e \mid \lambda x.e \mid \tilde{\mu}x.c \mid \mathsf{ff} & \text{eval. contexts}
\end{array}$$

There are three kinds of variables, proof term variables $a, b, \ldots$, evaluation context variables $\alpha, \beta, \ldots$ and individual (quantifier) variables $x, y, \ldots$. We rely on these conventions to resolve the apparent ambiguity of the syntax: the abstraction $\lambda a.p$ is a proof term for implication, $\lambda x.p$ is a proof term for $\forall$, while $\lambda x.e$ is an evaluation context for $\exists$; also, the application $p \cdot e$ is an evaluation context for $\Rightarrow$, while $t \cdot e$ is an evaluation context for $\forall$; finally, (p, q) is a proof term for $\wedge$, while (t, q) is a proof term for $\exists$.

Properly speaking, proof terms are only those ones that can annotate valid derivations of $\mathrm{LK}_{\mu\tilde{\mu}}$ according to Table 2.

As any other λ-calculus, $\bar{\lambda}\mu\tilde{\mu}$ comes with a set of reduction rules that describe its computational behaviour. A difference with more conventional λ-calculi, but a similarity with such calculi for classical logic, is that the reduction is not defined on proof terms proper. Rather, reduction is defined on *commands*, which compound a proof term with an environment (evaluation context).

2.2.2. Definition. The reduction relation on $\bar{\lambda}\mu\tilde{\mu}$-commands is defined via the following rewrite rules:

$$\begin{array}{rr}
\langle \lambda a.p \| q \cdot e \rangle \to \langle q \| \tilde{\mu}a.\langle p \| e \rangle \rangle & (\to_{\Rightarrow}) \\
\langle (p_1, p_2) \| \pi_i e \rangle \to \langle p_i \| e \rangle & (\to_{\wedge}) \\
\langle \iota_i p \| [e_1, e_2] \rangle \to \langle p \| e_i \rangle & (\to_{\vee}) \\
\langle \lambda x.p \| t \cdot e \rangle \to \langle p\{t/x\} \| e \rangle & (\to_{\forall}) \\
\langle (t, p) \| \lambda x.e \rangle \to \langle p \| e\{t/x\} \rangle & (\to_{\exists}) \\
\langle \mu\alpha.c \| e \rangle \to c\{e/\alpha\} & (\to_{\mu}) \\
\langle p \| \tilde{\mu}a.c \rangle \to c\{p/a\} & (\to_{\tilde{\mu}})
\end{array}$$

There are no reduction rules for $\top$ and $\bot$.

[1] The rules (μ) and $(\tilde{\mu})$ are not introduction rules, because they do not construct a formula.

$$\frac{}{\Gamma|\alpha : A \vdash (\alpha : A), \Delta}\ (\mathrm{Ax}_L) \qquad \frac{}{(a : A), \Gamma \vdash a : A|\Delta}\ (\mathrm{Ax}_R)$$

$$\frac{c : (\Gamma, (a : A) \vdash \Delta)}{\Gamma|\tilde{\mu}a.c : A \vdash \Delta}\ (\tilde{\mu}) \qquad \frac{c : (\Gamma \vdash (\alpha : A), \Delta)}{\Gamma \vdash \mu\alpha.c : A|\Delta}\ (\mu)$$

$$\frac{\Gamma \vdash p : A|\Delta \qquad \Gamma|e : B \vdash \Delta}{\Gamma|p \cdot e : A \Rightarrow B \vdash \Delta}\ (\Rightarrow_L) \qquad \frac{\Gamma, (a : A) \vdash p : B|\Delta}{\Gamma \vdash \lambda a.p : A \Rightarrow B|\Delta}\ (\Rightarrow_R)$$

$$\frac{\Gamma|e : A \vdash \Delta \qquad \Gamma|f : B \vdash \Delta}{\Gamma|[e, f] : A \vee B \vdash \Delta}\ (\vee_L) \qquad \frac{\Gamma \vdash p : A_i|\Delta}{\Gamma \vdash \iota_i p : A_1 \vee A_2|\Delta}\ (\vee^i_R)$$

$$\frac{\Gamma|e : A_i \vdash \Delta}{\Gamma|\pi_i e : A_1 \wedge A_2 \vdash \Delta}\ (\wedge^i_L) \qquad \frac{\Gamma \vdash p : A|\Delta \qquad \Gamma \vdash q : B|\Delta}{\Gamma \vdash (p, q) : A \wedge B|\Delta}\ (\wedge_R)$$

$$\frac{\Gamma|e : A(x) \vdash \Delta \qquad x \text{ fresh}}{\Gamma|\lambda x.e : \exists x A(x) \vdash \Delta}\ (\exists_L) \qquad \frac{\Gamma \vdash p : A(t)|\Delta}{\Gamma \vdash (t, p) : \exists x A(x)|\Delta}\ (\exists_R)$$

$$\frac{\Gamma|e : A(t) \vdash \Delta}{\Gamma|t \cdot e : \forall x A(x) \vdash \Delta}\ (\forall_L) \qquad \frac{\Gamma \vdash p : A(x)|\Delta \qquad x \text{ fresh}}{\Gamma \vdash \lambda x.p : \forall x A(x)|\Delta}\ (\forall_R)$$

$$\frac{}{\Gamma|\mathsf{ff} : \bot \vdash \Delta}\ (\bot_L) \qquad \frac{}{\Gamma \vdash \mathsf{tt} : \top|\Delta}\ (\top_R)$$

$$\frac{\Gamma \vdash p : A|\Delta \qquad \Gamma|e : A \vdash \Delta}{\langle p \| e \rangle : (\Gamma \vdash \Delta)}\ (\mathrm{Cut})$$

Table 2: The sequent calculus $\mathrm{LK}_{\mu\tilde{\mu}}$ with proof terms

We will only need the reduction relation on $\bar{\lambda}\mu\tilde{\mu}$ in Section 2.5, when we discuss the computational content of our completeness proofs, because the point of normalisation-by-evaluation was precisely to bypass reasoning with a rewrite system.

In the following lemmas, we give some standard results about derivations in $\mathrm{LK}_{\mu\tilde{\mu}}$. All proofs are by a simple induction on the derivation, proving simultaneously the three clauses for commands, terms and evaluation contexts. Also, it will be important for the completeness theorems later, that the proofs do not introduce any new cuts.

2.2.3. Lemma. *The following hold for $LK_{\mu\tilde{\mu}}$:*

$$c : (\Gamma \vdash \Delta) \longrightarrow \textit{for all } (\Gamma', \Delta') \geq (\Gamma, \Delta),\ c : (\Gamma' \vdash \Delta')$$
$$\Gamma \vdash p : A|\Delta \longrightarrow \textit{for all } (\Gamma', \Delta') \geq (\Gamma, \Delta),\ \Gamma' \vdash p : A|\Delta'$$
$$\Gamma|e : A \vdash \Delta \longrightarrow \textit{for all } (\Gamma', \Delta') \geq (\Gamma, \Delta),\ \Gamma'|e : A \vdash \Delta'$$

2.2.4. Lemma. *The following hold for $LK_{\mu\tilde{\mu}}$, for any free variable x and any individual term t:*

$$c : (\Gamma \vdash \Delta) \longrightarrow c\{t/x\} : (\Gamma\{t/x\} \vdash \Delta\{t/x\})$$
$$\Gamma \vdash p : A | \Delta \longrightarrow \Gamma\{t/x\} \vdash p\{t/x\} : A\{t/x\} | \Delta\{t/x\}$$
$$\Gamma | e : A \vdash \Delta \longrightarrow \Gamma\{t/x\} | e\{t/x\} : A\{t/x\} \vdash \Delta\{t/x\}$$

2.2.5. Corollary. *The following hold for $LK_{\mu\tilde{\mu}}$, for any x that does not appear in Γ and Δ, and any individual term t:*

$$\Gamma \vdash p : A(x) | \Delta \longrightarrow \Gamma \vdash p\{t/x\} : A(t) | \Delta$$
$$\Gamma | e : A(x) \vdash \Delta \longrightarrow \Gamma | e\{t/x\} : A(t) \vdash \Delta$$

2.3. Kripke-style models, call-by-name variant

We will now define a notion of model, which is similar to the notion of intuitionistic Kripke model, but which we can show sound and complete for the $LK_{\mu\tilde{\mu}}$ sequent calculus. To account for classical logic, we modify the traditional notion of Kripke model in the following two ways.

(1) *Not taking the forcing relation as primitive.* We take as primitive the notion of "strong refutation", and define forcing in terms of it. The forcing definition we get in this way partially coincides with the traditional definition of forcing, as shown by Proposition 2.3.5.

(2) *Allowing certain nodes to validate absurdity.* We allow certain possible worlds to be marked as "fallible", or "exploding". This approach has been taken for Kripke models by Veldman [**175**], for Beth models by Friedman [**161**], and for Boolean models by Krivine (Section 1.1), and seems necessary in order to have a constructive proof of completeness, in the view of the meta-mathematical results of Gödel, Kreisel and McCarthy [**120, 136, 135, 137**], which preclude constructive proofs of completeness in case one wants to retain that absurdity must never be valid in a possible world.

2.3.1. Definition. A *classical Kripke model*, or *classical Kripke-style model*, is given by:

- a preorder $(K, \leq)$ of *possible worlds*;
- a unary relation on worlds $(-) \Vdash_\perp$ labelling a world as *exploding*;
- a binary relation $(-) : (-) \Vdash_s$ of *strong refutation* between worlds and atomic formulae, such that

$$\text{for all } w' \geq w, w : X \Vdash_s \rightarrow w' : X \Vdash_s,$$
$$w : \perp \Vdash_s \text{ is true,}$$
$$w : \top \Vdash_s \text{ iff } w \Vdash_\perp;$$

- and a domain of quantification $D(w)$ for each world w, such that

$$\text{for all } w' \geq w, D(w) \subseteq D(w').$$

The relation $(-) : (-) \Vdash_s$ of *strong refutation* is *extended from atomic to composite formulae* inductively and by simultaneously defining two new relations, forcing and (non-strong) refutation:

⋆ A formula A is *forced* in the world w (notation $w \Vdash A$) if any world $w' \geq w$, which strongly refutes A, is exploding;

- ⋆ A formula A is *refuted* in the world w (notation $w : A \Vdash$) if any world $w' \geq w$, which forces A, is exploding;
- $w : A \wedge B \Vdash_s$ if $w : A \Vdash$ or $w : B \Vdash$;
- $w : A \vee B \Vdash_s$ if $w : A \Vdash$ and $w : B \Vdash$;
- $w : A \Rightarrow B \Vdash_s$ if $w \Vdash A$ and $w : B \Vdash$;
- $w : \forall x.A(x) \Vdash_s$ if $w : A(t) \Vdash$ for some $t \in D(w)$;
- $w : \exists x.A(x) \Vdash_s$ if, for any $w' \geq w$ and $t \in D(w')$, $w : A(t) \Vdash$.

We have the following basic properties of the defined relations.

2.3.2. Lemma. *Strong refutation, forcing and refutation are monotone in any classical Kripke model.*

PROOF. Monotonicity of strong refutation is proved by induction on the complexity of the formula, while monotonicity of forcing and of non-strong refutation follows directly from their definitions. □

2.3.3. Lemma. *In all worlds w and for all formulae A, if $w : A \Vdash_s$, then $w : A \Vdash$.*

PROOF. Immediate, from the definition of refutation. □

We will write $w : \Gamma \Vdash_s$, $w \Vdash \Gamma$, and $w : \Gamma \Vdash$, to mean that all formulae of Γ are, respectively, strongly forced, refuted, and forced. We now have the following theorem that says that we can evaluate $\mathrm{LK}_{\mu\tilde{\mu}}$ derivations into inhabitants of classical Kripke models.

2.3.4. Theorem (Soundness)**.** *In any classical Kripke model the following hold:*

$$c : (\Gamma \vdash \Delta) \longrightarrow \textit{for any } w \textit{ such that } w \Vdash \Gamma \textit{ and } w : \Delta \Vdash, w \Vdash_\perp$$
$$\Gamma \vdash p : A | \Delta \longrightarrow \textit{for any } w \textit{ such that } w \Vdash \Gamma \textit{ and } w : \Delta \Vdash, w \Vdash A$$
$$\Gamma | e : A \vdash \Delta \longrightarrow \textit{for any } w \textit{ such that } w \Vdash \Gamma \textit{ and } w : \Delta \Vdash, w : A \Vdash$$

PROOF. One proves easily the three statements simultaneously, by induction on the derivation. □

It is natural to wonder about the relationship between the intuitionistic and the classical forcing relations. We characterise that relationship in the next two propositions.

2.3.5. Proposition. *The following hold in any world w of any classical Kripke model:*

(1) $$w \Vdash A \Rightarrow B \longleftrightarrow \textit{for all } w' \geq w, w' \Vdash A \Rightarrow w' \Vdash B$$
(2) $$w \Vdash \forall x.A(x) \longleftrightarrow \textit{for all } w' \geq w \textit{ and } t \in D(w'), w' \Vdash A(t)$$
(3) $$w \Vdash \perp \longleftrightarrow w \Vdash_\perp$$
(4) $$w \Vdash \top \longleftrightarrow \textit{true}$$
(5) $$w \Vdash A \wedge B \longleftrightarrow w \Vdash A \textit{ and } w \Vdash B$$
(6) $$w \Vdash A \vee B \longleftarrow w \Vdash A \textit{ or } w \Vdash B$$
(7) $$w \Vdash \exists x.A(x) \longleftarrow \textit{for some } t \in D(w), w \Vdash A(t)$$

PROOF. *(1)* Suppose $w \Vdash A \Rightarrow B$, $w' \geq w$ and $w' \Vdash A$. To show $w' \Vdash B$ we let $w'' \geq w'$ and $w'' : B \Vdash_s$ and have to show that w'' is exploding. Since then $w'' : A \Rightarrow B \Vdash_s$ holds by monotonicity and Lemma 2.3.3, the claim follows from

the definition of $w \Vdash A \Rightarrow B$. For the other direction, suppose a world $w' \geq w$ in which $A \Rightarrow B$ is strongly refuted, i.e. $w' \Vdash A$ and $w' : B \Vdash$, and we have to show w' is exploding. But, this is immediate, since B is also forced by hypothesis (the right-hand side of the equivalence).

(2) By definition, $w \Vdash \forall x.A(x)$ iff $\forall w' \geq w, (\exists s \in D(w').w' : A(s) \Vdash) \Rightarrow w' \Vdash_\perp$, which is equivalent to the right-hand side of the equivalence thanks to Lemma 2.3.3 and refutation being defined in terms of forcing. (We used quantifier symbols at meta-level.)

(5) Assume $w \Vdash A$, $w \Vdash B$, $w \leq w'$, and $w' : A \wedge B \Vdash_s$. Therefore we have $w' : A \Vdash$ or $w' : B \Vdash$. Each case leads to $w' \Vdash_\perp$ since $w' \Vdash A$, $w' \Vdash B$ with monotonicity.

The rest of the cases follow from the definitions and the monotonicity of "$\Vdash$" and $D(-)$. □

2.3.6. Remark. Note, however, that although the characterisations of our and intuitionistic forcing "match" on the fragment $\{\Rightarrow, \wedge, \forall, \top, \perp\}$, that does not mean that a formula in that fragment is forced in our sense if and only if it is forced in the intuitionistic sense. The law of Peirce $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ is one counterexample to that, it is classically but not intuitionistically forced; this is so because in our forcing, hidden under the surface, there lays a notion of refutation which can be used.

We now consider the following double-negation translation $(\cdot)^*$, which is the one of Gödel[**87, 161**], except that atomic formulae are not doubly negated:

$$\begin{aligned}
X^* &:= X \quad (X\text{-atomic}) \\
(A \wedge B)^* &:= A^* \wedge B^* \\
(A \to B)^* &:= A^* \to B^* \\
(\forall x.A)^* &:= \forall x.A^* \\
(A \vee B)^* &:= \neg(\neg A^* \wedge \neg B^*) \\
(\exists x.A)^* &:= \neg\forall x.\neg A^*
\end{aligned}$$

2.3.7. Proposition. *Every classical Kripke model $\mathscr{C} = (K, \leq, D, \Vdash_s, \Vdash_\perp)$ gives rise to an intuitionistic Kripke model with exploding worlds $\mathscr{I} = (K, \leq, D, \Vdash_i, \Vdash_\perp)$, that inherits all components of $\mathscr{C}$, except for $\Vdash_i$ which is defined for atomic formulae by non-strong forcing, i.e.*

$$w \Vdash_i X \textit{ iff } w \Vdash X$$

The translation $(\cdot)^*$ relates $\mathscr{C}$ and $\mathscr{I}$, that is, for any world w and any formula A, we have

$$w \Vdash_i A^* \textit{ iff } w \Vdash A.$$

PROOF. By induction on the complexity of A and by using Lemmas 2.3.5 and 2.3.3. We detail only the induction case for $\vee$, which is the most involved one:

$$\begin{array}{lcl}
 & w \Vdash_i (A \vee B)^* & \longleftrightarrow \\
 & w \Vdash_i \neg(\neg A^* \wedge \neg B^*) & \longleftrightarrow \\
(\forall w' \geq w) & [w' \Vdash_i \neg A^*, w' \Vdash_i \neg B^* \longrightarrow w' \Vdash_i \bot] & \longleftrightarrow \\
(\forall w' \geq w)[& (\forall w'' \geq w')[w'' \Vdash_i A^* \longrightarrow w'' \Vdash_i \bot], & \\
 & (\forall w'' \geq w')[w'' \Vdash_i B^* \longrightarrow w'' \Vdash_i \bot] & \\
 & \longrightarrow w' \Vdash_i \bot] & \longleftrightarrow \\
(\forall w' \geq w)[& (\forall w'' \geq w')[w'' \Vdash A \longrightarrow w'' \Vdash_\bot], & \\
 & (\forall w'' \geq w')[w'' \Vdash B \longrightarrow w'' \Vdash_\bot] & \\
 & \longrightarrow w' \Vdash_\bot] & \longleftrightarrow \\
(\forall w' \geq w) & [w' : A \Vdash, w' : B \Vdash \longrightarrow w' \Vdash_\bot] & \longleftrightarrow \\
(\forall w' \geq w) & [w' : A \vee B \Vdash_s \longrightarrow w' \Vdash_\bot] & \longleftrightarrow \\
 & w \Vdash A \vee B &
\end{array}$$

□

We now define a universal model $\mathscr{U}$ analogous to the one for intuitionistic completeness from Lemma 2.1.3.

2.3.8. Definition. The *Universal classical Kripke model* $\mathscr{U}$ is obtained by setting:

- K to be the set of contexts (Γ, Δ) of $\mathrm{LK}_{\mu\tilde{\mu}}$;
- $(\Gamma, \Delta) \leq (\Gamma', \Delta')$ iff both $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$;
- $(\Gamma, \Delta) : X \Vdash_s$ iff the sequent $\Gamma | X \vdash \Delta$ is provable without a cut[2] in $\mathrm{LK}_{\mu\tilde{\mu}}$;
- $(\Gamma, \Delta) \Vdash_\bot$ iff the sequent $\Gamma \vdash \Delta$ is provable without a cut in $\mathrm{LK}_{\mu\tilde{\mu}}$;
- for any w, $D(w)$ is the set of individuals of $\mathrm{LK}_{\mu\tilde{\mu}}$(that is, $D(-)$ is a constant function from worlds to sets of individuals).

$(-) : (-) \Vdash_s$ is monotone because of Lemma 2.2.3.

We now have the following theorem.

2.3.9. Theorem (Cut-Free Completeness for $\mathscr{U}$, simplified)**.** *For any closed formula A and closed contexts Γ and Δ, the following hold in $\mathscr{U}$:*

(1) $(\Gamma, \Delta) :\Vdash A \longrightarrow \{p \mid \Gamma \vdash p : A | \Delta\}$ ($\downarrow -$*term reify*)

(2) $(\Gamma, \Delta) : A \Vdash \longrightarrow \{e \mid \Gamma | e : A \vdash \Delta\}$ ($\Downarrow -$*eval. context reify*)

Moreover, the derivations constructed in (1) *and* (2) *are cut-free.*

The proof of this version of the theorem was given in our article [**109**], and it will be given in the form of a λ-term in Section 2.5. We will proceed now to prove a more complex version of it. The reason for doing that is that it is this more complex version the one that we formalised in Coq, and only for it can we guarantee the computational behaviour described in Section 2.5.

For the more complex version we proceed like in the proof of Lemma 2.1.3. We need two notions of "neutrality", by analogy to neutral terms of page 32.

[2]Recall that by "cut" we do not mean the simple application of the CUT rule, but what we explained on page 35.

2.3.10. Definition (NT(−))**.** A variable declaration $a : A$ is said to be *neutral with respect to provability* in the context (Γ, Δ) (notation $\mathrm{NT}(a : A, \Gamma, \Delta)$), if, for any evaluation context e and any $(\Gamma', \Delta') \geq (\Gamma, \Delta)$, we have that

$$\Gamma' | e : A \vdash \Delta' \longrightarrow \langle a \| e \rangle : (\Gamma' \vdash \Delta').$$

2.3.11. Definition (NE(−))**.** A variable declaration $\alpha : A$ is said to be *neutral with respect to refutability* in the context (Γ, Δ) (notation $\mathrm{NE}(\alpha : A, \Gamma, \Delta)$), if, for any proof term p and any $(\Gamma', \Delta') \geq (\Gamma, \Delta)$, we have that

$$\Gamma' \vdash p : A | \Delta' \longrightarrow \langle p \| \alpha \rangle : (\Gamma' \vdash \Delta').$$

We will omit the proof term annotations in order to decrease the level of detail, however we remark that the formalisation was carried out with proof term annotations.

2.3.12. Theorem (Cut-Free Completeness for $\mathscr{U}$)**.** *For any closed A, Γ and Δ, the following hold in $\mathscr{U}$:*

(1) $(\Gamma, \Delta) \Vdash A \longrightarrow \Gamma \vdash A | \Delta$ ($\downarrow$ *−term reify*)

(2) $\mathrm{NT}(A, \Gamma, \Delta) \longrightarrow (\Gamma, \Delta) \Vdash A$ ($\uparrow$ *−term reflect*)

(3) $(\Gamma, \Delta) : A \Vdash \longrightarrow \Gamma | A \vdash \Delta$ ($\Downarrow$ *−eval. context reify*)

(4) $\mathrm{NE}(A, \Gamma, \Delta) \longrightarrow (\Gamma, \Delta) : A \Vdash$ ($\Uparrow$ *−eval. context reflect*)

Moreover, the derivations on the right-hand side of (1) and (3) are cut-free.

PROOF. We proceed by simultaneously proving all four statements by induction on the complexity of A.

Base case. In the base case we have forcing and refutation on atomic formulae, which by definition reduce to strong refutation on atomic formulae, which by definition reduces just to statements about the deductions in $\mathrm{LK}_{\mu\tilde{\mu}}$

(1) Suppose that

(*) $$\forall (\Gamma', \Delta') \geq (\Gamma, \Delta).\ \Gamma' | X \vdash \Delta' \to \Gamma' \vdash \Delta'$$

Then:

$$\dfrac{\dfrac{\dfrac{}{\Gamma | X \vdash X, \Delta}\,(\mathrm{Ax}_L)}{\Gamma \vdash X, \Delta}\,(*)}{\Gamma \vdash X | \Delta}\,(\mu)$$

(2) The hypothesis is $\mathrm{NT}(X, \Gamma, \Delta)$. Given $(\Gamma', \Delta') \geq (\Gamma, \Delta)$ such that $\Gamma' | X \vdash \Delta'$, we have:

$$\dfrac{\Gamma' | X \vdash \Delta'}{\Gamma' \vdash \Delta'}\,\mathrm{NT}(X, \Gamma, \Delta)$$

(3) We have $(\Gamma, \Delta) : X \Vdash$, i.e.,

(*) $$\forall (\Gamma', \Delta') \geq (\Gamma, \Delta).\ \left\{ \forall (\Gamma'', \Delta'') \geq (\Gamma', \Delta').\ \Gamma'' | X \vdash \Delta'' \to \Gamma'' \vdash \Delta'' \right\} \to \Gamma' \vdash \Delta'$$

We can show $\Gamma | X \vdash \Delta$ by applying the $(\tilde{\mu})$-rule and (*), but we also have to show the sub-statement in curly brackets of (*):

$$\dfrac{\dfrac{\text{because } X \in (X, \Gamma) \subseteq \Gamma''}{\Gamma'' \vdash X | \Delta''}\,(\mathrm{Ax}_R) \qquad \Gamma'' | X \vdash \Delta''}{\Gamma'' \vdash \Delta''}\,(\mathrm{Cut})$$

(4) Suppose $\mathrm{NE}(X,\Gamma,\Delta)$ and suppose $(\Gamma',\Delta') \geq (\Gamma,\Delta)$ such that

(#) $$\forall(\Gamma'',\Delta'') \geq (\Gamma',\Delta').\ \Gamma''|X \vdash \Delta'' \to \Gamma'' \vdash \Delta''$$

Then:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma', X \vdash X|\Delta'}\,(\mathrm{Ax}_R)}{\Gamma', X \vdash \Delta'}\,\mathrm{NE}(X,\Gamma,\Delta)}{\Gamma'|X \vdash \Delta'}\,(\tilde{\mu})}{\Gamma' \vdash \Delta'}\,(\#)$$

Induction case for implication.

(1) We can strengthen the hypothesis $(\Gamma,\Delta) \Vdash A_1 \Rightarrow A_2$ using the induction hypotheses to obtain:

(#) $$\forall(\Gamma',\Delta') \geq (\Gamma,\Delta).\ \mathrm{NT}(A_1,\Gamma',\Delta') \to \mathrm{NE}(A_2,\Gamma',\Delta') \to \Gamma' \vdash \Delta'$$

Now we have:

$$\dfrac{\dfrac{\dfrac{}{A_1,\Gamma \vdash A_2,\Delta}\,(\#)}{A_1,\Gamma \vdash A_2|\Delta}\,(\mu)}{\Gamma \vdash A_1 \Rightarrow A_2|\Delta}\,(\Rightarrow_R)$$

And we have to show $\mathrm{NT}(A_1,(A_1,\Gamma),(A_2,\Delta))$ and $\mathrm{NE}(A_2,(A_1,\Gamma),(A_2,\Delta))$, which is easy using weakening because the neutral formulae already appear in the contexts.

(2) Suppose $\mathrm{NT}(A_1 \Rightarrow A_2,\Gamma,\Delta)$ and suppose $(\Gamma',\Delta') \geq (\Gamma,\Delta)$ such that $(\Gamma',\Delta') \Vdash A_1$ and $(\Gamma',\Delta') : A_2 \Vdash$. The induction hypotheses give us that $\Gamma' \vdash A_1|\Delta'$ and $\Gamma'|A_2 \vdash \Delta'$. Now we have:

$$\dfrac{\dfrac{\Gamma' \vdash A_1|\Delta' \qquad \Gamma'|A_2 \vdash \Delta'}{\Gamma'|A_1 \Rightarrow A_2 \vdash \Delta'}\,(\Rightarrow_L)}{\Gamma' \vdash \Delta'}\,\mathrm{NT}(A_1 \Rightarrow A_2,\Gamma,\Delta)$$

(3) We have $(\Gamma,\Delta) : A_1 \Rightarrow A_2 \Vdash$, i.e.,

(*) $$\forall(\Gamma',\Delta') \geq (\Gamma,\Delta).\ \{\forall(\Gamma'',\Delta'') \geq (\Gamma',\Delta').\ \mathrm{NT}(A_1,\Gamma'',\Delta'') \to \mathrm{NE}(A_2,\Gamma'',\Delta'') \to \Gamma'' \vdash \Delta''\} \to \Gamma' \vdash \Delta'$$

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash A_1,\Delta}\,(*)}{\Gamma \vdash A_1|\Delta}\,(\mu) \qquad \dfrac{\dfrac{}{\Gamma, A_2 \vdash \Delta}\,(*)}{\Gamma|A_2 \vdash \Delta}\,(\tilde{\mu})}{\Gamma|A_1 \Rightarrow A_2 \vdash \Delta}\,(\Rightarrow_L)$$

Due to the use of (*) we have to show the sub-expression in curly brackets. Let us show only one case, the other is symmetric:

$$\dfrac{\dfrac{}{\Gamma''|A_1 \vdash \Delta''}\,(\mathrm{Ax}_L),\ \text{since } (A_1,\Delta) \subseteq \Delta''}{\Gamma'' \vdash \Delta''}\,\mathrm{NT}(A_1,\Gamma,(A_1,\Delta))$$

(4) Let $\mathrm{NE}(A_1 \Rightarrow A_2,\Gamma,\Delta)$ and let $(\Gamma',\Delta') \geq (\Gamma,\Delta)$ be given such that:

(#) $$\forall(\Gamma'',\Delta'') \geq (\Gamma',\Delta').\ \mathrm{NT}(A_1,\Gamma'',\Delta'') \to \mathrm{NE}(A_2,\Gamma'',\Delta'') \to \Gamma'' \vdash \Delta''$$

We show $\Gamma' \vdash \Delta'$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A_1,\Gamma' \vdash A_2,\Delta'}\,(\#)}{A_1,\Gamma' \vdash A_2|\Delta'}\,(\mu)}{\Gamma' \vdash A_1 \Rightarrow A_2|\Delta'}\,(\Rightarrow_R)}{\Gamma' \vdash \Delta'}\,\mathrm{NE}(A_1 \Rightarrow A_2,\Gamma,\Delta)$$

For the application of (#) we have to show the corresponding $\mathrm{NT}(A_1,(A_1,\Gamma'),(A_2,\Delta'))$ and $\mathrm{NE}(A_2,(A_1,\Gamma'),(A_2,\Delta'))$, but this is easy since the formulae are already inside the corresponding contexts.

Induction case for $\vee$.

(1) Suppose $(\Gamma,\Delta)\Vdash A_1\vee A_2$, which can be strengthened using the induction hypotheses to:

(*) $$\forall(\Gamma',\Delta')\geq(\Gamma,\Delta).\ \mathrm{NE}(A_1,\Gamma',\Delta')\to\mathrm{NE}(A_2,\Gamma',\Delta')\to\Gamma'\vdash\Delta'$$

Here is a derivation of $\Gamma\vdash A_1\vee A_2|\Delta$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma\vdash A_2,A_1,A_1\vee A_2,\Delta}\,(*)}{\Gamma\vdash A_2|A_1,A_1\vee A_2,\Delta}\,(\mu)}{\Gamma\vdash A_1\vee A_2|A_1,A_1\vee A_2,\Delta}\,(\vee_L^2)\qquad\dfrac{}{\Gamma|A_1\vee A_2\vdash A_1,A_1\vee A_2,\Delta}\,(\mathrm{Ax}_L)}{\dfrac{\dfrac{\dfrac{\Gamma\vdash A_1,A_1\vee A_2,\Delta}{\Gamma\vdash A_1|A_1\vee A_2,\Delta}\,(\mu)}{\Gamma\vdash A_1\vee A_2|A_1\vee A_2,\Delta}\,(\vee_L^1)\qquad\dfrac{}{\Gamma|A_1\vee A_2\vdash A_1\vee A_2,\Delta}\,(\mathrm{Ax}_L)}{\Gamma\vdash A_1\vee A_2,\Delta}\,(\mathrm{Cut})}\,(\mu)}{\Gamma\vdash A_1\vee A_2|\Delta}\,(\mu)$$

It is only left to prove that $\mathrm{NE}(A_1,\Gamma,(A_2,A_1,A_1\vee A_2,\Delta))$ and $\mathrm{NE}(A_2,\Gamma,(A_2,A_1,A_1\vee A_2,\Delta))$, but that is trivial because the neutral formulae are already in the context.

(2) Let $\mathrm{NT}(A_1\vee A_2,\Gamma,\Delta)$ and suppose given a $(\Gamma',\Delta')\geq(\Gamma,\Delta)$ such that (by induction hypotheses) $\mathrm{NE}(A_1,\Gamma',\Delta')$ and $\mathrm{NE}(A_2,\Gamma',\Delta')$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma',A_1\vdash A_1|\Delta'}\,(\mathrm{Ax}_R)}{\Gamma',A_1\vdash\Delta'}\,\mathrm{NE}(A_1,\Gamma',\Delta')}{\Gamma'|A_1\vdash\Delta'}\,(\tilde{\mu})\qquad\dfrac{\dfrac{\dfrac{}{\Gamma',A_2\vdash A_2|\Delta'}\,(\mathrm{Ax}_R)}{\Gamma',A_2\vdash\Delta'}\,\mathrm{NE}(A_2,\Gamma',\Delta')}{\Gamma'|A_2\vdash\Delta'}\,(\tilde{\mu})}{\Gamma'|A_1\vee A_2\vdash\Delta'}\,(\vee_L)}{\Gamma'\vdash\Delta'}\,\mathrm{NT}(A_1\vee A_2,\Gamma,\Delta)$$

(3) Using the induction hypotheses we get from $(\Gamma,\Delta):A_1\vee A_2\Vdash$:

(*) $$\begin{aligned}&\forall(\Gamma',\Delta')\geq(\Gamma,\Delta).\ \{\forall(\Gamma'',\Delta'')\geq(\Gamma',\Delta').\ \mathrm{NE}(A_1,\Gamma'',\Delta'')\to\\&\quad\mathrm{NE}(A_2,\Gamma'',\Delta'')\to\Gamma''\vdash\Delta''\}\to\Gamma'\vdash\Delta'\end{aligned}$$

We can have the following derivation

$$\dfrac{\dfrac{\dfrac{}{\Gamma,A_1\vdash\Delta}\,(*)}{\Gamma|A_1\vdash\Delta}\,(\tilde{\mu})\qquad\dfrac{\dfrac{}{\Gamma,A_2\vdash\Delta}\,(*)}{\Gamma|A_2\vdash\Delta}\,(\tilde{\mu})}{\Gamma|A_1\vee A_2\vdash\Delta}\,(\vee_L)$$

but, we have to prove that the sub-statement in curly brackets from (*) holds for both the context A_1,Γ and the context A_2,Γ. Here is one of them: (the other is analogous)

$$\dfrac{\dfrac{}{\Gamma''\vdash A_1|\Delta''}\,(\mathrm{Ax}_R),\ \text{since}\ (A_1,\Gamma)\subseteq\Gamma''}{\Gamma''\vdash\Delta''}\,\mathrm{NE}(A_1,\Gamma'',\Delta'')$$

(4) Suppose $\mathrm{NE}(A1\vee A_2,\Gamma,\Delta),(\Gamma',\Delta')\geq(\Gamma,\Delta)$ and, using the induction hypothesis, suppose:

(*) $$\forall(\Gamma'',\Delta'')\geq(\Gamma',\Delta').\ \mathrm{NE}(A_1,\Gamma'',\Delta'')\to\mathrm{NE}(A_2,\Gamma'',\Delta'')\to\Gamma''\vdash\Delta''$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash A_2, A_1, \Delta'}(*)}{\Gamma' \vdash A_2 | A_1, \Delta'}(\mu)}{\Gamma' \vdash A_1 \vee A_2 | A_1, \Delta'}(\vee^2_L)}{\Gamma' \vdash A_1, \Delta'}\mathrm{NE}(A_1 \vee A_2, \Gamma, \Delta)}{\Gamma' \vdash A_1 | \Delta'}(\mu)}{\Gamma' \vdash A_1 \vee A_2 | \Delta'}(\vee^1_L)}{\Gamma' \vdash \Delta'}\mathrm{NE}(A_1 \vee A_2, \Gamma, \Delta)$$

This constitutes a derivation as required, given that it is easy to prove $\mathrm{NE}(A_1, \Gamma', (A_2, A_1, \Delta'))$ and $\mathrm{NE}(A_2, \Gamma', (A_2, A_1, \Delta'))$ which arise from the use of (*).

Induction case for $\wedge$.

(1) Let

(*) $$\forall (\Gamma', \Delta') \geq (\Gamma, \Delta).\ \mathrm{NE}(A_1, \Gamma', \Delta') \text{ or } \mathrm{NE}(A_2, \Gamma', \Delta') \rightarrow \Gamma' \vdash \Delta'$$

Here is the required derivation:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash A_1, \Delta}(*)}{\Gamma \vdash A_1 | \Delta}(\mu) \quad \dfrac{\dfrac{}{\Gamma \vdash A_2, \Delta}(*)}{\Gamma \vdash A_2 | \Delta}(\mu)}{\Gamma \vdash A_1 \wedge A_2 | \Delta}(\wedge_R)$$

Where it is easy to show that A_1 and A_2 are neutral in the two cases arising from the use of (*).

(2) Suppose $\mathrm{NT}(A_1 \wedge A_2, \Gamma, \Delta)$. To show $(\Gamma, \Delta) \Vdash A_1 \wedge A_2$, let $(\Gamma', \Delta') \geq (\Gamma, \Delta)$ and ($\mathrm{NE}(A_1, \Gamma', \Delta')$ or $\mathrm{NE}(A_2, \Gamma', \Delta')$) be true. Without loss of generality, let $\mathrm{NE}(A_1, \Gamma', \Delta')$ be true. Then:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma', A_1 \vdash A_1 | \Delta'}(\mathrm{Ax}_R)}{\Gamma', A_1 \vdash \Delta'}\mathrm{NE}(A_1, \Gamma', \Delta')}{\Gamma' | A_1 \vdash \Delta'}(\tilde{\mu})}{\Gamma' | A_1 \wedge A_2 \vdash \Delta'}(\wedge^1_L)}{\Gamma' \vdash \Delta'}\mathrm{NT}(A_1 \wedge A_2, \Gamma', \Delta')$$

(3) Suppose $(\Gamma, \Delta) : A_1 \wedge A_2 \Vdash$, which can be strengthened using the induction hypotheses to:

(*) $$\forall (\Gamma', \Delta') \geq (\Gamma, \Delta).\ \{\forall (\Gamma'', \Delta'') \geq (\Gamma', \Delta').\ \mathrm{NE}(A_1, \Gamma'', \Delta'') \text{ or } \mathrm{NE}(A_2, \Gamma'', \Delta'') \rightarrow \Gamma'' \vdash \Delta''\} \rightarrow \Gamma' \vdash \Delta'$$

Now we have:

$$\dfrac{\dfrac{}{\Gamma, A_1 \wedge A_2 \vdash \Delta}(*)}{\Gamma | A_1 \wedge A_2 \vdash \Delta}(\tilde{\mu})$$

But we have to show the hypothesis of (*). Let $(\Gamma'', \Delta'') \geq ((A_1 \wedge A_2, \Gamma), \Delta)$ and suppose, without loss of generality, that the left disjunct is true, i.e., we have $\mathrm{NE}(A_1, \Gamma'', \Delta'')$. Then:

$$\dfrac{\dfrac{}{\Gamma'' \vdash A_1 \wedge A_2 | \Delta''}(\mathrm{Ax}_R) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma'', A_1 \vdash A_1 | \Delta''}(\mathrm{Ax}_R)}{\Gamma'', A_1 \vdash \Delta''}\mathrm{NE}(A_1, \Gamma'', \Delta'')}{\Gamma'' | A_1 \vdash \Delta''}(\tilde{\mu})}{\Gamma'' | A_1 \wedge A_2 \vdash \Delta''}(\wedge^1_L)}{\Gamma'' \vdash \Delta''}(\mathrm{Cut})$$

(4) Suppose $\mathrm{NE}(A_1 \wedge A_2, \Gamma, \Delta)$, $(\Gamma', \Delta') \geq (\Gamma, \Delta)$ and, using the induction hypotheses, suppose:

(#) $\quad \forall (\Gamma'', \Delta'') \geq (\Gamma', \Delta').\ \mathrm{NE}(A_1, \Gamma'', \Delta'')$ or $\mathrm{NE}(A_2, \Gamma'', \Delta'') \to \Gamma'' \vdash \Delta''$

We have:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash A_1, \Delta'}\,(\#)}{\Gamma' \vdash A_1 | \Delta'}\,(\mu) \quad \dfrac{\dfrac{}{\Gamma' \vdash A_2, \Delta'}\,(\#)}{\Gamma' \vdash A_2 | \Delta'}\,(\mu)}{\Gamma' \vdash A_1 \wedge A_2 | \Delta'}\,(\wedge_R)}{\Gamma' \vdash \Delta'}\,\mathrm{NE}(A_1 \wedge A_2, \Gamma, \Delta)$$

where $\mathrm{NE}(A_1, \Gamma', (A_1, \Delta'))$ and $\mathrm{NE}(A_2, \Gamma', (A_2, \Delta'))$ arising from the use of (#) are easy to prove, because the formulae are already inside the contexts.

Induction case for $\forall$. In the induction cases for $\forall$ and $\exists$ we leave out the membership in the domain of individuals $D(-)$ since in $\mathscr{U}$ we have a constant domain and we use the quantifier symbols also at the meta-level, to shorten the notation.

(1) Let $(\Gamma, \Delta) \Vdash \forall x.A(x)$. Using the induction hypotheses we get:

(*) $\quad \forall (\Gamma', \Delta') \geq (\Gamma, \Delta).\ (\exists t.\ \mathrm{NE}(A(t), \Gamma', \Delta')) \to \Gamma' \vdash \Delta'$

Here follows the required derivation:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash A(x), \Delta}\,(*)}{\Gamma \vdash A(x) | \Delta}\,(\mu)}{\Gamma \vdash \forall x.A(x) | \Delta}\,(\forall_R),\ x\text{-fresh}$$

One easily shows that $\mathrm{NE}(A(x), \Gamma, (A(x), \Delta))$.

(2) Suppose $\mathrm{NT}(\forall x.A(x), \Gamma, \Delta)$, $(\Gamma', \Delta') \geq (\Gamma, \Delta)$ and we have t such that, by induction hypothesis, $\Gamma' | A(t) \vdash \Delta'$. Here is a derivation of $\Gamma' \vdash \Delta'$:

$$\dfrac{\dfrac{\Gamma' | A(t) \vdash \Delta'}{\Gamma' | \forall x.A(x) \vdash \Delta'}\,(\forall_L)}{\Gamma' \vdash \Delta'}\,\mathrm{NT}(\forall x.A(x), \Gamma, \Delta)$$

(3) Suppose $(\Gamma, \Delta) : \forall x.A(x) \Vdash$. We strengthen this using the induction hypothesis to:

(*)
$$\begin{array}{l} \forall (\Gamma', \Delta') \geq (\Gamma, \Delta).\ \{\forall (\Gamma'', \Delta'') \geq (\Gamma', \Delta').\\ \qquad (\exists t.\ \mathrm{NE}(A(t), \Gamma'', \Delta'')) \to \Gamma'' \vdash \Delta''\} \to \Gamma' \vdash \Delta' \end{array}$$

This

$$\dfrac{\dfrac{}{\Gamma, \forall x.A(x) \vdash \Delta}\,(*)}{\Gamma | \forall x.A(x) \vdash \Delta}\,(\tilde{\mu})$$

is the derivation we need, in case we prove the sub-expression in curly brackets of (*). Therefore, suppose $(\Gamma'', \Delta'') \geq ((\forall x.A(x), \Gamma), \Delta)$ and suppose a t with $\mathrm{NE}(A(t), \Gamma'', \Delta'')$. We have to prove $\Gamma'' \vdash \Delta''$:

$$\dfrac{\dfrac{}{\Gamma'' \vdash \forall x.A(x) | \Delta''}\,(\mathrm{Ax}_R) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma'', A(t) \vdash A(t) | \Delta''}\,(\mathrm{Ax}_R)}{\Gamma'', A(t) \vdash \Delta''}\,\mathrm{NE}(A(t), \Gamma'', \Delta'')}{\Gamma'' | A(t) \vdash \Delta''}\,(\tilde{\mu})}{\Gamma'' | \forall x.A(x) \vdash \Delta''}\,(\forall_L)}{\Gamma'' \vdash \Delta''}\,(\mathrm{Cut})$$

(4) Suppose $\mathrm{NE}(\forall x.A(x),\Gamma,\Delta)$. To show $(\Gamma,\Delta):\forall x.A(x)\Vdash$, let $(\Gamma',\Delta')\geq(\Gamma,\Delta)$ and let

$$(*)\qquad \forall(\Gamma'',\Delta'')\geq(\Gamma',\Delta').\ \big(\exists t.\ \mathrm{NE}(A(t),\Gamma'',\Delta'')\big)\to\Gamma''\vdash\Delta''$$

Here is the required derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma'\vdash A(x),\Delta'}\,(*)}{\Gamma'\vdash A(x)|\Delta'}\,(\mu)}{\Gamma'\vdash\forall x.A(x)|\Delta'}\,(\forall_R),\ x\text{-fresh}}{\Gamma'\vdash\Delta'}\,\mathrm{NE}(\forall x.A(x),\Gamma,\Delta)$$

For the application of (*) one can easily show that $\mathrm{NE}(A(x),\Gamma',(A(x),\Delta'))$.

Induction case for $\exists$.

(1) Suppose $(\Gamma,\Delta)\Vdash\exists x.A(x)$, which using the induction hypothesis can be strengthened to:

$$(*)\qquad \forall(\Gamma',\Delta')\geq(\Gamma,\Delta).\ (\forall t.\ \mathrm{NE}(A(t),\Gamma',\Delta'))\Rightarrow\Gamma'\vdash\Delta'$$

The following is a good derivation

$$\dfrac{\dfrac{}{\Gamma\vdash\exists x.A(x),\Delta}\,(*)}{\Gamma\vdash\exists x.A(x)|\Delta}\,(\mu)$$

if we manage to show the hypothesis from applying (*). For that, let $t,\Gamma'\supseteq\Gamma,\Delta'\supseteq(\exists x.A(x),\Delta)$ be given such that $\Gamma'\vdash A(t)|\Delta'$.

$$\dfrac{\dfrac{\Gamma'\vdash A(t)|\Delta'}{\Gamma'\vdash\exists x.A(x)|\Delta'}\,(\exists_R)\qquad\dfrac{}{\Gamma'|\exists x.A(x)\vdash\Delta'}\,(\mathrm{Ax}_L)}{\Gamma'\vdash\Delta'}\,(\mathrm{Cut})$$

(2) Let $\mathrm{NT}(\exists x.A(x),\Gamma,\Delta)$ and $(\Gamma',\Delta')\geq(\Gamma,\Delta)$ be given and suppose

$$(\#)\qquad \forall t.\ \Gamma'|A(t)\vdash\Delta'$$

The required derivation is:

$$\dfrac{\dfrac{\dfrac{}{\Gamma'|A(x)\vdash\Delta'}\,(\#)}{\Gamma'|\exists x.A(x)\vdash\Delta'}\,(\exists_L),\ x\text{-fresh}}{\Gamma'\vdash\Delta'}\,\mathrm{NT}(\exists x.A(x),\Gamma,\Delta)$$

(3) Suppose $(\Gamma,\Delta):\exists x.A(x)\Vdash$ which gives, thanks to the induction hypothesis:

$$(*)\qquad \begin{array}{l}\forall(\Gamma',\Delta')\geq(\Gamma,\Delta).\ \{\forall(\Gamma'',\Delta'')\geq(\Gamma',\Delta').\\ \qquad(\forall t.\ \mathrm{NE}(A(t),\Gamma'',\Delta''))\Rightarrow\Gamma''\vdash\Delta''\}\Rightarrow\Gamma'\vdash\Delta'\end{array}$$

The required derivation is

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma,A(x)\vdash\Delta}\,(*)}{\Gamma|A(x)\vdash\Delta}\,(\tilde{\mu})}{\Gamma|\exists x.A(x)\vdash\Delta}\,(\exists_L),\ x\text{-fresh}}$$

but we also have to show the statement in curly brackets arising from the use of (*). Therefore, suppose $(\Gamma'',\Delta'')\geq((A(x),\Gamma),\Delta)$ and suppose, using the induction hypothesis, that $\forall t.\,\Gamma''|A(t)\vdash\Delta''$. We have to prove $\Gamma''\vdash\Delta''$:

$$\dfrac{\dfrac{}{\Gamma''\vdash A(x)|\Delta''}\,(\mathrm{Ax}_R)\qquad\dfrac{}{\Gamma''|A(x)\vdash\Delta''}\,(\#)}{\Gamma''\vdash\Delta''}\,(\mathrm{Cut})$$

(4) Suppose $\mathrm{NE}(\exists x.A(x),\Gamma,\Delta)$ and let $(\Gamma',\Delta') \geq (\Gamma,\Delta)$ such that $(\Gamma',\Delta') \Vdash \exists x.A(x)$. Using the induction hypothesis, this last thing strengthens to:

(*) $$\forall (\Gamma'',\Delta'') \geq (\Gamma',\Delta').\ (\forall t.\ \mathrm{NE}(A(t),\Gamma'',\Delta'')) \Rightarrow \Gamma'' \vdash \Delta''$$

To show $\Gamma' \vdash \Delta'$, we immediately apply (*) and then have to show the hypothesis: let t,Γ_3,Δ_3 be such that $\Gamma_3 \vdash A(t)|\Delta_3$ and $(\Gamma_3,\Delta_3) \geq (\Gamma'',\Delta'')$. Then this is what we are looking for:

$$\dfrac{\dfrac{\Gamma_3 \vdash A(t)|\Delta_3}{\Gamma_3 \vdash \exists x.A(x)|\Delta_3}\,(\exists_R)}{\Gamma_3 \vdash \Delta_3}\,\mathrm{NE}(\exists x.A(x),\Gamma,\Delta)$$

Induction case for $\top$.

(1) Immediate, from $(\top_R)$.
(2) Easy, using Proposition 2.3.5.
(3) Easy, using Proposition 2.3.5.
(4) Easy, a (Cut) with $\top$ and then $(\top_R)$ and (Ax_L).

Induction case for $\bot$.

(1) Easy, using Proposition 2.3.5.
(2) Easy, a (Cut) with $\bot$ and then $(\bot_L)$ and (Ax_R).
(3) Immediate, from $(\bot_L)$.
(4) Immediate, by applying the hypothesis.

All the given derivations are cut-free. By inspection of the proof trees, having in mind that NT, NE and weakening do not introduce new cuts, we convince ourselves that the completeness theorem indeed produces only cut-free derivations. □

2.3.13. Corollary. *For any* A,Γ *and* Δ*, the following hold in* $\mathscr{U}$*:*

(1) *If* $A \in \Gamma$ *then* $(\Gamma,\Delta) \Vdash A$.
(2) *If* $B \in \Delta$ *then* $(\Gamma,\Delta) : B \Vdash$.

PROOF. (1) follows from (2) of Theorem 2.3.12, and (2) follows from (4) of Theorem 2.3.12. □

2.3.14. Corollary (Completeness of Classical Logic)**.** *If in every classical Kripke model, at every possible world, the formula* A *is forced whenever all the formulae of* Γ *are forced and all the formulae of* Δ *are refuted, then there exists a derivation in* $\mathrm{LK}_{\mu\tilde{\mu}}$ *of the sequent* $\Gamma \vdash A|\Delta$.

PROOF. If the hypothesis holds for any classical Kripke model, so does it hold for $\mathscr{U}$. To show $\Gamma \vdash A|\Delta$, by Theorem 2.3.12, it is enough to show that $(\Gamma,\Delta) \Vdash B$ for all $B \in \Gamma$ and $(\Gamma,\Delta) : C \Vdash$ for all $C \in \Delta$, something shown in Corollary 2.3.13. □

A constructive cut-free completeness theorem can also be used for proof normalisation.

2.3.15. Corollary (Semantic Cut-Elimination)**.** *For all closed contexts* Γ,Δ*, if there is a derivation of* $\Gamma \vdash \Delta$*, then there is a cut-free derivation of* $\Gamma \vdash \Delta$.

PROOF. From the hypothesis $\Gamma \vdash \Delta$, the soundness theorem applied to $\mathscr{U}$ gives us that there is indeed a cut-free derivation for $\Gamma \vdash \Delta$ because the world

(Γ, Δ) forces all formulae of Γ and refutes all formulae of Δ as shown in Corollary 2.3.13. Notice that the composition of soundness and completeness, that is characteristic of NBE, takes place via Corollary 2.3.13. □

2.4. Kripke-style models, call-by-value variant

In this section we give an alternative version of classical Kripke model, in which we again do not take forcing as primitive, but we now have a primitive relation of *strong forcing* from which refutation and (non-strong) forcing are defined.

2.4.1. Definition. A *classical Kripke model*, or *classical Kripke-style model*, is given by:

- a preorder $(K, \leq)$ of *possible worlds*;
- a unary relation on worlds $(-) \Vdash_\bot$ labelling a world as *exploding*;
- a binary relation $(-) \Vdash_s (-)$ of *strong forcing* between worlds and atomic formulae, such that

$$\text{for all } w' \geq w, w \Vdash_s X \to w' \Vdash_s X,$$

$$w \Vdash_s \top \text{ is true,}$$

$$w \Vdash_s \bot \text{ iff } w \Vdash_\bot;$$

- and a domain of quantification $D(w)$ for each world w, such that

$$\text{for all } w' \geq w, D(w) \subseteq D(w').$$

The relation $(-) \Vdash_s (-)$ of *strong forcing* is *extended from atomic to composite formulae* inductively and by simultaneously defining two new relations, refutation and (non-strong) forcing:

- ⋆ A formula A is *refuted* in the world w (notation $w : A \Vdash$) if any world $w' \geq w$, which strongly forces A, is exploding;
- ⋆ A formula A is *forced* in the world w (notation $w \Vdash A$) if any world $w' \geq w$, which refutes A, is exploding;
- $w \Vdash_s A \wedge B$ if $w \Vdash A$ and $w \Vdash B$;
- $w \Vdash_s A \vee B$ if $w \Vdash A$ or $w \Vdash B$;
- $w \Vdash_s A \Rightarrow B$ if for all $w' \geq w$, $w \Vdash A$ implies $w \Vdash B$;
- $w \Vdash_s \forall x.A(x)$ if for all $w' \geq w$ and all $t \in D(w')$, $w' \Vdash A(t)$;
- $w \Vdash_s \exists x.A(x)$ if $w \Vdash A(t)$ for some $t \in D(w)$.

We have the following basic properties, which are proved analogously to the ones of Section 2.3.

2.4.2. Lemma. *Strong forcing, refutation and forcing are monotone in any classical Kripke model.*

2.4.3. Lemma. *In all worlds w and for all formulae A, if $w \Vdash_s A$, then $w \Vdash A$.*

2.4.4. Theorem (Soundness)**.** *In any classical Kripke model the following hold:*

$$c : (\Gamma \vdash \Delta) \longrightarrow \textit{for any } w \textit{ such that } w \Vdash \Gamma \textit{ and } w : \Delta \Vdash, w \Vdash_\bot$$
$$\Gamma \vdash p : A | \Delta \longrightarrow \textit{for any } w \textit{ such that } w \Vdash \Gamma \textit{ and } w : \Delta \Vdash, w \Vdash A$$
$$\Gamma | e : A \vdash \Delta \longrightarrow \textit{for any } w \textit{ such that } w \Vdash \Gamma \textit{ and } w : \Delta \Vdash, w : A \Vdash$$

The characterisation of forcing from Proposition 2.3.5 remains the same, but its proof is different (although analogous). Also, as in Proposition 2.3.7, we can construct an intuitionistic Kripke model starting from a classical one, but the double-negation translation is the one of Kolmogorov.

2.4.5. Proposition. *The following hold in any world w of any classical Kripke model:*

$$\begin{aligned}
w \Vdash A \Rightarrow B &\longleftrightarrow \textit{for all } w' \geq w, w' \Vdash A \Rightarrow w' \Vdash B \\
w \Vdash \forall x.A(x) &\longleftrightarrow \textit{for all } w' \geq w \textit{ and } t \in D(w'), w' \Vdash A(t) \\
w \Vdash \bot &\longleftrightarrow w \Vdash_\bot \\
w \Vdash \top &\longleftrightarrow \textit{true} \\
w \Vdash A \wedge B &\longleftrightarrow w \Vdash A \textit{ and } w \Vdash B \\
w \Vdash A \vee B &\longleftarrow w \Vdash A \textit{ or } w \Vdash B \\
w \Vdash \exists x.A(x) &\longleftarrow \textit{for some } t \in D(w), w \Vdash A(t)
\end{aligned}$$

PROOF. Direction "$\leftarrow$" of all statements is by Lemma 2.4.3. Direction "$\rightarrow$" we show for $\Rightarrow$ and $\wedge$, the rest of the cases are either analogous, or follow by definition and monotonicity.

Let $w \Vdash A \Rightarrow B$ and $w' \geq w$ with $w' \Vdash A$ be given. Let also $w' : B \Vdash$. To show $w' \Vdash_\bot$ we use the first hypothesis, and then given $w'' \geq w'$ and $w'' \Vdash_s A \Rightarrow B$ we have to show $w'' \Vdash_\bot$, but that follows directly from the rest of the hypotheses and monotonicity.

Let $w \Vdash A \wedge B$. We show only $w \Vdash A$, because the proof of $w \Vdash B$ is analogous. Let $w' \geq w$ be such that $w' : A \Vdash$. After applying the first hypothesis, we are given $w'' \geq w'$ with $w'' \Vdash_s A \wedge B$, that is, $w'' \Vdash A$ and $w'' \Vdash B$, and have to show $w'' \Vdash_\bot$, but this is immediate from the second hypothesis and monotonicity. □

This time, we consider the double-negation translation $(\cdot)^{\mathrm{k}}$, which is the one of Kolmogorov[**115, 161**], except that atomic formulae are not doubly negated:

$$\begin{aligned}
X^{\mathrm{k}} &:= X \quad (X\text{-atomic}) \\
(A \wedge B)^{\mathrm{k}} &:= \neg\neg(A^{\mathrm{k}} \wedge B^{\mathrm{k}}) \\
(A \to B)^{\mathrm{k}} &:= \neg\neg(A^{\mathrm{k}} \to B^{\mathrm{k}}) \\
(\forall x.A)^{\mathrm{k}} &:= \neg\neg(\forall x.A^{\mathrm{k}}) \\
(A \vee B)^{\mathrm{k}} &:= \neg\neg(A^{\mathrm{k}} \vee B^{\mathrm{k}}) \\
(\exists x.A)^{\mathrm{k}} &:= \neg\neg(\exists x.A^{\mathrm{k}})
\end{aligned}$$

2.4.6. Proposition. *Every classical Kripke model $\mathscr{C} = (K, \leq, D, \Vdash_s, \Vdash_\bot)$ gives rise to an intuitionistic Kripke model with exploding worlds $\mathscr{I} = (K, \leq, D, \Vdash_i, \Vdash_\bot)$, that inherits all components of $\mathscr{C}$, except for $\Vdash_i$ which is defined for atomic formulae by non-strong forcing, i.e.*

$$w \Vdash_i X \textit{ iff } w \Vdash X$$

The translation $(\cdot)^k$ relates $\mathscr{C}$ and $\mathscr{I}$, that is, for any world w and any formula A, we have

$$w \Vdash_i A^k \textit{ iff } w \Vdash A.$$

PROOF. By induction on the complexity of A. This time the proof is direct, because Kolmogorov's translation closely matches our definition of non-strong forcing. □

2.4.7. Definition. The *Universal classical Kripke model* $\mathscr{U}$ is obtained by setting:

- K to be the set of contexts (Γ, Δ) of $\mathrm{LK}_{\mu\tilde{\mu}}$;
- $(\Gamma, \Delta) \leq (\Gamma', \Delta')$ iff both $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$;
- $(\Gamma, \Delta) \Vdash_s X$ iff the sequent $\Gamma \vdash X | \Delta$ is provable without a cut in $\mathrm{LK}_{\mu\tilde{\mu}}$;
- $(\Gamma, \Delta) \Vdash_\bot$ iff the sequent $\Gamma \vdash \Delta$ is provable without a cut in $\mathrm{LK}_{\mu\tilde{\mu}}$;
- for any w, $D(w)$ is the set of individuals of $\mathrm{LK}_{\mu\tilde{\mu}}$(that is, $D(-)$ is a constant function from worlds to sets of individuals).

$(-) \Vdash_s (-)$ is monotone because of Lemma 2.2.3.

This time we will not prove a more complex version of the theorem, because it is the simpler version that we formalised in Coq.

2.4.8. Theorem (Cut-Free Completeness for $\mathscr{U}$)**.** *For any closed formula A and closed contexts Γ and Δ, the following hold in $\mathscr{U}$:*

(1) $\quad (\Gamma, \Delta) \Vdash A \longrightarrow \{p \mid \Gamma \vdash p : A | \Delta\}$ $\qquad$ ($\downarrow$ *−term reify*)

(2) $\quad (\Gamma, \Delta) : A \Vdash \longrightarrow \{e \mid \Gamma | e : A \vdash \Delta\}$ $\qquad$ ($\Downarrow$ *−eval. context reify*)

Moreover, the derivations constructed in (1) *and* (2) *are cut-free.*

PROOF. We will once again skip writing the proof term annotations in order to decrease the level of detail. The algorithm behind this proof that concentrates on proof terms is given in Table 7, page 55.

Base case. Let $(\Gamma, \Delta) \Vdash A$ for A-atomic. By definition, this is

$$(*) \quad \forall \left(\Gamma', \Delta'\right) \geq (\Gamma, \Delta) . \left(\forall \left(\Gamma'', \Delta''\right) \geq \left(\Gamma', \Delta'\right) . \Gamma'' \vdash A | \Delta'' \to \Gamma'' \vdash \Delta''\right) \to \Gamma' \vdash \Delta'.$$

We can derive $\Gamma \vdash A | \Delta$ by first applying the (μ)rule, and then (*) with the (Ax_L)rule.

Let $(\Gamma, \Delta) : A \Vdash$ for A-atomic, that is, let

$$(\#) \qquad \forall \left(\Gamma', \Delta'\right) \geq (\Gamma, \Delta) . \Gamma' | A \vdash \Delta' \to \Gamma' \vdash \Delta'.$$

The derivation of $\Gamma | A \vdash \Delta$ is immediate by applying $(\tilde{\mu})$, (*) and (Ax_R).

Induction case for $(\wedge)$. Let $(\Gamma, \Delta) \Vdash A \wedge B$, that is,

$$(*) \quad \forall \left(\Gamma', \Delta'\right) \geq (\Gamma, \Delta) . \left(\forall \left(\Gamma'', \Delta''\right) \geq \left(\Gamma', \Delta'\right) . \left(\Gamma'', \Delta''\right) \Vdash A \text{ and } \left(\Gamma'', \Delta''\right) \Vdash B \to \Gamma'' \vdash \Delta''\right) \to \Gamma' \vdash \Delta'.$$

We derive $\Gamma \vdash A \wedge B | \Delta$ by the rule $(\wedge_R)$and show each of the conjuncts separately. For example, to show $\Gamma \vdash A | \Delta$, we apply (μ), then (*), and then, given

$$(\Gamma, (A, \Delta)) \Vdash A \text{ and } (\Gamma, (A, \Delta)) \Vdash B,$$

we have to show $\Gamma \vdash A, \Delta$, but that is immediate by the induction hypothesis (1) and the (Ax_L)rule.

Let $(\Gamma, \Delta) : A \wedge B \Vdash$, that is,

$$(\#) \qquad \forall \left(\Gamma', \Delta'\right) \geq (\Gamma, \Delta) . \left(\Gamma', \Delta'\right) \Vdash A \text{ and } \left(\Gamma', \Delta'\right) \Vdash B \to \Gamma' \vdash \Delta'.$$

To derive $\Gamma | A \wedge B \vdash \Delta$, we apply $(\tilde{\mu})$and then (#). We have to prove each of $((A \wedge B, \Gamma), \Delta) \Vdash A$ and $((A \wedge B, \Gamma), \Delta) \Vdash B$ separately. Let $(\Gamma', \Delta') \geq ((A \wedge B, \Gamma), \Delta)$ and let $(\Gamma', \Delta') : A \Vdash$. We show $\Gamma' \vdash \Delta'$ by using (Ax_R), $(\wedge^1_L)$ and the induction hypothesis (2). The proof of $((A \wedge B, \Gamma), \Delta) \Vdash B$ is analogous.

Induction case for $(\vee)$. Let $(\Gamma,\Delta) \Vdash A \vee B$, that is,

$$(*)\quad \forall\left(\Gamma',\Delta'\right) \geq (\Gamma,\Delta).\ \left(\forall\left(\Gamma'',\Delta''\right) \geq \left(\Gamma',\Delta'\right).\right.$$
$$\left.\left(\Gamma'',\Delta''\right) \Vdash A \text{ or } \left(\Gamma'',\Delta''\right) \Vdash B \to \Gamma'' \vdash \Delta''\right) \to \Gamma' \vdash \Delta'.$$

We derive $\Gamma | A \vee B \vdash \Delta$ by the rule $(\mu)$and applying (*). Then, given

$$((A \vee B,\Gamma),\Delta) \Vdash A \text{ or } ((A \vee B,\Gamma),\Delta) \Vdash B,$$

we have to show that $A \vee B,\Gamma \vdash \Delta$. This is done by case distinction; if $((A \vee B,\Gamma),\Delta) \Vdash A$, then we use the induction hypothesis (1) to obtain $A \vee B,\Gamma \vdash A|\Delta$, and hence, using $(\vee^1_R)$ and (Ax_L)we get $A \vee B,\Gamma \vdash \Delta$. The case $((A \vee B,\Gamma),\Delta) \Vdash A$ is analogous.

Let $(\Gamma,\Delta) : A \vee B \Vdash$, that is,

$$(\#)\qquad \forall\left(\Gamma',\Delta'\right) \geq (\Gamma,\Delta).\ \left(\Gamma',\Delta'\right) \Vdash A \text{ or } \left(\Gamma',\Delta'\right) \Vdash B \to \Gamma' \vdash \Delta'.$$

To derive $\Gamma | A \vee B \vdash \Delta$, we apply $(\vee_L)$ and we have to derive each of $\Gamma | A \vdash \Delta$ and $\Gamma | B \vdash \Delta$ separately. We only derive the first one, by applying (μ), then (Cut)with $(\mathrm{Ax}_R)$and applying induction hypothesis (2) for A: given $((A,\Gamma),\Delta) \Vdash_s A$, we can use (#) and Lemma 2.4.3 to derive $(A,\Gamma) \vdash \Delta$.

Induction case for $(\Rightarrow)$. Let $(\Gamma,\Delta) : A \Rightarrow B \Vdash$, that is,

$$(*)\quad \forall\left(\Gamma',\Delta'\right) \geq (\Gamma,\Delta).\ \left(\forall\left(\Gamma'',\Delta''\right) \geq \left(\Gamma',\Delta'\right).\right.$$
$$\left.\left[\forall(\Gamma''',\Delta''') \geq (\Gamma'',\Delta'').\ (\Gamma''',\Delta''') \Vdash A \to (\Gamma''',\Delta''') \Vdash B\right] \to \Gamma'' \vdash \Delta''\right) \to \Gamma' \vdash \Delta'$$

To derive $\Gamma \vdash A \Rightarrow B | \Delta$, we apply $(\Rightarrow_R)$, then (μ), then (*), and, given

$$(\#)\qquad \forall(\Gamma_3,\Delta_3) \geq ((A,\Gamma),(B,\Delta)).\ (\Gamma_3,\Delta_3) \Vdash A \to (\Gamma_3,\Delta_3) \Vdash B,$$

we have to derive $\Gamma_3 \vdash \Delta_3$. We apply (Cut)with the formula B, and then use IH (1) followed by (#). Now, we have to show $(\Gamma_3,\Delta_3) \Vdash A$. Let $(\Gamma_4,\Delta_4) \geq (\Gamma_3,\Delta_3)$ and $(\Gamma_4,\Delta_4) : A \Vdash$. We derive $\Gamma_4 \vdash \Delta_4$ by a (Cut)with $(\mathrm{Ax}_R)$and IH (2) applied to the last hypothesis.

Let now $(\Gamma,\Delta) : A \Rightarrow B \Vdash$, that is,

(#)

$$\forall\left(\Gamma',\Delta'\right) \geq (\Gamma,\Delta).\ \left[\forall(\Gamma'',\Delta'') \geq (\Gamma',\Delta').\ (\Gamma'',\Delta'') \Vdash A \to (\Gamma'',\Delta'') \Vdash B\right] \to \Gamma' \vdash \Delta'.$$

We derive $\Gamma | A \Rightarrow B \vdash \Delta$ by applying $(\tilde{\mu})$, then (#), and then, given $((A \Rightarrow B,\Gamma),\Delta) \Vdash A$ and $((A \Rightarrow B,\Gamma),\Delta) : B \Vdash$, we can derive $A \Rightarrow B,\Gamma \vdash \Delta$ by a (Cut)with $A \Rightarrow B$, using the evaluation context constructed by combining the last two hypothesis with IH (1) and IH (2), respectively.

The induction cases for $\forall$ and $\exists$ are proved analogously to those of $\Rightarrow$ and $\vee$, but more simply. The cases of $\top$ and $\bot$ are trivial. □

2.4.9. Corollary. *For any sentence A and contexts of sentences Γ,Δ, the following hold for $\mathscr{U}$:*

(1) *If $A \in \Gamma$ then $(\Gamma,\Delta) \Vdash A$.*
(2) *If $B \in \Delta$ then $(\Gamma,\Delta) : B \Vdash$.*

PROOF. (1) Assume $A \in \Gamma$, $(\Gamma',\Delta') \geq (\Gamma,\Delta)$ and $(\Gamma',\Delta') : A \Vdash$. Then, by Theorem 2.4.8, $\Gamma' \mid A \vdash \Delta'$, hence we can obtain a cut-free proof of $\Gamma' \vdash \Delta'$ using left contraction (page 34).

(2) Assume $B \in \Delta$, $(\Gamma',\Delta') \geq (\Gamma,\Delta)$ and $(\Gamma',\Delta') \Vdash_s B$. Then, by Lemma 2.4.3 and Theorem 2.4.8, $\Gamma' \vdash B \mid \Delta'$, hence we can obtain a cut-free proof of $\Gamma' \vdash \Delta'$ using right contraction (page 35).

□

2.4.10. Corollary (Completeness of Classical Logic)**.** *If in every classical Kripke model, at every possible world, the formula A is forced whenever all the formulae of Γ are forced and all the formulae of Δ are refuted, then there exists a derivation in $\mathrm{LK}_{\mu\tilde{\mu}}$ of the sequent $\Gamma \vdash A|\Delta$.*

PROOF. Analogous to Corollary 2.3.14, by using Corollary 2.4.9. □

We again have the following corollary.

2.4.11. Corollary (Semantic Cut-Elimination)**.** *For all contexts Γ, Δ of sentences, if there is a derivation of $\Gamma \vdash \Delta$, then there is a cut-free derivation of $\Gamma \vdash \Delta$.*

PROOF. From the hypothesis $\Gamma \vdash \Delta$, the soundness theorem applied to $\mathscr{U}$ gives us that there is indeed a cut-free derivation for $\Gamma \vdash \Delta$ because the world (Γ, Δ) forces all formulae of Γ and refutes all formulae of Δ as shown in Corollary 2.4.9. Note, however, that the proof of Corollary 2.4.9 is not analogous to the one of Corollary 2.3.13. □

2.5. Computational content

So far, we have not justified the headings "call-by-name variant" and "call-by-value variant" of Sections 2.3 and 2.4. We do that in this section by showing that the computational content of Corollary 2.3.15 is $\bar{\lambda}\mu\tilde{\mu}$-command normalisation in call-by-name strategy, while the computational content of Corollary 2.4.11 is call-by-value normalisation.

One of the strengths of the $\bar{\lambda}\mu\tilde{\mu}$-calculus is that it is able to very simply characterise call-by-name (CBN) and call-by-value (CBV) evaluation strategies. If we consider again the reduction rules of Definition 2.2.2, we see that there is a critical pair $\langle \mu\alpha.c \| \tilde{\mu}a.c' \rangle$ for which we do not know whether to first apply the $\rightarrow_\mu$ rule or the $\rightarrow_{\tilde{\mu}}$ rule. As it is shown in [**53**], giving priority to the $\rightarrow_{\tilde{\mu}}$ rule gives rise to a CBN reduction strategy, while giving priority to the $\rightarrow_\mu$ rule gives rise to a CBV strategy.

Now, since we have formalised our proofs in the Coq proof assistant, which is based on a constructive theory of types, we can compute with the proofs directly, and see how "semantic cut-elimination" is performed. This is the technique of proof by reflection that we mentioned in Section 1.5. We restrict ourselves to the propositional fragment, because we did not formalise the quantifier version of the CBN classical Kripke model (see Section 2.6). The results are given in Tables 3 and 4.

We can see that Corollary 2.3.15 reduces all commands in perfect accord with the reduction relation of Definition 2.2.2 for CBN strategy, however, Corollary 2.4.11 is not in perfect accord, because, although it reduces all commands according to the CBV strategy, there are two η-redexes dangling on the evaluation context side. Hence, one is forced to perform two more steps of reduction,

$$\langle p \| \tilde{\mu}d.\langle \mu\delta.\langle d \| \delta \rangle \| \beta \rangle \rangle \rightarrow_{\tilde{\mu}} \langle \mu\delta.\langle p \| \delta \rangle \| \beta \rangle \rightarrow_\mu \langle p \| \beta \rangle,$$

in order to obtain a final cut-free form.

We have tracked this difference of behaviour down to the difference in the proofs of Corollaries 2.3.13 and 2.4.9. However, we do not claim that the same problem with a final η-redex would happen if the simplified Theorem 2.3.9 had

$$\begin{aligned}
\langle \mu\alpha.\langle a\|\beta\rangle \| \tilde{\mu}b.\langle c\|\gamma\rangle\rangle &\to \langle c\|\gamma\rangle \\
\langle \mu\alpha.\langle a\|\alpha\rangle \| \beta\rangle &\to \langle a\|\beta\rangle \\
\langle \lambda a.a\| b\cdot\beta\rangle &\to \langle b\|\beta\rangle \\
\langle (a,b)\|\pi_1\alpha\rangle &\to \langle a\|\alpha\rangle \\
\langle (a,b)\|\pi_2\alpha\rangle &\to \langle b\|\alpha\rangle \\
\langle \iota_1 a\|[\alpha,\beta]\rangle &\to \langle a\|\alpha\rangle \\
\langle \iota_2 a\|[\alpha,\beta]\rangle &\to \langle a\|\beta\rangle
\end{aligned}$$

(different letters stand for distinct entities)

Table 3: Algorithmic behaviour of the CBN models

$$\begin{aligned}
\langle \mu\alpha.\langle a\|\beta\rangle \| \tilde{\mu}b.\langle c\|\gamma\rangle\rangle &\to \langle a\|\tilde{\mu}d.\langle \mu\delta.\langle d\|\delta\rangle \|\beta\rangle\rangle &&\to^2 \langle a\|\beta\rangle \\
\langle \mu\alpha.\langle a\|\alpha\rangle \| \beta\rangle &\to \langle a\|\tilde{\mu}d.\langle \mu\delta.\langle d\|\delta\rangle \|\beta\rangle\rangle &&\to^2 \langle a\|\beta\rangle \\
\langle \lambda a.a\| b\cdot\beta\rangle &\to \langle b\|\tilde{\mu}d.\langle \mu\delta.\langle d\|\delta\rangle \|\beta\rangle\rangle &&\to^2 \langle b\|\beta\rangle \\
\langle (a,b)\|\pi_1\alpha\rangle &\to \langle a\|\tilde{\mu}d.\langle \mu\delta.\langle d\|\delta\rangle \|\alpha\rangle\rangle &&\to^2 \langle a\|\alpha\rangle \\
\langle (a,b)\|\pi_2\alpha\rangle &\to \langle b\|\tilde{\mu}d.\langle \mu\delta.\langle d\|\delta\rangle \|\alpha\rangle\rangle &&\to^2 \langle b\|\alpha\rangle \\
\langle \iota_1 a\|[\alpha,\beta]\rangle &\to \langle a\|\tilde{\mu}d.\langle \mu\delta.\langle d\|\delta\rangle \|\alpha\rangle\rangle &&\to^2 \langle a\|\alpha\rangle \\
\langle \iota_2 a\|[\alpha,\beta]\rangle &\to \langle a\|\tilde{\mu}d.\langle \mu\delta.\langle d\|\delta\rangle \|\beta\rangle\rangle &&\to^2 \langle a\|\beta\rangle
\end{aligned}$$

(different letters stand for distinct entities)

Table 4: Algorithmic behaviour of the CBV models

been used for computation. It may well be that a different notion of call-by-value model is necessary in order to obtain full normalisation, for example a model along the lines of the CPS translation of Section 4.2.

We have manually extracted the algorithms behind the proofs of Theorems 2.3.9, 2.3.12 and 2.4.8, because the machine extracted algorithms of 2.3.12 and 2.4.8 contain too many details to be comprehensible, while 2.3.9 was not at all formalised. They are given in Tables 5, 6, and 7. For meta-level abstraction and application we use "$\mapsto$" and "$\cdot$", for meta-level pair projections we use "fst" and "snd", while for meta-level left- and right-injection we use "inl" and "inr". Definition by cases is written as usually in Mathematics, pairs are written by (H,G), and there is a shortcut "ret$\cdot H$" which stands for "$G \mapsto G\cdot H$".

2.6. Aspects of the Coq formalisation

The source code of the Coq formalisation is available at the address `http://www.lix.polytechnique.fr/~danko/code`. We had formalised first the proof of Theorem 2.3.12, only for the fragment $\{\Rightarrow,\wedge,\vee\}$, and then we rewrote the formalisation for Theorem 2.4.8, in parallel to developing the formalisation of Chapter 3. These last two formalisations include also the quantifiers. We remark that the proof of Theorem 2.4.8 is analogous to the one of Theorem 2.3.9,

$$
\begin{aligned}
&\downarrow^A : (\Gamma,\Delta) \Vdash A \to \{p \mid \Gamma \vdash p : A \mid \Delta\}\\
&\uparrow^A : \mathrm{NT}(a : A,\Gamma,\Delta) \to (\Gamma,\Delta) \Vdash A\\
&\Downarrow^A : (\Gamma,\Delta) : A \Vdash\!\to \{e \mid \Gamma \mid e : A \vdash \Delta\}\\
&\Uparrow^A : \mathrm{NE}(\alpha : A,\Gamma,\Delta) \to (\Gamma,\Delta) : A \Vdash\\
\\
&\downarrow^X := H \mapsto \mu\alpha.H\cdot\alpha\\
&\uparrow^X := H \mapsto H' \mapsto H\cdot H'\\
&\Downarrow^X := H \mapsto \tilde{\mu}a.H\cdot(e \mapsto \langle a \| e\rangle)\\
&\Uparrow^X := H \mapsto H' \mapsto H\cdot\left(\mu\alpha.H'\cdot\alpha\right)\\
&\downarrow^{A\wedge B} := H \mapsto \begin{array}{l}\left(\mu\alpha.H\cdot\left(\mathrm{inl}\cdot\left(\Uparrow^A p \mapsto \langle p\|\alpha\rangle\right)\right),\right.\\ \left.\mu\alpha.H\cdot\left(\mathrm{inr}\cdot\left(\Uparrow^B p \mapsto \langle p\|\alpha\rangle\right)\right)\right)\end{array}\\
&\uparrow^{A\wedge B} := H \mapsto H' \mapsto \begin{cases} H\cdot\left(\iota_1\left(\tilde{\mu}a.H''\cdot\left(\uparrow^A e \mapsto \langle a\|e\rangle\right)\right)\right) & \text{if } H' = \mathrm{inl}\cdot H''\\ H\cdot\left(\iota_2\left(\tilde{\mu}a.H''\cdot\left(\uparrow^B e \mapsto \langle a\|e\rangle\right)\right)\right) & \text{if } H' = \mathrm{inr}\cdot H''\end{cases}\\
&\Downarrow^{A\wedge B} := H \mapsto \tilde{\mu}a.H\cdot\left(H' \mapsto \begin{cases}\langle a\|\pi_1\left(\tilde{\mu}b.H''\cdot\left(\uparrow^A e \mapsto \langle b\|e\rangle\right)\right)\rangle, & \text{if } H' = \mathrm{inl}\cdot H''\\ \langle a\|\pi_2\left(\tilde{\mu}b.H''\cdot\left(\uparrow^B e \mapsto \langle b\|e\rangle\right)\right)\rangle, & \text{if } H' = \mathrm{inr}\cdot H''\end{cases}\right)\\
&\Uparrow^{A\wedge B} := H \mapsto H' \mapsto H\cdot \begin{array}{l}\left(\mu\alpha.H'\cdot\left(\mathrm{inl}\cdot\left(\Uparrow^A p \mapsto \langle p\|\alpha\rangle\right)\right),\right.\\ \left.\mu\alpha.H'\cdot\left(\mathrm{inr}\cdot\left(\Uparrow^B p \mapsto \langle p\|\alpha\rangle\right)\right)\right)\end{array}\\
&\downarrow^{A\vee B} := H \mapsto \mu\gamma.\left\langle\iota_1\left(\mu\alpha.\left\langle\iota_2\left(\mu\beta.H\cdot\left(\Uparrow^A p \mapsto \langle p\|\alpha\rangle, \Uparrow^B p \mapsto \langle p\|\beta\rangle\right)\right)\|\gamma\right\rangle\right)\|\gamma\right\rangle\\
&\uparrow^{A\vee B} := H \mapsto H' \mapsto H\cdot \begin{array}{l}\left[\tilde{\mu}a.\left(\mathrm{fst}\cdot H'\right)\cdot\left(\uparrow^A e \mapsto \langle a\|e\rangle\right),\right.\\ \left.\tilde{\mu}a.\left(\mathrm{snd}\cdot H'\right)\cdot\left(\uparrow^B e \mapsto \langle a\|e\rangle\right)\right]\end{array}\\
&\Downarrow^{A\vee B} := H \mapsto \begin{array}{l}\left[\tilde{\mu}a.H\cdot\left(H' \mapsto \left(\mathrm{fst}\cdot H'\right)\cdot\left(\uparrow^A e \mapsto \langle a\|e\rangle\right)\right),\right.\\ \left.\tilde{\mu}a.H\cdot\left(H' \mapsto \left(\mathrm{snd}\cdot H'\right)\cdot\left(\uparrow^B e \mapsto \langle a\|e\rangle\right)\right)\right]\end{array}\\
&\Uparrow^{A\vee B} := H \mapsto H' \mapsto H\cdot\left(\iota_1\left(\mu\alpha.H\cdot\left(\iota_2\left(\mu\beta.H'\cdot\left(\Uparrow^A p \mapsto \langle p\|\alpha\rangle, \Uparrow^B p \mapsto \langle p\|\beta\rangle\right)\right)\right)\right)\right)\\
&\downarrow^{A\Rightarrow B} := H \mapsto \lambda a.\mu\alpha.H\cdot\left(\uparrow^A e \mapsto \langle a\|e\rangle\right)\cdot\left(\Uparrow^B p \mapsto \langle p\|\alpha\rangle\right)\\
&\uparrow^{A\Rightarrow B} := H \mapsto H' \mapsto H\cdot\left(\left(\downarrow^A \mathrm{fst}\cdot H'\right)\cdot\left(\Downarrow^B \mathrm{snd}\cdot H'\right)\right)\\
&\Downarrow^{A\Rightarrow B} := H \mapsto \left(\mu\alpha.H\cdot\left(\uparrow^A p \mapsto \langle p\|\alpha\rangle\right)\right)\cdot\left(\tilde{\mu}a.H\cdot\left(\Uparrow^B e \mapsto \langle a\|e\rangle\right)\right)\\
&\Uparrow^{A\Rightarrow B} := H \mapsto H' \mapsto H\cdot\left(\lambda a.\mu\alpha.H'\cdot\left(\downarrow^A e \mapsto \langle a\|e\rangle\right)\cdot\left(\Downarrow^B p \mapsto \langle p\|\alpha\rangle\right)\right)
\end{aligned}
$$

Table 5: The normalisation algorithm behind Theorem 2.3.12

which we gave in our article [**109**], therefore obtaining a formal version of the theorem from the article should not be too difficult.

The way of representing binders, substitutions and the co-finite quantification rule is the same as in Section 1.5, except that the definition of formulae does not include Henkin constants. However, the definition of individual terms from the formalisations of Chapters 2 and 3, allows to have multiple arguments for predicate and function symbols, which means that these completeness proofs are more prepared for practical application inside Coq, that the one of Chapter 1. Formally, in type theory, fixed points for working with such individual

$$
\begin{aligned}
&\downarrow^{A} : (\Gamma,\Delta) \Vdash A \to \{p \mid \Gamma \vdash p : A \mid \Delta\}\\
&\Downarrow^{A} : (\Gamma,\Delta) : A \Vdash\!\to \{e \mid \Gamma \mid e : A \vdash \Delta\}\\
&\\
&\downarrow^{X} := H \mapsto \mu\alpha.H \cdot \alpha\\
&\Downarrow^{X} := H \mapsto \tilde{\mu}a.H \cdot (e \mapsto \langle a \| e \rangle)\\
&\downarrow^{A\wedge B} := H \mapsto \begin{array}{l}\left(\downarrow^{A}\left(H' \mapsto H \cdot \left(\mathrm{ret} \cdot \left(\mathrm{inl} \cdot H'\right)\right)\right)\right),\\ \downarrow^{B}\left(H' \mapsto H \cdot \left(\mathrm{ret} \cdot \left(\mathrm{inr} \cdot H'\right)\right)\right)\end{array}\\
&\Downarrow^{A\wedge B} := H \mapsto \tilde{\mu}a.H \cdot \left(H' \mapsto \begin{cases} \langle a \| \pi_1 \left(\Downarrow^{A} H''\right)\rangle, & \text{if } H' = \mathrm{inl} \cdot H'' \\ \langle a \| \pi_2 \left(\Downarrow^{B} H''\right)\rangle, & \text{if } H' = \mathrm{inr} \cdot H'' \end{cases}\right)\\
&\downarrow^{A\vee B} := H \mapsto \mu\alpha.\left\langle \iota_1 \left(\mu\beta.\left\langle \iota_2 \left(\mu\gamma.H \cdot \left(\left\langle H' \mapsto \downarrow^{A} H' \| \gamma \right\rangle, \left\langle H' \mapsto \downarrow^{B} H' \| \gamma \right\rangle\right)\right) \| \beta \right\rangle\right) \| \alpha \right\rangle\\
&\Downarrow^{A\vee B} := H \mapsto \begin{array}{l}\left[\Downarrow^{A}\left(H' \mapsto H \cdot \left(H'' \mapsto \left(\mathrm{fst} \cdot H''\right) \cdot H'\right)\right),\right.\\ \left.\Downarrow^{B}\left(H' \mapsto H \cdot \left(H'' \mapsto \left(\mathrm{snd} \cdot H''\right) \cdot H'\right)\right)\right]\end{array}\\
&\downarrow^{A\Rightarrow B} := H \mapsto \lambda a.\mu\alpha.H \cdot \left(H' \mapsto \left\langle a \| \Downarrow^{A} \left(\mathrm{ret} \cdot H'\right)\right\rangle, H' \mapsto \left\langle \downarrow^{B} H' \| \alpha \right\rangle\right)\\
&\Downarrow^{A\Rightarrow B} := H \mapsto \tilde{\mu}a.H \cdot \left(H' \mapsto \left\langle a \| \left(\downarrow^{A} \left(\mathrm{fst} \cdot H'\right)\right) \cdot \left(\Downarrow^{B} \left(\mathrm{snd} \cdot H'\right)\right)\right\rangle\right)
\end{aligned}
$$

Table 6: The normalisation algorithm behind Theorem 2.3.9

$$
\begin{aligned}
&\downarrow^{A} : (\Gamma,\Delta) \Vdash A \to \{p \mid \Gamma \vdash p : A \mid \Delta\}\\
&\Downarrow^{A} : (\Gamma,\Delta) : A \Vdash\!\to \{e \mid \Gamma \mid e : A \vdash \Delta\}\\
&\\
&\downarrow^{X} := H \mapsto \mu\alpha.H \cdot (p \mapsto \langle p \| \alpha \rangle)\\
&\Downarrow^{X} := H \mapsto \tilde{\mu}a.H \cdot a\\
&\downarrow^{A\wedge B} := H \mapsto \begin{array}{l}\left(\mu\alpha.H \cdot \left(H' \mapsto \langle \downarrow^{A} \left(\mathrm{fst} \cdot H'\right) \| \alpha \rangle\right),\right.\\ \left.\mu\alpha.H \cdot \left(H' \mapsto \langle \downarrow^{B} \left(\mathrm{snd} \cdot H'\right) \| \alpha \rangle\right)\right)\end{array}\\
&\Downarrow^{A\wedge B} := H \mapsto \tilde{\mu}a.H \cdot (H' \mapsto \langle a \| \pi_1 (\Downarrow^{A} H') \rangle, H' \mapsto \langle a \| \pi_2 (\Downarrow^{B} H') \rangle)\\
&\downarrow^{A\vee B} := H \mapsto \mu\alpha.H \cdot \left(H' \mapsto \begin{cases} \langle \iota_1 \left(\downarrow^{A} H''\right) \| \alpha \rangle, & \text{if } H' = \mathrm{inl} \cdot H'' \\ \langle \iota_2 \left(\downarrow^{B} H''\right) \| \alpha \rangle, & \text{if } H' = \mathrm{inr} \cdot H'' \end{cases}\right)\\
&\Downarrow^{A\vee B} := H \mapsto \begin{array}{l}\left[\tilde{\mu}a.\langle a \| \Downarrow^{A} \left(H' \mapsto H \cdot \left(\mathrm{inl} \cdot \left(\mathrm{ret} \cdot H'\right)\right)\right)\rangle,\right.\\ \left.\tilde{\mu}a.\langle a \| \Downarrow^{B} \left(H' \mapsto H \cdot \left(\mathrm{inr} \cdot \left(\mathrm{ret} \cdot H'\right)\right)\right)\rangle\right]\end{array}\\
&\downarrow^{A\Rightarrow B} := H \mapsto \lambda a.\mu\alpha.H \cdot \left(H' \mapsto \langle \downarrow^{B} \left(H' \cdot \left(H'' \mapsto \langle a \| \Downarrow^{A} H''\rangle\right) \| \alpha \rangle\right)\right)\\
&\Downarrow^{A\Rightarrow B} := H \mapsto \tilde{\mu}a.H \cdot \left(H' \mapsto H'' \mapsto \langle a \| \left(\downarrow^{A} H'\right) \cdot \left(\Downarrow^{B} H''\right)\rangle\right)
\end{aligned}
$$

Table 7: The normalisation algorithm behind Theorem 2.4.8

terms require a "trick" so that they are definable by primitive (structural) recursion. We have to mutually define two fixpoints, one working on an individual, the other on a list of individuals, but the one that works on lists has to be a local definition to the first one. For example, here is the formalisation of the fixpoint

which substitutes the free variable k of the individual term a with the individual term d:

```
Fixpoint subst_indiv (k:nat)(a:indiv)(d:indiv) {struct a} : indiv :=
  let subst_list := fix subst_list (l:list indiv) {struct l} : list indiv :=
    match l with
      | nil ⇒ nil
      | cons a' l' ⇒ cons (subst_indiv k a' d) (subst_list l')
    end in
  match a with
    | func f ld ⇒
      func f (subst_list ld)
    | fvar x ⇒ fvar x
    | bvar i ⇒ if beq_nat k i then d else bvar i
  end.
```

This kind of definition is analogous to the one of the Ackermann function using higher-type primitive recursion.

Coq was clever in being able to generate for us, using the `Scheme` command, a mutual induction predicate for derivations of $\mathrm{LK}_{\mu\tilde{\mu}}$. A slight anomaly was, however, that it knew how to generate such predicates only for sort Prop, not sort Type.

We can also say that in general the separation Prop/Set of Coq was not very convenient. While "usually", when an algorithm and its correctness proof are well separated, this distinction makes extracted terms be more readable, in these kind of meta-theoretic algorithms, logic is an integral part of the computation. It was thus necessary to rewrite a couple of lemmas from the Coq standard library, that deal with lists and number comparison, so that they live in Set instead of Prop.

2.7. Related and future work

2.7.1. Normalisation by evaluation. In Section 2.5 we showed that the computational content of the completeness proofs of this chapter is cut-elimination, or proof normalisation. However, we did not obtain a result as good as the one of Berger and Schwichtenberg [**33**], because they showed that the input and the output of their NBE algorithm are $\beta\eta$-equal. In future, we would like to extend our work in that direction. We will have to be more careful when treating η-equality in the context of $\bar{\lambda}\mu\tilde{\mu}$ because a more finely grained reduction rule for $(\rightarrow_{\Rightarrow})$ will then be needed (see the Habilitation thesis of Herbelin [**98**]).

The work of Okada [**147**] is related to NBE for classical logic. Directly inspired by Girard's phase semantics [**80**] for linear logic, Okada shows that there is a uniform cut-eliminating completeness proof for various extensions of intuitionistic linear logic, including classical predicate calculus. His approach is more algebraic than ours. While he defines his phase semantics from an abstract monoid structure and uses set comprehension to interpret the logical constants, we rely on a Kripke-style model, which is nothing else but a continuations monad. We are not aware of any explicit algorithms for cut-elimination that follow his approach.

Okada's semantics is related to Sambin's pre-topology semantics [**153**].

There are a number of works connected to NBE for intuitionistic systems that we review in Section 3.6.

2.7.2. Using Intuitionistic Kripke models on doubly negated formulae. Although one could probably use a double-negation interpretation A^* of formulae and an intuitionistic completeness theorem for tradition Kripke models to obtain a normalisation result, one would have to do it in multiple phases,

$$(*) \qquad \vdash_c A \longrightarrow \vdash_i A^* \longrightarrow \Vdash_i A^* \longrightarrow \vdash_i^{nf} A^* \longrightarrow \vdash_c^{nf} A$$

where "i" stands for "intuitionistic", "c" for "classical" and "nf" for "in normal form", in which how to do the last step is not obvious. We consider that to be a *detour* since we can prove, simply,

$$\vdash_c A \longrightarrow \Vdash_c A \longrightarrow \vdash_c^{nf} A$$

The interest in having a direct-style semantics for classical logic is the same as the interest in having a proof calculus for classical logic instead of restricting oneself to an intuitionistic calculus and working with DN-translated formulae; or, in the theory of programming languages, to having a separate constant *call/cc* instead of writing all programs in continuation-passing style.

Avigad shows in [**16**] how classical cut-elimination is a special case of intuitionistic one, work which resembles the phases (*). However, his work is specialised to negative formulae, that is, it is not clear how to extend it to formulae that use $\vee$ and $\exists$.

2.7.3. Boolean versus Kripke semantics for classical logic. We began this chapter by a desire to obtain a more "canonical" completeness proof for classical logic. We showed that completeness w.r.t. classical Kripke models consists of β-reduction of $\bar{\lambda}\mu\tilde{\mu}$-commands, while for Boolean models the computational contents is finding maximum values using a supplied enumeration of formulae.

In future, we would like to investigate how practical classical Kripke models are for model theoretic reasoning. Some work along those lines has been done in the 1960s by Fitting [**77**]. Although conducted in a *classical* meta-language, the work indicates that it is possible to use "intuitionistic" Kripke models to express elegantly some cumbersome constructions of model theory, like set theoretic forcing [**54, 77**]. Indeed, the connection between the two had been spotted already by Kripke [**122, 123**] and that is why the term "forcing" appeared in his semantics. (however, see also [**117**]) We hope that looking at those kind of constructions inside Kripke models, but this time inside a *constructive* meta-language, might be one way to finding out the constructive content of techniques of classical model theory.

In this respect, our work can also be seen as a contribution to the field of constructive model theory of classical logic.

CHAPTER 3

Kripke-style models for intuitionistic logic

We saw in Section 2.1 that normalisation-by-evaluation can be seen as completeness for intuitionistic logic with respect to Kripke models, with "$\Rightarrow$" as the sole logical connective. It is easy to add "$\wedge$" to the picture (by NBE for λ-calculus with pairs), as well as "$\forall$" (see [**102**] for a formal proof in Coq). The starting point for the work presented in this chapter was the attempt to extend completeness for Kripke models to handle also the connectives "$\vee$" and "$\exists$".

The problem of the possibility of a constructive completeness proof for full intuitionistic logic has been considered in the past, notably by Kreisel and Veldman, as we review in Section 3.1. In Section 3.2, we look at the other side of the coin NBE/completeness, an algorithm of Danvy related to NBE, the one of type-directed partial evaluation (TDPE) for λ-calculus with sums. In Section 3.3, inspired from Danvy's TDPE algorithm, we introduce a notion of model, similar to Kripke models, for which intuitionistic predicate logic (with $\vee$ and $\exists$) can be shown to be sound and complete. In Section 3.4, we give the normalisation algorithm behind the completeness proof of Section 3.3. In Section 3.5, we discuss some aspects of the Coq formalisation of the proofs, and, finally, we conclude in Section 3.6 by discussing related and future work.

3.1. Historical overview

The intuitionistic interpretation of the logical constants today known as the Brouwer-Heyting-Kolmogorov (BHK) interpretation appeared, implicitly, in the PhD thesis of Brouwer.[1] It was Kolmogorov the first who explicitly wrote about what it means for a formula to be intuitionistically true, for the propositional fragment, and, independently, Heyting gave essentially the same explanation that we know today, based on the notion of proof as a mathematical object: A is valid if there is a proof of A; a proof of $A \vee B$ is a proof of either A or B, together with information about which one of the two has been proved; a proof of $A \Rightarrow B$ is a procedure which transforms a proof of A into a proof of B; etc.

Kleene [**113**] took an alternative approach, by proposing interpretations of formulae by "realisability": a formula A is realisable if there is a recursive procedure computing a witness for A. For a historical overview of realisability, from Kleene to modern times, the article of van Oosten is to be recommended [**174**].

Both the BHK and Kleene's interpretation, as well as the various semantics that go under the heading Algebraic semantics ([**162**, Section 13.5]), can be seen as just an informal reformulation of the derivation rules of a formal system for intuitionistic logic. The earliest semantics that is not immediately connectible

[1] For references and more detail about this paragraph please look at Troelstra's historical survey [**164**].

to a derivation system is the one of Beth [**163**], who was also the first to propose a completeness proof for it.[2] The importance of Beth's semantics is in its giving an *independent* way to characterise intuitionistic validity, something that should be parallelled to what happens in the classical case: Boolean semantics and classical provability appear to be different ways to specify classical validity.

After Beth, Kripke [**123**] also presented such an independent kind of semantics (given in Definition 2.1.1), based on his previous work on semantics of modal logic [**121**], together with a classical completeness proof for it, a more modern version of which can be found in [**161**, Section 2.6]. Kripke's intuitionistic semantics was inspired from Gödel's embedding of intuitionistic into modal logic [**86**] (see Troelstra's introduction note to [**86**] and the introduction section of [**123**]).

3.1.1. Kreisel's meta-mathematical argument. In the interim between Beth and Kripke, in a series of articles [**119, 116, 120**][3], Kreisel, based on observations of Gödel, treated the problem of constructive provability of completeness for intuitionistic logic (Heyting's predicate calculus, IQC), while being purposely imprecise [**119**, pp.318–320] about the notion of truth used.

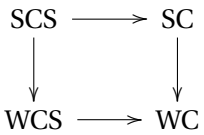
He considers a formula A to be valid ($\vDash A$) if it is valid for any domain of quantification D and any interpretation $\vec{P}$ of atomic formulae appearing in A (notation $(D,\vec{P}) \vDash A$), without going into details of how $(D,\vec{P}) \vDash A$ is to be defined. He just remarks that his approach is parallel to the one of Gödel, when Gödel proved completeness relying on an intuitive notion of Boolean semantics (Section 1.1), and that both Beth's and Topological (Algebraic) semantics can be fit into this framework. Because every Kripke model can be converted to a Beth model[4], the results of Kreisel apply also to Kripke models.

However, it is not completely clear which principles are necessary to show the equivalence of the intuitive intuitionistic semantics "à la Tarski" and Beth semantics. In [**116**], Kreisel uses his system FC (abbreviation for "free choice sequences") to connect the two, while Troelstra and van Dalen [**162**, Sections 13.1-3] use the Fan theorem.

In the mentioned series of papers, Kreisel considers four statements of completeness, (we denote by $\mathscr{M}$ a pair $(D,\vec{P})$)

SC: Strong completeness: $(\forall \mathscr{M}.\, \mathscr{M} \vDash A) \rightarrow \vdash A$;
WC: Weak completeness: $\nvdash A \rightarrow \neg\, (\forall \mathscr{M}.\, \mathscr{M} \vDash A)$;
SCS: Strong completeness by substitution: $\exists \mathscr{M}\, (\mathscr{M} \vDash A \rightarrow \vdash A)$;
WCS: Weak completeness by substitution: $\exists \mathscr{M}\, (\nvdash A \rightarrow \mathscr{M} \nvDash A)$,

for which the following implications hold:



[2] Beth's proof turned out to be faulty, as noticed by Dyson and Kreisel [**64**], but it is possible to repair it. (See for example [**162**, Chapter 13].)

[3] See also the reviews [**140, 139, 141**].

[4] But not vice versa, since there are Beth models where $\Vdash A \vee B$ holds, but neither $\Vdash A$ nor $\Vdash B$ hold.

In Chapters 1 and 2 we have been proving SCS (but where $\vdash$ was classical derivation), by building universal models from which completeness in any model follows.

He then proves these results, about certain subclasses of formulae:

- IQC is WCS for negative formulae (i.e. without $\vee, \exists$ and where all atomic formulae are negated), that is, formulae having the form of a result of Gödel's double-negation translation; [**119**, Theorem 4]
- IQC is WCS for formulae where each atomic formula and each logical connective are doubly negated, that is, formulae that are a product of Kolmogorov's double-negation translation; [**119**, Theorem 5];
- IQC is SCS for quantifier-free and prenex formulae; [**116**, Theorem 5].

But, more interestingly for us, he also proves the following two theorems, that connect weak and strong completeness via Markov's Principle,

$$\text{(MP)} \qquad \neg\forall n A_0(n) \to \exists n \neg A_0(n),$$

and two schemes related to the Double-negation shift (DNS) schema, restricted to Σ^0_1-formulae,

$$\text{(DNS}^\Sigma_+\text{)} \qquad \forall\alpha\neg\neg\exists n A_0(\alpha, n) \to \forall\alpha\exists n A_0(\alpha, n),$$

and

$$\text{(DNS}^\Sigma\text{)} \qquad \forall\alpha\neg\neg\exists n A_0(\alpha, n) \to \neg\neg\forall\alpha\exists n A_0(\alpha, n),$$

where A_0 is a primitive recursive formula of HA^ω (i.e. A_0 is such that there exists a primitive recursive function f, such that $\mathrm{HA}^\omega \vdash \forall\alpha\forall n(f(\alpha, n) = 0 \leftrightarrow A_0(\alpha, n))$).

3.1.1. Theorem (Theorem 2 of [**119**])**.** *Given Markov's Principle, weak completeness of IQC implies strong completeness.*

PROOF. The derivability predicate is primitive recursive, that is, there is a primitive recursive function Prf, and an enumeration of formulae $\ulcorner\cdot\urcorner$, such that $\vdash A$ is a shortcut for $\exists n.\mathrm{Prf}(n, \ulcorner A\urcorner) = 0$.

Let now $\forall\mathscr{M}.\mathscr{M} \vDash A$. Then also $\neg\neg\forall\mathscr{M}.\mathscr{M} \vDash A$. By "contraposition" of WC, we get that $\neg \nvdash A$, that is, $\neg\neg\exists n.\mathrm{Prf}(n, \ulcorner A\urcorner) = 0$, which is in a form equivalent to the premise of MP. Hence, MP gives us that $\exists n.\neg\neg\,\mathrm{Prf}(n, \ulcorner A\urcorner) = 0$, and, since Prf is primitive recursive, $\exists n.\mathrm{Prf}(n, \ulcorner A\urcorner) = 0$ i.e. $\vdash A$. □

The proof did not use a definition of validity of A, and so it holds for any kind of semantics, that is, regardless of our previous comments about the link between intuitive and Beth/Kripke semantics via the Fan theorem.

3.1.2. Theorem (Theorem 1 of [**120**])**.** *If strong completeness of IQC holds, so does* DNS^Σ_+*. In particular, from the strong completeness of the negative fragment of formulae, MP holds.*

Kreisel announces this theorem already in [**119**], credits the observation to Gödel, and gives a proof in [**120**], saying that the result essentially relies on Gödel's incompleteness theorem. However, due to his use of the system FC (a system that diverges from both constructive and intuitionistic (in today's sense) systems), we have not yet been able to understand the proof.

3.1.3. Corollary. *Weak completeness of IQC implies* DNS^Σ.

PROOF. From $\mathrm{SC} \to \mathrm{DNS}^{\Sigma}_{+}$, we get $\neg\neg\,\mathrm{SC} \to \neg\neg\,\mathrm{DNS}^{\Sigma}_{+}$, hence[5]

$$\mathrm{WC} \to \neg\neg\,\mathrm{DNS}^{\Sigma}_{+} \to \mathrm{DNS}^{\Sigma}.$$

□

Thus, strong and weak completeness are equivalent in the presence of MP.

Finally, Kreisel remarks that, due to Theorem 3.1.2, the possibility of having strong completeness for IQC is not "plausible.", because MP is "not at all plausible on the intended interpretation of logical constants"; while, the possibility of weak completeness is "not so implausible", because DNS^{Σ} "may be provable on the basis of as yet undiscovered axioms which hold for the intended interpretation".

3.1.2. Veldman's proof. Nevertheless, Veldman gave in [**175**] an intuitionistic completeness proof for IQC with respect to Kripke models. He reasoned that one can circumvent the argument of Gödel and Kreisel, if "loosely speaking, negation is treated positively". He modified the notion of Kripke model to allow certain worlds to force absurdity, and proved strong completeness by substitution, that is, he built a universal Kripke model out of a derivation system, the set of possible worlds being a spread [**161**], and then used the Fan theorem to handle the cases of "$\vee$" and "$\exists$".

Note that, if one is interested only in completeness of minimal logic (MQC), Veldman's modified Kripke models become the same thing as standard Kripke models. In the models that we will give in Section 3.3, we will also use exploding nodes to "treat negation positively".

Veldman's approach has been extended by Friedman to handle Beth models. It appears that he did not publish his results, we know about them only from the detailed reconstruction by Troelstra and van Dalen [**162**, Chapter 13]. They, too, make use of the Fan theorem.

3.2. Type-directed partial evaluation for λ-calculus with sum

Instead of trying to find out the computational content of the arguments of the previous section by looking at realisability interpretations which validate the Fan theorem, we decided to follow the trail of an approach from Theoretical Computer Science.

It is the one of Danvy, who developed a computer program that characterises quite succinctly the computational content behind NBE for λ-calculus with sums. Actually, what he gave was an algorithm for *type-directed partial evaluation* (TDPE) of such a calculus.

Partial evaluation, in general, is a group of techniques from Semantics of Programming Languages, that allow a program to be specialised when some of its inputs are known in advance (static inputs), before the program is actually run on other inputs determined by the run-time environment (dynamic inputs). If we represent a program by a term in some typed λ-calculus,

$$\vdash p : \tau_s \to \tau_d \to \tau_r,$$

[5] Note, however, that $\mathrm{WC} := \neg\neg\,\mathrm{SC}$, and WC as defined in the beginning of this subsection, are only equivalent over intuitionistic logic; if one uses minimal logic, a principle like $\mathrm{DNS}^{\Rightarrow}_{T}$ of Section 4.2 is needed.

(where τ_s is a type for the static input, τ_d is a type for the dynamic input, and τ_r is a type of the result) a partial evaluator's task is to produce from p, and some fixed static input s, another program $\vdash p_s : \tau_d \to \tau_r$ such that the original and the produced program evaluate to the same value for any dynamic input d. Of course, one can trivially define $p_s := ps$, but the point in the context of programming languages is for the specialised program p_s to be more efficient than p applied to s at run-time. Typically, p_s is generated from p in order to skip unnecessary case distinctions on the input s, since branching (case distinction, if-then-else) is an expensive operation in today's microprocessors.

Type-directed partial evaluation is a technique invented by Danvy [**57, 56**] for generating a specialised program p_s, by recursion on the structure of the (desired) type of p_s, starting from the static input – evaluated/compiled version of the original program p. Danvy remarked that TDPE for simply typed λ-calculus coincides[6] with Berger and Schwichtenberg's NBE from Section 2.1, that is, we have the same pair of "reify" and "reflect" functions, but this time extended to the case of "$\vee$":

$$
\begin{array}{rll}
\downarrow^{\tau} : D \to \Lambda\text{-nf} & & \\
\downarrow^{\tau} := a \mapsto a & & \tau\text{-atomic} \\
\downarrow^{\tau\to\sigma} := S \mapsto \lambda a.\# \downarrow^{\sigma} (S \cdot \uparrow^{\tau} a) & & a\text{-fresh} \\
\downarrow^{\tau\vee\sigma} := S \mapsto \begin{cases} \iota_1(\downarrow^{\tau} S') & \text{, if } S = \mathrm{inl}\cdot S' \\ \iota_2(\downarrow^{\sigma} S') & \text{, if } S = \mathrm{inr}\cdot S' \end{cases} & &
\end{array}
$$

$$
\begin{array}{rll}
\uparrow^{\tau} : \Lambda\text{-ne} \to D & & \\
\uparrow^{\tau} := a \mapsto a & & \tau\text{-atomic} \\
\uparrow^{\tau\to\sigma} := e \mapsto S \mapsto \uparrow^{\sigma} e(\downarrow^{\tau} S) & & \\
\uparrow^{\tau\vee\sigma} := e \mapsto \mathscr{S}\kappa.\mathsf{case}\ e\ \mathsf{of}\ \left(a_1.\#\kappa \cdot (\mathrm{inl}\cdot(\uparrow^{\tau} a_1)) \| a_2.\#\kappa \cdot (\mathrm{inr}\cdot(\uparrow^{\sigma} a_2))\right) & & a_i\text{-fresh}
\end{array}
$$

As before, we can characterise explicitly typed λ-terms, normal λ-terms, and neutral λ-terms by the following inductive definitions.

$$
\begin{array}{rl}
\Lambda \ni p, q & := a^{\tau} \mid \lambda a^{\tau}.p^{\sigma} \mid p^{\tau\to\sigma} q^{\tau} \mid \iota_1 p \mid \iota_2 p \mid \mathsf{case}\ p^{\tau\vee\sigma}\ \mathsf{of}\ \left(a_1^{\tau}.q_1^{\rho} \| a_2^{\sigma}.q_2^{\rho}\right) \\
\Lambda\text{-nf} \ni r & := e^{\tau} \mid \lambda a^{\tau}.r^{\sigma} \mid \iota_1^{\tau} r \mid \iota_2^{\tau} r \\
\Lambda\text{-ne} \ni e & := a^{\tau} \mid e^{\tau\to\sigma} r^{\tau} \mid \mathsf{case}\ e^{\tau\vee\sigma}\ \mathsf{of}\ \left(a_1^{\tau}.r_1^{\rho} \| a_2^{\sigma}.r_2^{\rho}\right)
\end{array}
$$

The new symbols that appear in the algorithm above, # and $\mathscr{S}$, read "reset" and "shift", are known as *delimited control operators* [**58, 59**].[7] They are a

[6] For a nice connection between normalisation and partial evaluation, the lecture notes of Dybjer and Filinski [**63**] are to be recommended.

[7] A different kind of delimited control operators were proposed independently by Felleisen[**69**], based on his previous works with Friedman, Kohlbecker, Duba and Merrill.[**71, 72, 70**]. For a historic reflection on the discovery of these delimited control operators see Felleisen's appendix to the article of Ariola and Herbelin [**12**], or the PhD thesis of Biernacki [**35**].

very general way for allowing side-effects in the otherwise pure (effect-free) λ-calculus. Actually, in a sense, shift and reset are the most general way of supplying side-effects, because of the result of Filinski [**73, 74**], that shift and reset can simulate any monadic computational effect. A proper treatment of these delimited control operators is the subject of Chapter 4, here we give by example an explanation of their computational behaviour.

Reset # is a "control delimiter", its purpose is to limit the scope of the control operators shift ($\mathcal{S}$) it envelops. Shift ($\mathcal{S}\kappa.\pi$) then captures the part of the program (the "context" $P[\cdot]$) between the delimiter and itself, names it κ, and then inverts the context and π. (The precise reduction rules can be seen in Definition 4.1.6.) This is to be contrasted with what happens with the common use of control operators like call/cc, where the effect of a call to the control operator is not limited, that is, the *entire* context the program is run in is captured.

Let us look at some examples of computation with a λ-calculus extended by shift and reset, and where we have the plus operation on numbers. (plus binds more strongly than reset and shift)

$$1+\#2+\mathcal{S}k.4 \quad\to\quad 1+\#4\left\{(\lambda a.\#2+a)\big/k\right\} \quad=\quad 1+\#4 \quad\to\quad 1+4\to 5$$

In this example, the continuation variable k, which represents the environment up to the reset, was not used, hence shift just "escaped" with its result to the delimiter, ignoring the context "$2+\Box$" it was wrapped in.

$$\begin{aligned}
&1+\#2+\mathcal{S}k.k4+k8\\
\to&1+\#(\lambda a.\#2+a)4+(\lambda a.\#2+a)8\\
\to^+&1+\#(\#6)+(\#10)\\
\to^+&1+\#6+10\\
\to^+&17
\end{aligned}$$

In this example, the abstracted context was used twice, in two independent subexpressions.

We can treat the continuation variable k as any other variable, hence also compose it with itself, as the following example shows.

$$\begin{aligned}
&1+\#2+\mathcal{S}k.k(k4)+k8\\
\to&1+\#(\lambda a.\#2+a)\,((\lambda a.\#2+a)4)+(\lambda a.\#2+a)8\\
\to^+&1+\#(\#\#8)+(\#10)\\
\to^+&1+\#8+10\\
\to^+&19
\end{aligned}$$

We now show how the TDPE algorithm works, by applying it to the lambda term $\lambda a.a$ at type $\tau\vee\sigma\to\tau\vee\sigma$, where both τ and σ are atomic types. We suppose that $\lambda a.a$ has been evaluated to the identity $\mathrm{id}:=\alpha\mapsto\alpha$ in the domain of interpretation D (recall the evaluation function $[\![-]\!]:\Lambda\to D$ of Section 2.1), although we have not specified anything about D. A precise specification of it is the subject of Sections 3.3 and 3.4.

We first remark that, for a fresh variable a, we have

$$\begin{aligned}
\uparrow^{\tau\vee\sigma} a &= \mathcal{S}\kappa.\mathsf{case}\ a\ \mathsf{of}\ \left(a_1.\#\kappa\cdot(\mathrm{inl}\cdot\uparrow^{\tau}a_1)\|a_2.\#\kappa\cdot(\mathrm{inr}\cdot\uparrow^{\sigma}a_2)\right)\\
&= \mathcal{S}\kappa.\mathsf{case}\ a\ \mathsf{of}\ \left(a_1.\#\kappa\cdot(\mathrm{inl}\cdot a_1)\|a_2.\#\kappa\cdot(\mathrm{inr}\cdot a_2)\right).
\end{aligned}$$

Now,

$$\begin{aligned}
\downarrow^{\tau\vee\sigma\to\tau\vee\sigma} \mathrm{id} &= \lambda a.\#\downarrow^{\tau\vee\sigma}\left(\mathrm{id}\cdot\uparrow^{\tau\vee\sigma} a\right)\\
&= \lambda a.\#\downarrow^{\tau\vee\sigma}\left(\uparrow^{\tau\vee\sigma} a\right)\\
&= \lambda a.\#\begin{cases}\iota_1(\downarrow^{\tau} S') & \text{, if } \mathrm{inl}\cdot S' = \uparrow^{\tau\vee\sigma} a\\ \iota_2(\downarrow^{\sigma} S') & \text{, if } \mathrm{inr}\cdot S' = \uparrow^{\tau\vee\sigma} a\end{cases}\\
&= \lambda a.\#\mathsf{case}\ a\ \mathsf{of}\ \left(a_1.\#\iota_1(\downarrow^{\tau} a_1)\,\|\,a_2.\#\iota_2(\downarrow^{\sigma} a_2)\right)\\
&= \lambda a.\#\mathsf{case}\ a\ \mathsf{of}\ (a_1.\#\iota_1 a_1\,\|\,a_2.\#\iota_2 a_2)\\
&= \lambda a.\mathsf{case}\ a\ \mathsf{of}\ (a_1.\iota_1 a_1\,\|\,a_2.\iota_2 a_2),
\end{aligned}$$

where, after the step for shift, we implicitly used the equations

$$\iota_1(\downarrow^{\tau} a_1) = \begin{cases}\iota_1(\downarrow^{\tau} S') & \text{, if } \mathrm{inl}\cdot S' = \mathrm{inl}\cdot a_1\\ \iota_2(\downarrow^{\sigma} S') & \text{, if } \mathrm{inr}\cdot S' = \mathrm{inl}\cdot a_1\end{cases}$$

and

$$\iota_2(\downarrow^{\sigma} a_2) = \begin{cases}\iota_1(\downarrow^{\tau} S') & \text{, if } \mathrm{inl}\cdot S' = \mathrm{inr}\cdot a_2\\ \iota_2(\downarrow^{\sigma} S') & \text{, if } \mathrm{inr}\cdot S' = \mathrm{inr}\cdot a_2.\end{cases}$$

3.3. Completeness for Kripke-style models

There is a close connection between shift and reset, and the continuation-passing style (CPS) translations, that is, the double-negation translations [**143**]. In fact, shift and reset have been invented for the purpose of being able to program in direct style, what would normaly have to be programmed in continuation-passing style. Besides the original [**58, 59**], Section 4.2 and Appendix A are explicit about the connection.

In this section we present a notion of model that we developed when writing a continuation-passing style *proof* based on Danvy's direct-style *program*. We show that intuitionistic predicate logic (IQC) is sound and complete for that notion of model.

The reader will see that the introduced notion is very similar to CBV Kripke models for classical logic from Section 2.4. However, the similarity was not intended by design, it was a subsequent realisation that the two notions share the same structure.

3.3.1. Definition. An *intuitionistic Kripke-style model* is given by:

- a preorder $(K,\leq)$ of *possible worlds*;
- a binary relation on worlds $(-)\Vdash_{\bot}^{(-)}$ labelling a world as *exploding*;
- a binary relation $(-)\Vdash_s(-)$ of *strong forcing* between worlds and atomic formulae, such that

$$\text{for all } w'\geq w,\ w\Vdash_s X\to w'\Vdash_s X,$$

$$w\Vdash_s \top \text{ is true,}$$

$$w\Vdash_s \bot \text{ iff } \forall C.\, w\Vdash_{\bot}^{C};$$

- and a domain of quantification $D(w)$ for each world w, such that

$$\text{for all } w'\geq w, D(w)\subseteq D(w').$$

The relation $(-)\Vdash_s(-)$ of *strong forcing* is *extended from atomic to composite formulae* inductively and by simultaneously defining one new relation, (non-strong) forcing:

⋆ A formula A is *forced* in the world w (notation $w \Vdash A$) if, for any formula C,

$$\forall w' \geq w.\ \left(\forall w'' \geq w'.\ w'' \Vdash_s A \to w'' \Vdash^C_\bot\right) \to w' \Vdash^C_\bot;$$

- $w \Vdash_s A \wedge B$ if $w \Vdash A$ and $w \Vdash B$;
- $w \Vdash_s A \vee B$ if $w \Vdash A$ or $w \Vdash B$;
- $w \Vdash_s A \Rightarrow B$ if for all $w' \geq w$, $w \Vdash A$ implies $w \Vdash B$;
- $w \Vdash_s \forall x.A(x)$ if for all $w' \geq w$ and all $t \in D(w')$, $w' \Vdash A(t)$;
- $w \Vdash_s \exists x.A(x)$ if $w \Vdash A(t)$ for some $t \in D(w)$.

3.3.2. Remark. The differences with Definition 2.4.1 are framed. We can also present the classical (non-strong) forcing relation using binary exploding nodes, by:

$$\forall w' \geq w.\ \left(\forall w'' \geq w'.\ w'' \Vdash_s A \to \forall I.w'' \Vdash^I_\bot\right) \to \forall O.w' \Vdash^O_\bot;$$

The difference between forcing in intuitionistic and classical Kripke-style models is, then, that both have the form of a continuations monad (see Lemma 3.3.4), but while for intuitionistic forcing the answer type of the monad and of the inside continuation is the same, C, in classical forcing we are allowed to change the answer type of the continuation, I, when computing the answer type of the monad, O.

3.3.3. Lemma. *Strong forcing and (non-strong) forcing are monotone in any Kripke-style model.*

PROOF. Monotonicity of strong forcing is proved by induction on the complexity of the formula, while that of forcing is by definition. The proof is easy and available in the Coq formalisation. □

3.3.4. Lemma. *The following monadic operations are definable for Kripke-style models:*

"unit" $\eta(\cdot)$**:** $w \Vdash_s A \to w \Vdash A$
"bind" $(\cdot)^*(\cdot)$**:** $(\forall w' \geq w.\ w' \Vdash_s A \to w' \Vdash B) \to w \Vdash A \to w \Vdash B$

PROOF. The proof for unit is obvious. Let

$$\forall w' \geq w.\ w' \Vdash_s A \to w' \Vdash B, \tag{3}$$

let $w \Vdash A$, and let a formula C and a world $w' \geq w$ be given s.t.

$$\forall w'' \geq w'.\ w'' \Vdash_s B \to w'' \Vdash^C_\bot. \tag{4}$$

To prove $w' \Vdash^C_\bot$ we apply $w \Vdash A$ and then, given $w'' \geq w'$ and $w'' \Vdash_s A$, we have to prove $w'' \Vdash^C_\bot$. This follows from Lemma 3.3.3 by combining (3) and (4).

If we leave implicit the handling of formulae C, worlds, and monotonicity, we have the following procedures behind the proofs.

$$\eta(\alpha) = \kappa \mapsto \kappa\alpha$$
$$(\phi)^*(\alpha) = \kappa \mapsto \alpha(\beta \mapsto \phi\beta\kappa)$$

We did not mark explicitly meta-level application by "·" here, like we did in Section 2.5, because there is no risk of confusion, for no object-language application has been introduced yet. □

$$\frac{(a:A)\in\Gamma}{\Gamma\vdash a:A}\ \text{Ax} \qquad \frac{\Gamma\vdash \mathsf{ff}:\bot}{\Gamma\vdash a:A}\ \bot_E$$

$$\frac{\Gamma\vdash p:A_1 \qquad \Gamma\vdash q:A_2}{\Gamma\vdash (p,q):A_1\wedge A_2}\ \wedge_I \qquad \frac{\Gamma\vdash p:A_1\wedge A_2}{\Gamma\vdash \pi_i p:A_i}\ \wedge^i_E$$

$$\frac{\Gamma\vdash p:A_i}{\Gamma\vdash \iota_i p:A_1\vee A_2}\ \vee^i_I$$

$$\frac{\Gamma\vdash p:A_1\vee A_2 \qquad \Gamma,a_1:A_1\vdash q_1:C \qquad \Gamma,a_2:A_2\vdash q_2:C}{\Gamma\vdash \mathsf{case}\ p\ \mathsf{of}\ \left(a_1.q_1\|a_2.q_2\right):C}\ \vee_E$$

$$\frac{\Gamma,a:A_1\vdash p:A_2}{\Gamma\vdash \lambda a.p:A_1\Rightarrow A_2}\ \Rightarrow_I \qquad \frac{\Gamma\vdash p:A_1\Rightarrow A_2 \qquad \Gamma\vdash q:A_1}{\Gamma\vdash pq:A_2}\ \Rightarrow_E$$

$$\frac{\Gamma\vdash p:A(x) \qquad x\text{-fresh}}{\Gamma\vdash \lambda x.p:\forall x.A(x)}\ \forall_I \qquad \frac{\Gamma\vdash p:\forall x.A(x)}{\Gamma\vdash pt:A(t)}\ \forall_E$$

$$\frac{\Gamma\vdash p:A(t)}{\Gamma\vdash (t,p):\exists x.A(x)}\ \exists_I$$

$$\frac{\Gamma\vdash p:\exists x.A(x) \qquad \Gamma,a:A(x)\vdash q:C \qquad x\text{-fresh}}{\Gamma\vdash \mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q:C}\ \exists_E$$

Table 1: Proof term annotation for the natural deduction system of IQC

With Table 1, we fix a derivation system for IQC. We use the same convention for names of variables and terms as in Definition 2.2.1. We also give the syntax of raw λ-terms via the following inductive definitions, in order to supplement the characterisation of normal and neutral terms from page 63.

$$\begin{aligned}
\Lambda \ni p,q \quad &::= a \mid \lambda a.p \mid pq \mid \mathsf{case}\ p\ \mathsf{of}\ \left(a_1.q_1\|a_2.q_2\right) \mid (p,q) \mid \pi_1 p \mid \pi_2 p \mid \\
&\quad \lambda x.p \mid pt \mid (t,p) \mid \mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q \\
\Lambda\text{-nf} \ni r \quad &::= e \mid \lambda a.r \mid \iota_1 r \mid \iota_2 r \mid (r_1,r_2) \mid \lambda x.r \mid (t,r) \\
\Lambda\text{-ne} \ni e \quad &::= a \mid er \mid \mathsf{case}\ e\ \mathsf{of}\ (a_1.r_1\|a_2.r_2) \mid \pi_1 e \mid \pi_2 e \mid et \mid \\
&\quad \mathsf{dest}\ e\ \mathsf{as}\ (x.a)\ \mathsf{in}\ r
\end{aligned}$$

As before, let $w\Vdash\Gamma$ denote that all formulae from Γ are forced.

3.3.5. Theorem (Soundness)**.** *If $\Gamma\vdash p:A$, then, in any world w of any Kripke-style model, if $w\Vdash\Gamma$, then $w\Vdash A$.*

PROOF. This is proved by a simple induction on the derivation. We give the algorithm behind it in section 3.4. □

3.3.6. Remark. The condition "for all formula C" in Definition 3.3.1 is only necessary for the soundness proof to go through, more precisely, the cases of elimination rules for $\vee$ and $\Rightarrow$. The completeness proof goes through even if we define forcing by

$$\forall w'\geq w.\ \left(\forall w''\geq w'.\ w''\Vdash_s A\to w''\Vdash^A_\bot\right)\to w'\Vdash^A_\bot.$$

3.3.7. Definition. The *Universal Kripke-style model $\mathscr{U}$* is obtained by setting:

- K to be the set of contexts Γ of IQC;
- $\Gamma \leq \Gamma'$ iff $\Gamma \subseteq \Gamma'$;
- $\Gamma \Vdash_s X$ iff there is a derivation in normal form of $\Gamma \vdash X$ in IQC;
- $\Gamma \Vdash_\bot^C$ iff there is a derivation in normal form of $\Gamma \vdash C$ in IQC;
- for any w, $D(w)$ is a set of individuals for IQC (that is, $D(-)$ is a constant function from worlds to sets of individuals).

$(-) \Vdash_s (-)$ is monotone because of the weakining property for intuitionistic "$\vdash$".

3.3.8. Remark. The difference between strong forcing and the exploding node predicate in $\mathscr{U}$ is that the former is defined on atomic formulae, while the latter is defined on any kind of formulae.

3.3.9. Lemma. *We can also define the monadic* "run" *operation on the universal model* $\mathscr{U}$, for atomic formulae A*:*

$$\mu(\cdot) : w \Vdash A \to w \Vdash_s A.$$

PROOF. Trivial, by setting $C := A$. □

3.3.10. Theorem (Cut-Free Completeness for $\mathscr{U}$)**.** *For any closed formula* A *and closed context* Γ*, the following hold for* $\mathscr{U}$*:*

$$(\downarrow) \qquad \Gamma \Vdash A \longrightarrow \{p \mid \Gamma \vdash p : A\} \qquad (\textit{"reify"})$$

$$(\uparrow) \qquad \Gamma \vdash e : A \longrightarrow \Gamma \Vdash A \qquad (\textit{"reflect"})$$

Moreover, the target of $(\downarrow)$ *is a normal term, while the source of* $(\uparrow)$ *is a neutral term.*

PROOF. We will once again skip writing the proof term annotations in order to decrease the level of detail. The algorithm behind this proof that concentrates on proof terms is given in Section 3.4.

Base case. $(\downarrow)$ is by "run" (Lemma 3.3.9), $(\uparrow)$ is by "unit" (Lemma 3.3.4).

Induction case for $\wedge$. Let $\Gamma \Vdash A \wedge B$ i.e.

$$\forall C.\ \forall \Gamma' \geq \Gamma.\ \big((\forall \Gamma'' \geq \Gamma'.\ \Gamma'' \Vdash A \text{ and } \Gamma'' \Vdash B \to \Gamma'' \vdash C) \to \Gamma' \vdash C\big).$$

We apply this hypothesis by setting $C := A \wedge B$ and $\Gamma' := \Gamma$, and then, given $\Gamma'' \geq \Gamma$ s.t. $\Gamma'' \Vdash A$ and $\Gamma'' \Vdash B$, we have to derive $\Gamma'' \vdash A \wedge B$. But, this is immediate by applying the $\wedge_I$ rule and the induction hypothesis $(\downarrow)$ twice, for A and for B.

Let $\Gamma \vdash A \wedge B$ be a neutral derivation. We prove $\Gamma \Vdash A \wedge B$ by applying unit (Lemma 3.3.4), and then applying the induction hypothesis $(\downarrow)$ on $\wedge_I^1$, $\wedge_I^2$, and the hypothesis.

Induction case for $\vee$. Let $\Gamma \Vdash A \vee B$ i.e.

$$\forall C.\ \forall \Gamma' \geq \Gamma.\ \big((\forall \Gamma'' \geq \Gamma'.\ \Gamma'' \Vdash A \text{ or } \Gamma'' \Vdash B \to \Gamma'' \vdash C) \to \Gamma' \vdash C\big).$$

We apply this hypothesis by setting $C := A \vee B$ and $\Gamma' := \Gamma$, and then, given $\Gamma'' \geq \Gamma$ s.t. $\Gamma'' \Vdash A$ or $\Gamma'' \Vdash B$, we have to derive $\Gamma'' \vdash A \vee B$. But, this is immediate, after a case distinction, by applying the $\vee_I^i$ rule and the induction hypothesis $(\downarrow)$.

We now consider the only case (besides $\uparrow^{\exists x A(x)}$ below) where using shift and reset, or our Kripke-style models, is crucial. Let $\Gamma \vdash A \vee B$ be a neutral derivation. Let a formula C and $\Gamma' \geq \Gamma$ be given, and let

$$(\#) \qquad \forall \Gamma'' \geq \Gamma'.\ \big(\Gamma'' \Vdash A \text{ or } \Gamma'' \Vdash B \to \Gamma'' \vdash C\big).$$

We prove $\Gamma' \vdash C$ by the following derivation tree:

$$\dfrac{\dfrac{\Gamma \vdash A \vee B}{\Gamma' \vdash A \vee B} \qquad \dfrac{\dfrac{\dfrac{\dfrac{A \in A, \Gamma'}{A, \Gamma' \vdash A}\,\text{Ax}}{A, \Gamma' \Vdash A}\,(\uparrow)}{A, \Gamma' \Vdash A \text{ or } A, \Gamma' \Vdash B}\,\text{inl} \qquad \dfrac{\dfrac{\dfrac{B \in B, \Gamma'}{B, \Gamma' \vdash B}\,\text{Ax}}{B, \Gamma' \Vdash B}\,(\uparrow)}{B, \Gamma' \Vdash A \text{ or } B, \Gamma' \Vdash B}\,\text{inr}}{A, \Gamma' \vdash C}\,(\#) \qquad \dfrac{}{B, \Gamma' \vdash C}\,(\#)}{\Gamma' \vdash C}\,\vee_E$$

Induction case for $\Rightarrow$. Let $\Gamma \Vdash A \Rightarrow B$ i.e.

$$\forall C.\, \forall \Gamma' \geq \Gamma.\, \big((\forall \Gamma'' \geq \Gamma'.\, (\forall \Gamma_3 \geq \Gamma''.\, \Gamma_3 \Vdash A \to \Gamma_3 \Vdash B) \to \Gamma'' \vdash C) \to \Gamma' \vdash C\big).$$

We apply this hypothesis by setting $C := A \Rightarrow B$ and $\Gamma' := \Gamma$, and then, given $\Gamma'' \geq \Gamma$ s.t.

$$(\#) \qquad \forall \Gamma_3 \geq \Gamma''.\, \Gamma_3 \Vdash A \to \Gamma_3 \Vdash B$$

we have to derive $\Gamma'' \vdash A \Rightarrow B$. This follows by applying $(\Rightarrow_I)$, the IH for$(\downarrow)$, then $(\#)$, and finally the IH for $(\uparrow)$ with the Ax rule.

Let $\Gamma \vdash A \Rightarrow B$ be a neutral derivation. We prove $\Gamma \Vdash A \Rightarrow B$ by applying unit (Lemma 3.3.4), and then, given $\Gamma' \geq \Gamma$ and $\Gamma' \Vdash A$, we have to show that $\Gamma' \Vdash B$. This is done by applying the IH for $(\uparrow)$ on the $(\Rightarrow_E)$ rule, with the IH for $(\downarrow)$ applied to $\Gamma' \Vdash A$.

Induction case for $\forall$. We recall that the domain function $D(-)$ is constant in the universal model $\mathscr{U}$. Let $\Gamma \Vdash \forall x A(x)$ i.e.

$$\forall C.\, \forall \Gamma' \geq \Gamma.\, \big((\forall \Gamma'' \geq \Gamma'.\, (\forall \Gamma_3 \geq \Gamma''.\, \forall t \in D.\, \Gamma_3 \Vdash A(t)) \to \Gamma'' \vdash C) \to \Gamma' \vdash C\big).$$

We apply this hypothesis by setting $C := \forall x A(x)$ and $\Gamma' := \Gamma$, and then, given $\Gamma'' \geq \Gamma$ s.t.

$$(\#) \qquad \forall \Gamma_3 \geq \Gamma''.\, \forall t \in D.\, \Gamma_3 \Vdash A(t)$$

we have to derive $\Gamma'' \vdash \forall x A(x)$. This follows by applying $(\forall_I)$, the IH for$(\downarrow)$, and then $(\#)$.

Let $\Gamma \vdash \forall x A(x)$ be a neutral derivation. We prove $\Gamma \Vdash \forall x A(x)$ by applying unit (Lemma 3.3.4), and then, given $\Gamma' \geq \Gamma$ and $t \in D$, we have to show that $\Gamma' \Vdash A(t)$. This is done by applying the IH for $(\uparrow)$ on the $(\forall_E)$ rule and the hypothesis $\Gamma \vdash \forall x A(x)$.

Induction case for $\exists$. Let $\Gamma \Vdash \exists x A(x)$ i.e.

$$\forall C.\, \forall \Gamma' \geq \Gamma.\, \big((\forall \Gamma'' \geq \Gamma'.\, (\exists t \in D.\, \Gamma'' \Vdash A(t)) \to \Gamma'' \vdash C) \to \Gamma' \vdash C\big).$$

We apply this hypothesis by setting $C := \exists x A(x)$ and $\Gamma' := \Gamma$, and then, given $\Gamma'' \geq \Gamma$ s.t. $\exists t \in D.\, \Gamma'' \Vdash A(t)$, we have to derive $\Gamma'' \vdash \exists x A(x)$. This follows by applying $(\exists_I)$ with $t \in D$, and the IH for$(\downarrow)$.

Let $\Gamma \vdash \exists x A(x)$ be a neutral derivation. Let a formula C and $\Gamma' \geq \Gamma$ be given, and let

$$(\#) \qquad \forall \Gamma'' \geq \Gamma'.\, \big(\exists t \in D. \Gamma'' \Vdash A(t) \to \Gamma'' \vdash C\big).$$

We prove $\Gamma' \vdash C$ by the following derivation tree:

$$\dfrac{\dfrac{\Gamma \vdash \exists x A(x)}{\Gamma' \vdash \exists x A(x)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{A(x) \in A(x), \Gamma'}{A(x), \Gamma' \vdash A(x)}\,\text{Ax}}{A(x), \Gamma' \Vdash A(x)}\,(\uparrow)}{A(x), \Gamma' \vdash C}\,(\#) \qquad x\text{-fresh}}{\Gamma' \vdash C}\,\exists_E$$

The output is in normal form. By inspection of the proof. □

3.3.1. A variant without the preorder $\leq$. We end this section by mentioning that a variant of Kripke-style models is possible where the preorder on the set of possible worlds is not primitive, but defined. Namely, using the exploding nodes predicate, we can define

$$w \leq w' := \forall C.\ w \Vdash^C_\bot \to w' \Vdash^C_\bot .$$

It is easy to see that this is a preorder.

In the universal model $\mathscr{U}$, the definition amounts to the implication

$$\big(\forall C.\, \Gamma \vdash C \to \Gamma' \vdash C\big) \to \Gamma \subseteq \Gamma'.$$

3.4. Computational content

By reflection inside the Coq proof assistant, we can check that the computational content of the composition of Theorem 3.3.5 and Theorem 3.3.10 is proof normalisation. The reader can look by himself at the examples in the Coq formalisation. The reduction relation here is just the standard one for λ-calculus, unlike the one of Chapter 2.

Also, unlike in Chapter 2, the question of whether we obtained call-by-value or call-by-name evaluation strategy does not make much sense, since in the intuitionistic case, a well-typed λ-term reduces to a unique normal form, regardless of the reduction strategy.

We again give hand-extracted versions of the algorithms behind the proofs, that deliberately avoid manipulating worlds and universally quantified formula C, in order to make the computational content clearer.

The following is the evaluation procedure, that is, the algorithm behind the proof of Soundness. We skip using "$\cdot$" for meta-level application, because there is no risk of confusion.

$$[\![\Gamma \vdash p : A]\!] : w \Vdash \Gamma \to w \Vdash A$$

$$\begin{aligned}
[\![(p,q)]\!] &:= \rho \mapsto \eta\big([\![p]\!]\rho, [\![q]\!]\rho\big)\\
[\![\pi_1 p]\!] &:= \rho \mapsto \kappa \mapsto [\![p]\!]\rho\,\big(\gamma \mapsto \kappa(\mathrm{fst}\,\gamma)\big)\\
[\![\pi_2 p]\!] &:= \rho \mapsto \kappa \mapsto [\![p]\!]\rho\,\big(\gamma \mapsto \kappa(\mathrm{snd}\,\gamma)\big)\\
[\![\iota_1 p]\!] &:= \rho \mapsto \eta\big(\mathrm{inl}\,([\![p]\!]\rho)\big)\\
[\![\iota_2 p]\!] &:= \rho \mapsto \eta\big(\mathrm{inr}\,([\![p]\!]\rho)\big)\\
[\![\mathsf{case}\ p\ \mathsf{of}\ (a_1.q_1 \| a_2.q_2)]\!] &:= \rho \mapsto \kappa \mapsto [\![p]\!]\rho\left(\gamma \mapsto \begin{cases} [\![q_1]\!](\rho,\alpha)\kappa & \text{if } \gamma = \mathrm{inl}\,\alpha \\ [\![q_2]\!](\rho,\beta)\kappa & \text{if } \gamma = \mathrm{inr}\,\beta \end{cases}\right)\\
[\![\lambda a.p]\!] &:= \rho \mapsto \eta\big(\alpha \mapsto [\![p]\!](\rho,\alpha)\big)\\
[\![pq]\!] &:= \rho \mapsto \kappa \mapsto [\![p]\!]\rho\,\big(\mathrm{ret}\,([\![q]\!]\kappa)\big)
\end{aligned}$$

The following is the algorithm behind the completeness proof. Here we use "$\cdot$" for meta-level application and no symbol for object-level application.

$$\downarrow^A : \Gamma \Vdash A \to \{p \in \Lambda\text{-nf} \mid \Gamma \vdash p : A\}$$
$$\uparrow^A : \{e \in \Lambda\text{-ne} \mid \Gamma \vdash e : A\} \to \Gamma \Vdash A$$

$$\downarrow^X := \alpha \mapsto \mu \cdot \alpha \qquad X\text{-atomic}$$
$$\uparrow^X := e \mapsto \eta \cdot e \qquad X\text{-atomic}$$
$$\downarrow^{A \wedge B} := \eta \cdot \left(\gamma \mapsto \left(\downarrow^A (\mathrm{fst} \cdot \gamma), \downarrow^B (\mathrm{snd} \cdot \gamma)\right)\right)$$
$$\uparrow^{A \wedge B} := e \mapsto \eta\left(\uparrow^A \pi_1 e, \uparrow^B \pi_2 e\right)$$
$$\downarrow^{A \vee B} := \eta \cdot \left(\gamma \mapsto \begin{cases} \downarrow^A \alpha & \text{if } \gamma = \mathrm{inl} \cdot \alpha \\ \downarrow^B \beta & \text{if } \gamma = \mathrm{inr} \cdot \beta \end{cases}\right)$$
$$\uparrow^{A \vee B} := e \mapsto \kappa \mapsto \mathsf{case}\ e\ \mathsf{of}\ \left(a_1.\kappa \cdot \left(\mathrm{inl} \cdot \uparrow^A a_1\right) \| a_2.\kappa \cdot \left(\mathrm{inr} \cdot \uparrow^B a_2\right)\right)$$
$$\downarrow^{A \Rightarrow B} := \eta \cdot \left(\phi \mapsto \lambda a.\, \downarrow^B \left(\phi \cdot \uparrow^A a\right)\right)$$
$$\uparrow^{A \Rightarrow B} := e \mapsto \eta \cdot \left(\alpha \mapsto \uparrow^B \left(e\left(\downarrow^A \alpha\right)\right)\right)$$

3.5. Aspects of the Coq formalisation

The formalisation is available at the address `http://www.lix.polytechnique.fr/~danko/code`.

The fragment without quantifiers was formalised independently of the formalisations of Chapter 2. For the version with quantifiers, once we realised that the intuitionistic and classical CBV Kripke-style models share the same structure, we developed it in parallel to the classical one. Thus, we have nothing to add that was not remarked already in Section 2.6. The sole difference is that in this chapter we rely on a natural deduction system, while in Chapter 2 we relied on a sequent calculus.

We did not include the $\wedge$ connective in the formalisation, because we knew that it would not be problematic to add it later.

3.6. Related and future work

3.6.1. The problem of canonical η-long normal form for sums. Although Danvy's program, and our proof, do produce η-long normal forms, these normal forms are not canonical.

Consider, for example, the proof term $\lambda a.a$, normalised for the formula $(A \vee B \Rightarrow C) \Rightarrow A \vee B \Rightarrow C$, (where A, B, C are atomic formulae)

$$\downarrow^{(A \vee B \Rightarrow C) \Rightarrow A \vee B \Rightarrow C} \lambda a.a = \lambda a.\lambda b.a\,(\mathsf{case}\ b\ \mathsf{of}\ (c_1.\iota_1 c_1 \| c_2.\iota_2 c_2))\,.$$

We should expect that the proof term

$$\lambda a.\lambda b.\mathsf{case}\ b\ \mathsf{of}\ (c_1.a(\iota_1 c_1) \| c_2.a(\iota_2 c_2))\,,$$

when normalised for the same formula, should be identified to the result of the first normalisation above, since they denote essentially the same derivation.

However, we get as result the starting term itself:

$$\downarrow^{(A\vee B\Rightarrow C)\Rightarrow A\vee B\Rightarrow C} \lambda a.\lambda b.\mathsf{case}\ b\ \mathsf{of}\ (c_1.a(\iota_1 c_1)\,\|\,c_2.a(\iota_2 c_2)) = \\ = \lambda a.\lambda b.\mathsf{case}\ b\ \mathsf{of}\ (c_1.a(\iota_1 c_1)\,\|\,c_2.a(\iota_2 c_2))\,.$$

This may be seen as a defect, and we might want to add to Danvy's program, or to our proof, a rewrite rule which would produce the same result on the two different inputs above. While that would work for this concrete example, adding just one more rewrite rule would not solve the problem in general.

Actually, the problem is related to Tarski's High-School Algebra Problem. Tarski asked [**44**] whether any true formula[8] of the structure $(\mathbb{N},+,\cdot,(-)^{(-)},1)$, that is, number theory with addition, multiplication, exponentiation and 1, can be deduced only on the basis of the eleven arithmetic identities taught in high-school:

$$\begin{array}{ccc} 1\cdot x = x & x\cdot y = y\cdot x & (x\cdot y)\cdot z = x\cdot(y\cdot z) \\ x^1 = x & 1^x = 1 & x^{y\cdot z} = (x^y)^z \\ (x\cdot y)^z = x^z\cdot y^z & x+y = y+x & (x+y)+z = x+(y+z) \\ x\cdot(y+z) = x\cdot y + x\cdot z & x^{(y+z)} = x^y\cdot x^z & \end{array}$$

While for the fragments $(\mathbb{N},+,\cdot,1)$ and $(\mathbb{N},\cdot,(-)^{(-)},1)$ a corresponding fragment of the above identities suffices, Wilkie showed in [**184**] that there exists a true equation which is not derivable on the bases of the above identities alone. Furthermore, Gurevič [**93**] showed that no finite number of identities added to the above ones would suffice, by giving an infinite number of equations resembling Wilkie's one.

This problem has a direct connection to intuitionistic logic, by its connection to typed λ-calculi, made precise in [**76**]. There, Fiore, Di Cosmo, and Balat, show that an infinite sequence of type isomorphisms analogous to Gurevič's equations exists, and that, therefore, the notion of canonical normal form for λ-calculus with sums is not finitely axiomatisable.

Although, as a consequence of their work, giving such a canonical normal form does not seem easy, Balat, Di Cosmo, and Fiore, propose, for practical purposes, to narrow down the notion of normal form arising from Danvy's program. In [**22, 19, 20**], they propose three kinds of optimisations to expand the equivalence classes of normal terms, and implement them [**21**] using delimited control. For example, by using their TDPE algorithm, the two example terms this section was started with would be normalised to the same term.

In the future, we would like to investigate the problem of canonical normal forms in the context of the logical system proposed in Chapter 4.

3.6.2. Normalisation by evaluation for intuitionistic systems. We mentioned the semantic cut-elimination techniques of Girard, Okada and Sambin in Section 2.7, which also apply to the intuitionistic case.

In [**10**], Altenkirch, Dybjer, Hofmann, and Scott, give a categorical proof of NBE for a typed λ-calculus with sums, by constructing a sheaf model. The connecting between sheaves and Beth semantics is well known. While the proof is

[8] Recall the truth definition of page 12.

constructive, due to their use of topos theory, we could not see what the algorithm behind their proof is and how they manage to produce canonical normal forms in spite of their non finite-axiomatisability.

We remark that, for the fragment $\{\Rightarrow, \forall, \wedge\}$, NBE can also be seen as completeness for *Beth* semantics, since forcing in Beth and Kripke models is the same thing on that fragment.

The NBE method has been applied to higher order and dependently typed intuitionistic systems, in the works of Abel, Aehlig, Coquand, and Dybjer [**6, 7, 9, 3, 5**].

3.6.3. Macedonio-Sambin's models. In [**133**], Macedonio and Sambin present a notion of model for extensions of Basic logic (a sub-structural logic that is more basic than Linear logic), which, for intuitionistic logic, appears to be related to our notion of model. However, they insist that their "set of worlds" must be saturated, while we do not. Also, Sambin and Macedonio remark [**134**] that their notion of model is neither Beth nor Kripke's one.

3.6.4. Using call/cc or exceptions instead of delimited control. The continuation-passing style proof of completeness (given as program in Section 3.4) could not be written in direct style by using the call/cc control operator. This is so, because the calls to the continuations in the case of sum-reflection are not tail calls, while it is well known that the CPS translation of a program that uses call/cc always produces a program where continuations only perform tail calls.

Nevertheless, in [**25**], Barral gives a program for NBE of λ-calculus with sums by just using the exceptions mechanism of a programming language, which is something a priori strictly weaker than using delimited control.

3.6.5. Constructive completeness for standard Kripke models. In Subsection 4.5.11, we discuss the possibility of a completeness proof w.r.t. *standard* Kripke models, in the context of the logical system introduced in Chapter 4.

CHAPTER 4

Extension of intuitionistic logic with delimited control

One starting point for the work presented in this chapter was the search for a more direct way of formulating the intuitionistic completeness proof of Chapter 3, that is, the quest for logical understanding of delimited control operators. Soon after the completeness proof of Chapter 3 was formalised, Herbelin observed that:

(1) If one restricts the computational power of delimited control to account for computational effects corresponding to programming language exceptions, one can build a simple intuitionistic logic that is capable of deriving Markov's Principle;
(2) Using the full power of delimited control, one is able to derive the Double Negation Shift schema.

The first point was worked out by Herbelin in [**100**]. In this chapter, we propose a logical system for addressing the second point. This is, of course, not the first attempt to extend intuitionistic logic with control operators. Since Griffin's observation [**92**] that the control operator call/cc can be assigned a type expressing Peirce's law, there has been much research on the computational content of classical logic. One particularly far reaching approach is Krivine's [**126, 127, 128**]. However, from the perspective of the Curry-Howard isomorphism, such approaches suffer a defect of using classical logic for program specification: for example, a proof of an existential statement does not necessarily contain the witness of existence, therefore one has to rely on meta-theorems in order to know that a witness will eventually (if ever) be computed from the proof.

We have introduced, from the computational perspective, the shift/reset delimited control operators in Chapter 3. From the logical perspective, in the context of proof terms, we see delimited control as means of being able to access *a certain part* of the surroundings of a proof term from inside the proof term itself, as opposed to non-delimited control (essentially the call/cc operator) where the proof term can only access the entire surroundings indiscriminately, which amounts computationally to aborting the entire computation and waiting to be restarted by a meta-level interpreter. The part of the surrounding that we want to be able to access will be defined as a "pure evaluation context" in Section 4.1; logically, it is the surroundings of a proof term for a $\{\Rightarrow, \forall\}$-free formula, which is the predicate logic equivalent of arithmetic Σ^0_1-formulae, for which we know that classical and intuitionistic provability coincide. In other words, we will propose a proof term calculus for a logic which is essentially intuitionistic, except that at the fragment Σ^0_1 we are allowed to use classical reasoning to obtain more efficient proofs.

The system proposed in this chapter does not come out of the blue. In fact, we think of this work as a contribution to merging two very active but distinct

research directions: the search for logical meaning of computational effects (or delimited control) from Semantics of Programming Languages and the realisability interpretations related to bar recursion from Mathematical Logic.

In the next section, Section 4.1, we will introduce the system MQC$^+$. The acronym comes from Troelstra: IQC is intuitionistic predicate logic, MQC is minimal predicate logic (IQC without the $\bot_E$ rule), and CQC is classical predicate logic.

In Section 4.2, we will give an elementary proof of equiconsistency with MQC, and further comparisons with provability in CQC and MQC. In particular, we will show that an extension to predicate logic of Glivenko's theorem holds for our system, unlike for MQC.

In Section 4.3, we will use the reduction relation on proof terms, to prove that: 1) reduction between proof terms does not change their logical meaning (Subject Reduction); 2) if a proof term is not in final form (a value), it will further reduce (Progress).

In Section 4.4, we will prove that every reduction sequence of proof terms is finite and ends with a value (Normalisation), and deduce that the disjunction and existence properties hold for MQC$^+$.

In the final Section 4.5, we will discuss related work and give a couple of possible directions for future work.

4.1. The system MQC$^+$

The natural deduction system of MQC$^+$ is shown in Table 1. The derivations can be annotated by a $\{\Rightarrow,\forall\}$-free formula T, but, at the moment[1], *at most one* such formula is allowed globally.[2] The symbol $\diamond$ is used in the annotations as a wild-card, to mean that there either is an annotating formula T, or there is none. In the proof rules where the wild-card appears both above and below the line, it means that either there is an annotation both above and below, or there is no annotation above and no annotation below. Following Ulrich Berger [**31**], we call the $\{\Rightarrow,\forall\}$-free formulae, Σ-formulae, and denote them by S,T,U, while general formulae are denoted by A,B,C.

The proof rules of MQC$^+$ are the proof rules of MQC, plus two new rules, ($\mathscr{S}$) and (#). The intuitionistic rules neither use nor set the annotating Σ-formula. The rule (#) can only be applied when the conclusion is a Σ-formula; since those formulae, loosely speaking, correspond to the Σ^0_1-formulae of the arithmetical hierarchy, the (#) rules envelop a piece of "classical" reasoning from which witnesses for disjunction and existential quantification can be "extracted". To use another metaphor, the (#) rule sets a "control delimiter" (for the Σ-formula T being set for the annotation) above which we can perform computational effects as long as the result of the entire computation is the data-type, or Σ-formula, T. The rule ($\mathscr{S}$) can be used only in a sub-derivation of a (#) derivation, that is, only above an already set control delimiter. The role of ($\mathscr{S}$) is to "escape" to the nearest enclosing control delimiter once an intuitionistic witness for the Σ-formula from the annotation has been found. Multiple uses of (#) and ($\mathscr{S}$) are allowed in a derivation, but due to current restrictions to only one global annotating Σ-formula, the uses of (#) and ($\mathscr{S}$) must be on the same annotating formula. In

[1] See Subsection 4.5.10 for a discussion.

[2] Were we in IQC, a natural choice for the global formula would be $\bot$.

$$\frac{A \in \Gamma}{\Gamma \vdash_\diamond A}\ \mathrm{Ax}$$

$$\frac{\Gamma \vdash_\diamond A_1 \qquad \Gamma \vdash_\diamond A_2}{\Gamma \vdash_\diamond A_1 \wedge A_2}\ \wedge_I \qquad\qquad \frac{\Gamma \vdash_\diamond A_1 \wedge A_2}{\Gamma \vdash_\diamond A_i}\ \wedge^i_E$$

$$\frac{\Gamma \vdash_\diamond A_i}{\Gamma \vdash_\diamond A_1 \vee A_2}\ \vee^i_I \qquad \frac{\Gamma \vdash_\diamond A_1 \vee A_2 \qquad \Gamma, A_1 \vdash_\diamond C \qquad \Gamma, A_2 \vdash_\diamond C}{\Gamma \vdash_\diamond C}\ \vee_E$$

$$\frac{\Gamma, A_1 \vdash_\diamond A_2}{\Gamma \vdash_\diamond A_1 \Rightarrow A_2}\ \Rightarrow_I \qquad\qquad \frac{\Gamma \vdash_\diamond A_1 \Rightarrow A_2 \qquad \Gamma \vdash_\diamond A_1}{\Gamma \vdash_\diamond A_2}\ \Rightarrow_E$$

$$\frac{\Gamma \vdash_\diamond A(x) \qquad x\text{-fresh}}{\Gamma \vdash_\diamond \forall x.A(x)}\ \forall_I \qquad\qquad \frac{\Gamma \vdash_\diamond \forall x.A(x)}{\Gamma \vdash_\diamond A(t)}\ \forall_E$$

$$\frac{\Gamma \vdash_\diamond A(t)}{\Gamma \vdash_\diamond \exists x.A(x)}\ \exists_I \qquad \frac{\Gamma \vdash_\diamond \exists x.A(x) \qquad \Gamma, A(x) \vdash_\diamond C \qquad x\text{-fresh}}{\Gamma \vdash_\diamond C}\ \exists_E$$

$$\frac{\Gamma \vdash_T T}{\Gamma \vdash_\diamond T}\ \#\ (\text{“reset”}) \qquad\qquad \frac{\Gamma, A \Rightarrow T \vdash_T T}{\Gamma \vdash_T A}\ \mathscr{S}\ (\text{“shift”})$$

Table 1: Natural deduction system of MQC$^+$

particular, when the (#) rule is applied with a conclusion which is annotated, the premise keeps the same annotation, and hence such a rule has no real use – the control delimited has already been set further down in the derivation.

As examples, we give the derivations for (generalisations of) the minimal-logic versions[3] of Markov's Principle,

$$(\mathrm{MP}_T) \qquad\qquad (T \Rightarrow S) \Rightarrow ((S \Rightarrow T) \Rightarrow T) \Rightarrow S,$$

and Double Negation Shift,

$$(\mathrm{DNS}_T) \qquad\qquad \forall x.((A(x) \Rightarrow T) \Rightarrow T) \Rightarrow (\forall x.A(x) \Rightarrow T) \Rightarrow T,$$

where, according to the already set convention, T and S are Σ-formulae, while $A(x)$ is a general one.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\cdots \vdash_S T \Rightarrow S}\ \mathrm{Ax} \qquad \dfrac{\dfrac{}{\cdots \vdash_S (S \Rightarrow T) \Rightarrow T}\ \mathrm{Ax} \qquad \dfrac{\dfrac{\dfrac{}{\cdots, S \vdash_S S}\ \mathscr{S}}{\cdots, S \vdash_S T}\ \Rightarrow_I}{\cdots \vdash_S S \Rightarrow T}\ \Rightarrow_I}{\cdots \vdash_S T}\ \Rightarrow_E}{T \Rightarrow S, (S \Rightarrow T) \Rightarrow T \vdash_S S}\ \Rightarrow_E}{T \Rightarrow S, (S \Rightarrow T) \Rightarrow T \vdash S}\ \#}{\vdash (T \Rightarrow S) \Rightarrow ((S \Rightarrow T) \Rightarrow T) \Rightarrow S}\ \Rightarrow_I$$

[3]The distinguished formula T plays the role of $\bot$ and the hypothesis $T \Rightarrow S$ plays the role of the ex-falso rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\cdots}}{\forall x.((A(x)\Rightarrow T)\Rightarrow T),\forall x.A(x)\Rightarrow T,A(x)\to T\vdash_T T}\ \forall_E,\Rightarrow_R,\text{ and Ax}}{\forall x.((A(x)\Rightarrow T)\Rightarrow T),\forall x.A(x)\Rightarrow T\vdash_T A(x)}\ \mathscr{S}}{\forall x.((A(x)\Rightarrow T)\Rightarrow T),\forall x.A(x)\Rightarrow T\vdash_T \forall x.A(x)}\ \forall_I,\ x\text{-fresh}}{\forall x.((A(x)\Rightarrow T)\Rightarrow T),\forall x.A(x)\Rightarrow T\vdash_T T}\ \Rightarrow_E}{\forall x.((A(x)\Rightarrow T)\Rightarrow T),\forall x.A(x)\Rightarrow T\vdash T}\ \#}{\forall x.((A(x)\Rightarrow T)\Rightarrow T)\vdash(\forall x.A(x)\Rightarrow T)\Rightarrow T}\ \Rightarrow_I}{\vdash\forall x.((A(x)\Rightarrow T)\Rightarrow T)\Rightarrow(\forall x.A(x)\Rightarrow T)\Rightarrow T}\ \Rightarrow_I$$

We will now define a calculus of proof term annotations for the natural deduction system of MQC$^+$, a version of lambda calculus with constants for handling the additional logical connectives and the delimited control operator, and then a reduction system for proof terms. All this is standard material, see for example the calculi of [**14, 155, 185**]. What is new is the connection to MQC and CQC, and the elementary proofs specific to our typing system.

$$\dfrac{(a:A)\in\Gamma}{\Gamma\vdash_\diamond a:A}\ \text{Ax}$$

$$\dfrac{\Gamma\vdash_\diamond p:A_1\qquad\Gamma\vdash_\diamond q:A_2}{\Gamma\vdash_\diamond(p,q):A_1\wedge A_2}\ \wedge_I\qquad\qquad\dfrac{\Gamma\vdash_\diamond p:A_1\wedge A_2}{\Gamma\vdash_\diamond\pi_i p:A_i}\ \wedge_E^i$$

$$\dfrac{\Gamma\vdash_\diamond p:A_i}{\Gamma\vdash_\diamond\iota_i p:A_1\vee A_2}\ \vee_I^i$$

$$\dfrac{\Gamma\vdash_\diamond p:A_1\vee A_2\qquad\Gamma,a_1:A_1\vdash_\diamond q_1:C\qquad\Gamma,a_2:A_2\vdash_\diamond q_2:C}{\Gamma\vdash_\diamond\mathsf{case}\ p\ \mathsf{of}\ \left(a_1.q_1\|a_2.q_2\right):C}\ \vee_E$$

$$\dfrac{\Gamma,a:A_1\vdash_\diamond p:A_2}{\Gamma\vdash_\diamond\lambda a.p:A_1\Rightarrow A_2}\ \Rightarrow_I\qquad\qquad\dfrac{\Gamma\vdash_\diamond p:A_1\Rightarrow A_2\qquad\Gamma\vdash_\diamond q:A_1}{\Gamma\vdash_\diamond pq:A_2}\ \Rightarrow_E$$

$$\dfrac{\Gamma\vdash_\diamond p:A(x)\qquad x\text{-fresh}}{\Gamma\vdash_\diamond\lambda x.p:\forall x.A(x)}\ \forall_I\qquad\qquad\dfrac{\Gamma\vdash_\diamond p:\forall x.A(x)}{\Gamma\vdash_\diamond pt:A(t)}\ \forall_E$$

$$\dfrac{\Gamma\vdash_\diamond p:A(t)}{\Gamma\vdash_\diamond(t,p):\exists x.A(x)}\ \exists_I$$

$$\dfrac{\Gamma\vdash_\diamond p:\exists x.A(x)\qquad\Gamma,a:A(x)\vdash_\diamond q:C\qquad x\text{-fresh}}{\Gamma\vdash_\diamond\mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q:C}\ \exists_E$$

$$\dfrac{\Gamma\vdash_T p:T}{\Gamma\vdash_\diamond\#p:T}\ \#\ (\text{“reset”})\qquad\qquad\dfrac{\Gamma,k:A\Rightarrow T\vdash_T p:T}{\Gamma\vdash_T\mathscr{S}k.p:A}\ \mathscr{S}\ (\text{“shift”})$$

Table 2: Proof term annotation for the natural deduction system of MQC$^+$

4.1.1. Definition. The set of *proof terms* is defined by the following grammar rules,

$$p,q,r ::= a \mid \iota_1 p \mid \iota_2 p \mid \mathsf{case}\ p\ \mathsf{of}\ \left(a.q\|b.r\right) \mid (p,q) \mid \pi_1 p \mid \pi_2 p \mid \lambda a.p \mid pq \mid \lambda x.p \mid pt \mid (t,p) \mid \mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q \mid \#p \mid \mathscr{S}k.p$$

where a, b, k, l denote hypotheses variables, x, y, z denote quantifier variables, and t, u, v denote quantifier terms (individuals); hence, $\lambda a.p$ is a constructor for implication, while $\lambda x.p$ is a constructor for universal quantification; (p, q) is a constructor for conjunction while (t, p) is a constructor for existential quantification, and pq is a destructor for implication while pt is a destructor for universal quantification.

Actually, we will call proof terms only those proof terms that can be given a valid derivation according to the rules of Table 2.

4.1.2. Remark. The $\mathscr{S}$ in $\mathscr{S}k.p$ is a binder, it binds k in p just as λ binds a in q in some $\lambda a.q$.

4.1.3. Definition. The subset of proof terms known as *values* is defined by:

$$V ::= a \mid \iota_1 V \mid \iota_2 V \mid (V, V) \mid (t, V) \mid \lambda a.p \mid \lambda x.p$$

4.1.4. Definition. The set of *pure evaluation contexts*, a subset of all proof terms with one placeholder or "hole", is defined by:

$$P ::= [\,] \mid \mathsf{case}\ P\ \mathsf{of}\ \left(a_1.p_1 \| a_2.p_2\right) \mid \pi_1 P \mid \pi_2 P \mid \mathsf{dest}\ P\ \mathsf{as}\ (x.a)\ \mathsf{in}\ p \mid Pq \mid (\lambda a.q)P \mid Pt \mid \iota_1 P \mid \iota_2 P \mid (P, p) \mid (V, P) \mid (t, P)$$

The association of proof terms to natural deduction derivations is given in Table 2. $P[p]$ denotes the proof term obtained from P by replacing its placeholder $[\,]$ with the proof term p.

In order to define a reduction relation on proof terms we also need the notion of (non-pure) *evaluation context*.

4.1.5. Definition. The set of *evaluation contexts* is given by the following grammar rule:

$$E ::= [\,] \mid \mathsf{case}\ E\ \mathsf{of}\ \left(a_1.p_1 \| a_2.p_2\right) \mid \pi_1 E \mid \pi_2 E \mid \mathsf{dest}\ E\ \mathsf{as}\ (x.a)\ \mathsf{in}\ p \mid Eq \mid (\lambda a.q)E \mid Et \mid \iota_1 E \mid \iota_2 E \mid (E, p) \mid (V, E) \mid (t, E) \mid \#E$$

The set of evaluation contexts is larger than the set of pure evaluation contexts, because it includes $\#$. As before, $E[p]$ denotes the proof term obtained from E by replacing its placeholder $[\,]$ with the proof term p.

4.1.6. Definition. The reduction relation on proof terms "$\to$" is defined by the following rewrite rules:

$$(\lambda a.p)V \to p\{V/a\} \qquad \mathsf{case}\ \iota_i V\ \mathsf{of}\ \left(a_1.p_1 \| a_2.p_2\right) \to p_i\{V/a_i\}$$

$$(\lambda x.p)t \to p\{t/x\} \qquad \mathsf{dest}\ (t, V)\ \mathsf{as}\ (x.a)\ \mathsf{in}\ p \to p\{t/x\}\{V/a\}$$

$$\pi_i(V_1, V_2) \to V_i \qquad \#P[\mathscr{S}k.p] \to \#p\,\{(\lambda a.\#P[a])/k\}$$

$$\#V \to V \qquad E[p] \to E[p'] \text{ when } p \to p'$$

The last rule is known as the "congruent closure" of the preceding rules. The rule for $\mathscr{S}$ applies only when the evaluation context P is pure. The reduction strategy determined by the rules is call-by-value reduction.

4.1.7. Example. The following are proof terms corresponding to the derivations given for MP_T and DNS_T in the beginning of this section.

$$\lambda e.\lambda a.\#e(a(\lambda b.\mathscr{S}k.b))$$

$$\lambda a.\lambda b.\#b(\lambda x.\mathscr{S}k.axk)$$

Remark that the proof term for MP_T does not make use of the continuation k, but only uses the $\mathscr{S}$ operator to pass the value b, once it has been found in the course of the computation, back to the control delimiter #.

4.2. Relationship to MQC and CQC

In this section we define a translation of MQC^+-derivations to MQC-derivations, thus obtaining the equiconsistency of the two systems, and some characterisation of the mutual-provability in MQC, CQC and MQC^+. The idea is to use a kind of continuations monad for interpreting a formula A whose structure depends on the syntactic shape of A.

4.2.1. Definition. The *superscript* translation A^T of a formula A with respect to a Σ-formula T is defined via the *subscript* translation A_T, which is defined by recursion on the structure of A:

$$A^T := (A_T \Rightarrow T) \Rightarrow T$$

$$\begin{aligned} A_T &:= A && \text{if } A \text{ is atomic} \\ (A \Box B)_T &:= A_T \Box B_T && \text{for } \Box = \vee, \wedge \\ (A \Rightarrow B)_T &:= A_T \Rightarrow B^T \\ (\exists A)_T &:= \exists A_T \\ (\forall A)_T &:= \forall A^T \end{aligned}$$

We write Γ_T for the translation $(-)_T$ applied to each formula of a context Γ individually.

The translation is the standard call-by-value CPS translation of types [**151**], and is similar to the Kuroda translation [**161**], the difference being that we add a double negation, not only after $\forall$, but also after $\Rightarrow$. Curiously, when interpreting, using DNS, the negative translation of the axiom of countable choice AC_0, a transformation from the Kuroda translation of AC_0 into our form, with $\neg\neg$ after $\Rightarrow$, seems [**114**, p. 200] to be needed. Avigad has remarked [**17**] that the Kuroda translation makes essential use of the $\bot_E$ rule.

4.2.2. Remark. A call-by-name version of the translation $(\cdot)^T$ can be used as well, as shown in Appendix A. We decided to stick with the call-by-value translation in order to correspond more closely with the simulation result of Section 4.4.

It will be sometimes convenient to use the standard monadic operations for manipulating the $(\cdot)^T$ translation. These take the form of proof transformations.

4.2.3. Lemma. *The following operations are definable for MQC:*

"unit" $\eta(\cdot)$**:** $\Gamma \vdash A_T \to \Gamma \vdash A^T$
"bind" $(\cdot)^*(\cdot)$**:** $\left(\forall \Gamma' \geq \Gamma.\, \Gamma' \vdash A_T \to \Gamma' \vdash B^T\right) \to \Gamma \vdash A^T \to \Gamma \vdash B^T$
"run" $\mu(\cdot)$**:** $\Gamma \vdash T^T \to \Gamma \vdash T$

PROOF. Using the proof terms:

$$\eta = \alpha \mapsto \lambda k.k\alpha$$
$$\mu = \alpha \mapsto \alpha(\lambda a.a)$$
$$\phi^* q = \lambda k.q(\lambda a.(\phi \cdot a)k)$$

□

Notice that we can only "run" a monad when it is of form T^T and T is a Σ-formula.

We will denote derivation in MQC$^+$by "$\vdash^+$", derivation in MQC by "$\vdash^m$", derivation in IQC by "$\vdash^i$", and the one in CQC by "$\vdash^c$".

4.2.4. Theorem (Equiconsistency)**.** *Given a derivation of* $\Gamma \vdash^+ A$*, which uses* $\mathscr{S}$ *and* # *for the* Σ*-formula* T*, we can build a derivation of* $\Gamma_T \vdash^m A^T$*.*

PROOF. By induction on the derivation. For each rule we use a corresponding proof term from the table below, except in the cases for $\wedge, \vee, \exists$ – when the formula being proved is a Σ-formula – then we can simply apply the last used derivation rule on the induction hypotheses.

$$\overline{a} = \lambda k.ka = \eta \cdot a$$
$$\overline{\lambda a.p} = \lambda k.k\left(\lambda a.\lambda k'.\overline{p}\left(\lambda b.k'b\right)\right)$$
$$\overline{pq} = \lambda k.\overline{p}\left(\lambda f.\overline{q}\left(\lambda a.fa\left(\lambda b.kb\right)\right)\right)$$
$$\overline{(p,q)} = \lambda k.\overline{p}\left(\lambda a.\overline{q}\left(\lambda b.k\left(a,b\right)\right)\right)$$
$$\overline{\pi_1 p} = \lambda k.\overline{p}\left(\lambda c.k\left(\pi_1 c\right)\right)$$
$$\overline{\iota_1 p} = \lambda k.\overline{p}\left(\lambda a.k\left(\iota_1 a\right)\right)$$
$$\overline{\mathsf{case}\ p\ \mathsf{of}\ \left(a_1.q_1 \| a_2.q_2\right)} = \lambda k.\overline{p}\left(\lambda c.\ \mathsf{case}\ c\ \mathsf{of}\ \left(a_1.\overline{q_1}k \| a_2.\overline{q_2}k\right)\right)$$
$$\overline{\lambda x.p} = \lambda k.k\left(\lambda x.\lambda k'.\overline{p}\left(\lambda b.k'b\right)\right)$$
$$\overline{pt} = \lambda k.\overline{p}\left(\lambda f.ftk\right)$$
$$\overline{(t,p)} = \lambda k.\overline{p}\left(\lambda a.k(t,a)\right)$$
$$\overline{\mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q} = \lambda k.\overline{p}\left(\lambda c.\ \mathsf{dest}\ c\ \mathsf{as}\ (x.a)\ \mathsf{in}\ \overline{q}k\right)$$
$$\overline{\#ap} = \lambda k.k\left(\overline{p}(\lambda a.a)\right) = \eta \cdot (\mu \cdot \overline{p})$$
$$\overline{\mathscr{S} l.p} = \lambda k.\left(\overline{p}(\lambda a.a)\right)\left\{\lambda a.\lambda k'.k'\left(ka\right)/\,l\right\}$$
$$= \lambda k.\left(\mu \cdot \overline{p}\right)\left\{\lambda a.\eta \cdot (ka)/\,l\right\}$$

□

We would now like to compare the MQC$^+$-provability of certain forms of formulae with their probability in MQC and CQC. Please note that we abuse the language by taking CQC to be MQC plus the double-negation elimination rule relative to a global fixed formula T i.e. $\vdash^c \neg_T\neg_T A \to \vdash^c A$. By $\neg_T A$ we denote the formula $A \Rightarrow T$. When it is clear from the context, we omit the annotation T from $\neg_T$.

4.2.5. Definition. The *Markov Principle for* T is the following generalisation of the minimal-logic version of Markov's principle:

$$(\mathrm{MP}_T) \qquad (T \Rightarrow S) \Rightarrow \neg_T\neg_T S \Rightarrow S,$$

or, more symmetrically,

$$\neg_T\neg_T S \Rightarrow \neg_S\neg_S T.$$

4.2.6. Definition. The *Double Negation Shift for T* (DNS_T) is the following generalisation of the usual DNS schema, extended with a clause handling implication:

($\mathrm{DNS}_T^{\forall}$) $\quad \forall x.\neg_T\neg_T A(x) \Rightarrow \neg_T\neg_T(\forall x.A(x))$

($\mathrm{DNS}_T^{\Rightarrow}$) $\quad (A \to \neg_T\neg_T B) \Rightarrow \neg_T\neg_T(A \to B)$

4.2.7. Proposition. *$DNS_T \vdash^m \neg_T\neg_T A \Leftrightarrow A^T$.*

PROOF. Induction on the complexity of A. When A is atomic, $A^T = \neg\neg A$.

($\wedge$) Both directions are via the proof term

$$\lambda c.\lambda k.\mathrm{IH}_A\big(\lambda k'.c\big(\lambda d.k'(\pi_1 d)\big)\\ \big(\lambda a.\mathrm{IH}_B\big(\lambda k'.c\big(\lambda d.k'(\pi_2 d)\big)\big)(\lambda b.k(a,b))\big)\big).$$

($\vee$) Both directions are via the proof term

$$\lambda a.\lambda k.a(\lambda c.\\ \mathsf{case}\ c\ \mathsf{of}\ (a_1.\mathrm{IH}_A(\lambda l.l a_1)(\lambda b.k(\iota_1 b)) \parallel a_2.\mathrm{IH}_B(\lambda l.l a_2)(\lambda b.k(\iota_2 b))))$$

($\exists$) Analogous to case ($\vee$).

($\Rightarrow$) From left to right via the proof term

$$\lambda c.\lambda k.\mathrm{IH}_A^{\rightarrow}\big(\lambda k'.k\big(\lambda a.\lambda k''.k' a\big)\big)\\ \big(\lambda a.\mathrm{IH}_B^{\leftarrow}\big(\lambda k'.c\big(\lambda f.k'(f a)\big)\big)\big(\lambda b.k\big(\lambda a'.b\big)\big)\big).$$

From right to left, had we had the ex-falso rule, we could have given the proof term

$$\lambda c.\lambda k.\mathrm{IH}_A^{\leftarrow}\big(\lambda k'.k\big(\lambda a.\mathrm{abort}(k' a)\big)\big)\\ \big(\lambda a.\mathrm{IH}_B^{\rightarrow}\big(\lambda k'.c\big(\lambda f.k'(f a)\big)\big)\big(\lambda b.k\big(\lambda a'.b\big)\big)\big).$$

But, since we are in minimal logic, we use $\mathrm{DNS}_T^{\Rightarrow}$:

$$\lambda c.\lambda k.\mathrm{IH}_A^{\leftarrow}\big(\lambda k'.\mathrm{DNS}_T^{\Rightarrow}\big(\lambda a.\lambda k''.k' a\big)k\big)\\ \big(\lambda a.\mathrm{IH}_B^{\rightarrow}\big(\lambda k'.c\big(\lambda f.k'(f a)\big)\big)\big(\lambda b.k\big(\lambda a'.b\big)\big)\big).$$

It is not known to the author if the use of $\mathrm{DNS}_T^{\Rightarrow}$ can be avoided.

($\forall$) We have:

$$(\forall x A(x))^T = \neg\neg(\forall x A^T(x)) \overset{\mathrm{IH}}{\leftrightarrow} \neg\neg(\forall x \neg\neg A(x))\\ \overset{\mathrm{DNS}_T^{\forall}}{\leftrightarrow} \neg\neg\neg\neg\forall x A(x) \leftrightarrow \neg\neg\forall x A(x)$$

□

4.2.8. Lemma. *$\Gamma \vdash^c A$ if and only if $\Gamma_T \vdash^m A^T$.*

PROOF. The direction right-to-left follows from the previous lemma. The other direction is by induction on the derivation of $\Gamma \vdash^c A$. Actually, we can use the CPS translation of Theorem 4.2.4 to treat all cases, except for the $\neg\neg_E$ rule

which was not covered by the translation. To show that $\Gamma_T \vdash^m A^T$ follows from $\Gamma_T \vdash^m (\neg\neg A)^T$, we use the fact that $\vdash^m \neg\neg(T_T) \leftrightarrow T$ (by the μ and η operators):

$$(\neg\neg A)^T = ((A \Rightarrow T) \Rightarrow T)^T = \neg\neg((A_T \Rightarrow \neg\neg T) \Rightarrow \neg\neg T)$$
$$\Leftrightarrow \neg\neg((A_T \Rightarrow T) \Rightarrow T) = \neg\neg\neg\neg A_T \Leftrightarrow \neg\neg A_T = A^T.$$

□

4.2.9. Corollary. *For any formula A, we have:*

$$\begin{array}{ccccc} \vdash^+ A & \xrightarrow{4.2.4} & \vdash^m A^T & \xleftarrow{4.2.8} & \vdash^c A \\ & & \downarrow{\scriptstyle 4.2.7} & & \\ \vdash^+ \neg\neg A & \longleftarrow & DNS_T \vdash^m \neg\neg A & & \end{array}$$

4.2.10. Corollary. *For any formula A, we have the following diagram:*

$$\begin{array}{ccccc} \vdash^+ \neg\neg A & \xrightarrow{4.2.4} & \vdash^m (\neg\neg A)^T & \xleftarrow{4.2.8} & \vdash^c A \\ \uparrow & & \downarrow{\scriptstyle 4.2.7} & & \\ DNS_T \vdash^m \neg\neg A & \longleftarrow & DNS_T \vdash^m \neg\neg\neg\neg A & & \end{array}$$

In particular, the statement $\vdash^+ \neg\neg A \longleftrightarrow \vdash^c A$ represents an extension of Glivenko's theorem [**81, 171, 181**] *to predicate logic.*

4.3. Subject reduction and progress

In this section we prove that the reduction on proof terms satisfies Subject Reduction and Progress. First, a couple of lemmas are in order.

4.3.1. Lemma (Annotation Weakening)**.** *If $\Gamma \vdash p : A$, then $\Gamma \vdash_T p : A$ for any T.*

PROOF. A simple induction on the derivation. □

4.3.2. Lemma (Substitutions)**.** *The following hold:*

(1) *If $\Gamma, a : A \vdash_\diamond p : B$ and $\Gamma \vdash_\diamond q : A$, then $\Gamma \vdash_\diamond p\{q/a\} : B$.*
(2) *If $\Gamma \vdash_\diamond p : B(x)$, where x is fresh, and t is a closed term, then $\Gamma \vdash_\diamond p\{t/x\} : B(t)$.*

PROOF. The proof is standard, by induction on the derivation (see for example [**155**]). The new rules $\mathscr{S}$ and # pose no problems, since we can use the identities $(\#p)\{q/a\} = \#(p\{q/a\})$ and $(\mathscr{S}k.p)\{q/a\} = \mathscr{S}k.(p\{q/a\})$ when k is fresh. □

4.3.3. Lemma (Decomposition)**.** *If $\Gamma \vdash_T P[\mathscr{S}k.p] : B$, then there is a formula A and derivations $\Gamma, k : A \Rightarrow T \vdash_T p : T$ and $\Gamma, a : A \vdash_T P[a] : B$.*

PROOF. The proof is by induction on the derivation. We only need to consider the rules that can generate a pure evaluation context of the required form. Of the rules that we consider, for the intuitionistic rules, the proof is simply by using the induction hypothesis, as shown below for the $\wedge_I$ rule; and the only non-intuitionistic rule to consider is $\mathscr{S}$, because # does not generate a pure evaluation context.

- For $\wedge_I$, there are two cases to consider, depending on whether the pure evaluation context is $(P[\mathscr{S}k.p], q)$ or $(V, P[\mathscr{S}k.p])$, but the proofs are analogous. Let the last rule in the derivation be:

$$\frac{\Gamma \vdash_T P[\mathscr{S}k.p] : B_1 \qquad \Gamma \vdash_T q : B_2}{\Gamma \vdash_T (P[\mathscr{S}k.p], q) : B_1 \wedge B_2}$$

The induction hypothesis gives us a formula A_1 and two derivations, $\Gamma, k : A_1 \Rightarrow T \vdash_T p : T$ and $\Gamma, a : A_1 \vdash_T P[a] : B_1$, from which the goal follows by choosing $A := A_1$.

- For $\mathscr{S}$, the pure evaluation context must be the empty one, so the last used rule is:

$$\frac{\Gamma, k : B \Rightarrow T \vdash_T p : T}{\Gamma \vdash_T [\mathscr{S}k.p] : B}$$

If we set $A := B$, the goal follows from the premise of the rule above and, for $\Gamma, a : A \vdash_T [a] : A$, from the Ax rule.

□

4.3.4. Lemma (Annotation Strengthening)**.** $\Gamma \vdash_S V : T \longrightarrow \Gamma \vdash V : T$

Proof. The proof is by induction on the derivation and very simple. We only need to consider the intuitionistic rules that introduce a value and that prove a Σ-formula, that is, the rules Ax, $\wedge_I$, $\vee_I^1$, $\vee_I^2$, and $\exists_I$. $\mathscr{S}$ and # do not introduce a value. □

4.3.5. Theorem (Subject Reduction)**.** *If* $\Gamma \vdash_\diamond p : A$ *and* $p \to q$*, then* $\Gamma \vdash_\diamond q : A$.

Proof. The proof is by induction on the derivation and is standard (see for example [**155**]), by using Substitutions Lemma 4.3.2 and Decomposition Lemma 4.3.3. Below, we consider the new rules and, for illustration, one of the intuitionistic rules.

(#) We have $\Gamma \vdash_\diamond \#p$ and $\#p \to q$ for some q. We look at three possible cases, because there are three rules for rewriting a term of form $\#p$. If $q \equiv \#q'$ and the reduction was by the congruence rule, we have $p \to q'$; now use IH and the # rule to finish the proof. If p is a value and $q \equiv p$, then $\Gamma \vdash_T q : T$; now use Strengthening Lemma 4.3.4 to conclude $\Gamma \vdash q : T$. The third case is when $p \equiv P[\mathscr{S}k.p']$ and $q \equiv \#p'\{(\lambda a.\#P[a])/k\}$, and the proof is by combining Lemmas 4.3.2 and 4.3.3.

($\mathscr{S}$) This case is impossible, since there are no rules for reducing a term of form $\mathscr{S}k.p$ on its own, and the set of evaluation contexts does not include a clause for $\mathscr{S}k.[\,]$.

($\wedge_E^1$) We have $\Gamma \vdash_\diamond p : A \wedge B$, $\Gamma \vdash_\diamond \pi_1 p : A$, and $\pi_1 p \to q$. If the reduction was by the congruence rule, then $q \equiv \pi_1 q'$ for some q', and we can use IH. Otherwise, $p \equiv (V_1, V_2)$ and $q \equiv V_1$, and $\Gamma \vdash_\diamond p : A \wedge B$ must have been proved by the $\wedge_I$ rule, which is enough.

□

While the theorem above shows that reducing a proof term does not change its logical specification, the next one shows that a proof term which is not in normal form does not get "stuck".

4.3.6. Theorem (Progress)**.** *If* $\vdash_\diamond p : A$*,* p *is not a value, and* p *is not of form* $P[\mathscr{S}k.p']$*, then* p *reduces in one step to some proof term* r.

Proof. By induction on the derivation. The cases Ax, ($\Rightarrow_I$), and ($\forall_I$) introduce a value, while the case ($\mathscr{S}$) introduces a $\mathscr{S}k.p$ term.

($\wedge_I$) We have that $\vdash_\diamond (p,q) : A \wedge B$ and (p,q) is neither a value nor of form $P[\mathscr{S}k.p']$. Then also none of p,q is of form $P[\mathscr{S}k.p']$. If p is not value, by IH, for some r, $(p,q) \to (r,q)$. If p is a value, then q must be a non-value, and then we use IH on q.

($\wedge_E^1$) We have that $\vdash_\diamond \pi_1 p : A$ and that $\pi_1 p$, hence p itself, is not of form $P[\mathscr{S}k.p']$. If p is a value, then it must be a pair (V_1,V_2), so $\pi_1(V_1,V_2) \to V_1$. If p is not a value, we can use IH to obtain $\pi_1 p \to \pi_1 r$ for some r.

($\vee_I$) From $\vdash_\diamond \iota_1 p : A \vee B$ and $\iota_1 p$ a non-value and not of form $P[\mathscr{S}k.p']$, we have that p is not a value and not of that form, so we use IH to obtain an r such that $p \to r$, hence $\iota_1 p \to \iota_1 r$.

($\vee_E$) We have $\vdash_\diamond \mathsf{case}\ p\ \mathsf{of}\ \left(a_1.p_1 \| a_2.p_2\right) : C$. If p is a value, then it is of form $\iota_i V$, therefore $\mathsf{case}\ \iota_i V\ \mathsf{of}\ \left(a_1.p_1 \| a_2.p_2\right) \to p_i\{V/a_i\}$. If p is of form $P[\mathscr{S}k.p']$, then so is $\mathsf{case}\ p\ \mathsf{of}\ \left(a_1.p_1 \| a_2.p_2\right)$. Otherwise, we use IH to obtain an r such that $\mathsf{case}\ p\ \mathsf{of}\ \left(a_1.p_1 \| a_2.p_2\right) \to \mathsf{case}\ r\ \mathsf{of}\ \left(a_1.p_1 \| a_2.p_2\right)$.

($\Rightarrow_E$) We have $\vdash_\diamond pq : B$. If either p or q is of form $P[\mathscr{S}k.p']$, then so is pq. If p is a value, then it is of form $\lambda a.r$; if q is a value $E[(\lambda a.r)q] \to E[r\{q/a\}]$; if q is not a value, by IH, $E[(\lambda a.r)q] \to E[(\lambda a.r)q']$ for some q'. Otherwise, by IH, $p \to r$ for some r, so $pq \to rq$.

($\forall_E$) We have $\vdash_\diamond pt : A(t)$. If p is of form $P[\mathscr{S}k.p']$, then so is pt. If p is a value, then it is of form $\lambda x.r$, hence $(\lambda x.r)t \to r\{t/x\}$. Otherwise, by IH, $p \to r$ for some r, so $pt \to rt$.

($\exists_I$) From $\vdash_\diamond (t,p) : A(t)$ and (t,p) a non-value and not of form $P[\mathscr{S}k.p']$, we have that p is not a value and not of that form, so we use IH to obtain an r such that $(t,p) \to (t,r)$.

($\exists_E$) We have $\vdash_\diamond \mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q : C$. If p is a value, then it is of form (t,V), therefore $\mathsf{dest}\ (t,V)\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q \to q\{t/x\}\{V/a\}$. If p is of form $P[\mathscr{S}k.p']$, then so is $\mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q$. Otherwise, we use IH to obtain an r such that $\mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q \to \mathsf{dest}\ r\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q$.

(#) We have $\vdash_\diamond \#p : T$. If p is a value, then $\#p \to p$. If $p \equiv P[\mathscr{S}k.p']$, then $\#p \to \#p'\{\lambda a.\#P[a]/k\}$. If p is neither a value nor of form $P[\mathscr{S}k.p']$, by IH, $p \to p'$, so $\#p \to \#p'$.

□

4.4. Normalisation, disjunction and existence properties

In this section we prove that every well-delimited[4] proof term of MQC$^+$, which is not a value, reduces to a value in finitely many steps.

A first approach would be to use Normalisation of MQC and Theorem 4.2.4, and prove in addition only that the CPS translation preserves reduction, that is, if $\vdash^+ p : A$ and $p \to r$, then $\overline{p}k \to^+ \overline{r}k$, for any proof term k of the right type. For the case of simply-typed call-by-value lambda calculus, and a CPS translation equivalent to ours, this statement is known as Plotkin's simulation theorem [**151**]. But, even for that restriction of our calculus (no pairs, no sums, and no delimited control), Plotkin had to come up with a technique known as the "colon" translation, in order to deal with "administrative" η-redexes, and the overall translation proceeds in two phases.

[4] By *well-delimited*, we will refer to proof terms in which there is an enclosing reset delimiter for the shift operators used. That is, a proof term is well-delimited, when it corresponds to a derivation $\vdash^+$ without an annotating Σ-formula at the root of the derivation tree.

Instead of extending the colon translation to the additional constructs that we have, we took the approach of Danvy and Filinski from [**60**]. They give a one-phase proof of simulation for the call-by-value lambda calculus that avoids the colon translation, and also shows how to handle the extra logical connectives and delimited control. The trick is to use a two-level lambda calculus: a meta-level consisting of the meta- function abstraction and application, and an object-level lambda calculus as target of the translation. In that way, the administrative η-redexes can be handled by the meta-language.

The two-level CPS translation is given in Table 3. The meta-level abstraction and application are denoted by "$\mapsto$" and "$\alpha \cdot b$", to differ from the object level ones, "λ" and "ab". The Greek letter κ will stand for meta-continuations, that is, proof transformations of type $\Gamma_T \vdash^m A_T \to \Gamma_T \vdash^m T$, and the letters $\alpha, \beta, \gamma, \phi$ will stand for derivations $\Gamma_T \vdash^m C$. The meta-continuation "id" will stand for $\alpha \mapsto \alpha$. For the typing of the two-level translation of p we thus have

$$\overline{p} : (\Gamma_T \vdash^m A_T \to \Gamma_T \vdash^m T) \to \Gamma_T \vdash^m T.$$

Note, however, that for the lemmas of this section to be well-typed, we need to be slightly more general in the typing as described in Appendix A. The added complexity does not change the argument, therefore we give a version which closelly follows the original work of Danvy and Filinski.

$$\begin{aligned}
\overline{a} &= \kappa \mapsto \kappa \cdot a \\
\overline{\lambda a.p} &= \kappa \mapsto \kappa \cdot \left(\lambda a.\lambda k.\overline{p} \cdot \left(\beta \mapsto k\beta\right)\right) \\
\overline{pq} &= \kappa \mapsto \overline{p} \cdot \left(\phi \mapsto \overline{q} \cdot \left(\alpha \mapsto \phi\alpha\,(\lambda b.\kappa \cdot b)\right)\right) \\
\overline{(p,q)} &= \kappa \mapsto \overline{p} \cdot \left(\alpha \mapsto \overline{q} \cdot \left(\beta \mapsto \kappa \cdot (\alpha,\beta)\right)\right) \\
\overline{\pi_1 p} &= \kappa \mapsto \overline{p} \cdot \left(\gamma \mapsto \kappa \cdot (\pi_1 \gamma)\right) \\
\overline{\iota_1 p} &= \kappa \mapsto \overline{p} \cdot (\alpha \mapsto \kappa \cdot (\iota_1 \alpha)) \\
\overline{\mathsf{case}\ p\ \mathsf{of}\ \left(a_1.q_1 \| a_2.q_2\right)} &= \kappa \mapsto \overline{p} \cdot \left(\gamma \mapsto \left(\mathsf{case}\ \gamma\ \mathsf{of}\ \left(a_1.\overline{q_1} \cdot \kappa \| a_2.\overline{q_2} \cdot \kappa\right)\right)\right) \\
\overline{\lambda x.p} &= \kappa \mapsto \kappa \cdot \left(\lambda x.\lambda k.\overline{p} \cdot \left(\beta \mapsto kx\beta\right)\right) \\
\overline{pt} &= \kappa \mapsto \overline{p} \cdot \left(\phi \mapsto \phi t\,(\lambda b.\kappa \cdot b)\right) \\
\overline{(t,p)} &= \kappa \mapsto \overline{p} \cdot (\alpha \mapsto \kappa \cdot (t,\alpha)) \\
\overline{\mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q} &= \kappa \mapsto \overline{p} \cdot \left(\gamma \mapsto \left(\mathsf{dest}\ \gamma\ \mathsf{as}\ (x.a)\ \mathsf{in}\ \overline{q} \cdot \kappa\right)\right) \\
\overline{\#p} &= \kappa \mapsto \kappa \cdot \left(\overline{p} \cdot \mathrm{id}\right) \\
\overline{\mathscr{S} l.p} &= \kappa \mapsto \left(\overline{p} \cdot \mathrm{id}\right)\left\{\lambda a.\lambda k.k\,(\kappa \cdot a) / l\right\}
\end{aligned}$$

Table 3: Two-level CPS translation

4.4.1. Lemma. *If $\vdash^+ V : A$ for V a value, then, for any κ, we have that $\overline{V} \cdot \kappa = \kappa \cdot (\overline{V} \cdot \mathrm{id})$.*

PROOF. We do induction on the derivation and consider only the rules that can introduce a value (Definition 4.1.3).

(AX) $\overline{a} \cdot \kappa = (\kappa \mapsto \kappa \cdot a) \cdot \kappa = \kappa \cdot a = \kappa \cdot ((\kappa \mapsto \kappa \cdot a) \cdot \mathrm{id}) = \kappa \cdot (\overline{a} \cdot \mathrm{id})$.

$(\wedge_I)$

$$\begin{aligned}\overline{(V_1,V_2)}\cdot\kappa &= \overline{V_1}\cdot\Big(\alpha\mapsto\overline{V_2}\cdot\big(\beta\mapsto\kappa\cdot(\alpha,\beta)\big)\Big)\\ &= \overline{V_2}\cdot\Big(\beta\mapsto\kappa\cdot\Big(\overline{V_1}\cdot\mathrm{id},\beta\Big)\Big)\\ &= \kappa\cdot\Big(\overline{V_1}\cdot\mathrm{id},\overline{V_2}\cdot\mathrm{id}\Big)\\ &= \kappa\cdot\Big(\overline{V_2}\cdot\Big(\beta\mapsto\Big(\overline{V_1}\cdot\mathrm{id},\beta\Big)\Big)\Big)\\ &= \kappa\cdot\Big(\overline{V_1}\cdot\Big(\alpha\mapsto\overline{V_2}\cdot\big(\beta\mapsto(\alpha,\beta)\big)\Big)\Big)\\ &= \kappa\cdot\Big(\overline{V_1}\cdot\Big(\alpha\mapsto\overline{V_2}\cdot\big(\beta\mapsto\mathrm{id}\cdot(\alpha,\beta)\big)\Big)\Big) \qquad = \kappa\cdot\Big(\overline{(V_1,V_2)}\cdot\mathrm{id}\Big)\end{aligned}$$

$(\vee_I^1)$

$$\begin{aligned}\overline{\iota_1 V}\cdot\kappa &= \overline{V}\cdot(\alpha\mapsto\kappa\cdot(\iota_1\alpha))\\ &= \kappa\cdot\Big(\iota_1\Big(\overline{V}\cdot\mathrm{id}\Big)\Big)\\ &= \kappa\cdot\Big(\overline{V}\cdot(\alpha\mapsto(\iota_1\alpha))\Big)\\ &= \kappa\cdot\Big(\overline{V}\cdot(\alpha\mapsto\mathrm{id}\cdot(\iota_1\alpha))\Big) \qquad = \kappa\cdot\Big(\overline{\iota_1 V}\cdot\mathrm{id}\Big)\end{aligned}$$

$(\Rightarrow_I)$

$$\begin{aligned}\overline{\lambda a.p}\cdot\kappa &= \kappa\cdot\big(\lambda a.\lambda k.\overline{p}\cdot\big(\beta\mapsto k\beta\big)\big)\\ &= \kappa\cdot\big(\mathrm{id}\cdot\big(\lambda a.\lambda k.\overline{p}\cdot\big(\beta\mapsto k\beta\big)\big)\big) \qquad = \kappa\cdot\Big(\overline{\lambda a.p}\cdot\mathrm{id}\Big)\end{aligned}$$

The cases $(\forall_I)$ and $(\exists_I)$ are analogous to the ones of $(\Rightarrow_I)$ and $(\vee_I^1)$. □

4.4.2. Lemma. *If* $\Gamma\vdash^+ q:A$ *and* V *is a closed value, then* $\overline{q\{V/a\}}\cdot\kappa = \big(\overline{q}\cdot\kappa\big)\{\overline{V}\cdot\mathrm{id}/a\}$, *for any* κ, a.

PROOF. By induction on the derivation of q.

If $\Gamma\vdash^+ q:A$ is derived by the AX rule, then q is a variable. If $q=a$, then by Lemma 4.4.1, $\overline{a\{V/a\}}\cdot\kappa = \overline{V}\cdot\kappa = \kappa\cdot(\overline{V}\cdot\mathrm{id}) = (\kappa\mapsto\kappa\cdot(\overline{V}\cdot\mathrm{id}))\cdot\kappa = ((\kappa\mapsto\kappa\cdot a)\cdot\kappa)\{(\overline{V}\cdot\mathrm{id})/a\} = (\overline{a}\cdot\kappa)\{\overline{V}\cdot\mathrm{id}/a\}$. Otherwise, $V=b\neq a$ and the substitution has no effect.

The rest of the cases are directly by the induction hypothesis, using the fact that V is closed and that the binders λ and $\mathscr{S}$ bind fresh variables, hence the substitution $\{V/a\}$ can traverse the structure of q. We give one case for illustration.

$$\begin{aligned}\overline{\big(\mathscr{S}l.p\big)\{V/a\}}\cdot\kappa &= \overline{\mathscr{S}l.\big(p\{V/a\}\big)}\cdot\kappa\\ &= \Big(\overline{p\{V/a\}}\cdot\mathrm{id}\Big)\big\{\lambda b.\lambda k.k\,(\kappa\cdot b)/l\big\}\\ &= \big(\overline{p}\cdot\mathrm{id}\big)\Big\{\overline{V}\cdot\mathrm{id}\Big/a\Big\}\big\{\lambda b.\lambda k.k\,(\kappa\cdot b)/l\big\}\\ &= \big(\overline{p}\cdot\mathrm{id}\big)\big\{\lambda b.\lambda k.k\,(\kappa\cdot b)/l\big\}\Big\{\overline{V}\cdot\mathrm{id}\Big/a\Big\}\\ &= \overline{\mathscr{S}l.p}\Big\{\overline{V}\cdot\mathrm{id}\Big/a\Big\}\end{aligned}$$

□

4.4.3. Lemma. *If P is a pure evaluation context, then, for every κ,*

$$\overline{P[\mathscr{S}\,l.s]}\cdot\kappa = \left(\overline{s}\cdot\mathrm{id}\right)\left\{\lambda a.\lambda k.k(\overline{P[a]}\cdot\kappa)/\,l\right\}.$$

PROOF. By induction on the derivation of P, considering only rules that can construct a pure evaluation context (Definition 4.1.4).

For the empty pure evaluation context $[\,]$, the equality follows directly from the two-level CPS translation of $\mathscr{S}$:

$$\begin{aligned}\overline{[\mathscr{S}\,l.s]}\cdot\kappa &= \left(\overline{s}\cdot\mathrm{id}\right)\left\{\,\lambda a.\lambda k.k\,(\kappa\cdot a)\big/\,l\right\}\\ &= \left(\overline{s}\cdot\mathrm{id}\right)\left\{\,\lambda a.\lambda k.k\left(\overline{[a]}\cdot\kappa\right)\Big/\,l\right\}\end{aligned}$$

For the other cases, it is enough to show that $Q[P[r]] = P[r]\cdot\kappa$ for some κ and any r. For example,

$$\begin{aligned}\overline{P[\mathscr{S}\,l.s]\,q}\cdot\kappa &= \overline{P[\mathscr{S}\,l.s]}\cdot\left(\phi\mapsto\overline{q}\cdot\left(\alpha\mapsto\phi\alpha\,(\lambda b.\kappa\cdot b)\right)\right)\\ &= \left(\overline{s}\cdot\mathrm{id}\right)\left\{\,\lambda a.\lambda k.k\left(\overline{P[a]}\cdot\left(\phi\mapsto\overline{q}\cdot\left(\alpha\mapsto\phi\alpha\,(\lambda b.\kappa\cdot b)\right)\right)\right)\Big/\,l\right\}\\ &= \left(\overline{s}\cdot\mathrm{id}\right)\left\{\,\lambda a.\lambda k.k\left(\overline{P[a]\,q}\cdot\kappa\right)\Big/\,l\right\}.\end{aligned}$$

From the sole form of the two-level CPS, this follows for the cases $\overline{P[\mathscr{S}\,l.s]\,q}$, $\overline{(P[\mathscr{S}\,l.s],q)}$, $\overline{(V,P[\mathscr{S}\,l.s])}$, $\overline{\pi_1 P[\mathscr{S}\,l.s]}$, $\overline{\iota_1 P[\mathscr{S}\,l.s]}$, $\overline{\mathsf{case}\ P[\mathscr{S}\,l.s]\ \mathsf{of}\ \left(a_1.q_1\|a_2.q_2\right)}$, $\overline{P[\mathscr{S}\,l.s]\,t}$, $\overline{(t,P[\mathscr{S}\,l.s])}$, and $\overline{\mathsf{dest}\ P[\mathscr{S}\,l.s]\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q}$. Hence we have to check only $\overline{\left(\lambda a.p\right)P[\mathscr{S}\,l.s]}$ (and $\overline{\left(\lambda x.p\right)P[\mathscr{S}\,l.s]}$ which is analogous):

$$\begin{aligned}\overline{(\lambda a.p)P[r]}\cdot\kappa &= \overline{\lambda a.p}\cdot\left(\phi\mapsto\overline{P[r]}\cdot\left(\alpha\mapsto\phi\alpha\,(\lambda b.\kappa\cdot b)\right)\right)\\ &= \left(\phi\mapsto\overline{P[r]}\cdot\left(\alpha\mapsto\phi\alpha\,(\lambda b.\kappa\cdot b)\right)\right)\cdot\left(\lambda a.\lambda k.\overline{s}\cdot\left(\beta\mapsto k\beta\right)\right)\\ &= \overline{P[r]}\cdot\left(\alpha\mapsto\left(\lambda a.\lambda k.\overline{s}\cdot\left(\beta\mapsto k\beta\right)\right)\alpha\,(\lambda b.\kappa\cdot b)\right)\end{aligned}$$

□

4.4.4. Lemma. *If V is a value, then so is $\overline{V}\cdot\mathrm{id}$.*

PROOF. By a simple induction on the derivation of V. We need to use Lemma 4.4.1 for the case of $\wedge$, $\vee$ and $\exists$. □

The following lemma is a strengthening of Theorem 4.3.6 that, besides proving that the proof term p reduces in one step to r in MQC$^+$, proves also that its two-level CPS translation applied to a meta-continuation κ gives rise to:

- no reduction in MQC, when $p\to r$ by one of the two reduction rules for reset;
- at least one step of reduction in MQC, when $p\to r$ by normal weak-head reduction, that is, by a reduction rule which is not one for reset and not the congruence rule;
- the same number of reduction steps in MQC, as the number of steps by the congruence rule.

We will use the wild-card notation $\to^\diamond$ to denote this dependency of the number of reduction steps for the translation, on the sort of the reduction rule used in MQC$^+$. We also use the wildcard for the Σ-formula annotation of $\vdash^+$, but the two uses of the wildcard are of course not related.

4.4.5. Lemma (CPS-Progress)**.** *If $\vdash^+_\diamond p : A$, p is not a value, and p is not of form $P[\mathscr{S}k.p']$, then, for some r, $p \to^1 r$ and, for any κ, $\overline{p}\cdot\kappa \to^\diamond \overline{r}\cdot\kappa$.*

PROOF. The proof has the same structure as the one of Theorem 4.3.6, therefore we will just amend it with the part concerning the two-level CPS translation.

($\wedge_I$) We consider the pair (p,q). If $p \to r$, then $\overline{(p,q)}\cdot\kappa = \overline{p}\cdot(\alpha \mapsto \overline{q}\cdot(\beta \mapsto \kappa\cdot(\alpha,\beta))) \overset{\text{IH}}{\to^\diamond} \overline{r}\cdot(\alpha \mapsto \overline{q}\cdot(\beta \mapsto \kappa\cdot(\alpha,\beta))) = \overline{(r,q)}\cdot\kappa$.

If p is a value and $q \to r$, then $\overline{(p,q)}\cdot\kappa = \overline{p}\cdot(\alpha \mapsto \overline{q}\cdot(\beta \mapsto \kappa\cdot(\alpha,\beta))) \overset{\text{IH}}{\to^\diamond} \overline{p}\cdot(\alpha \mapsto \overline{r}\cdot(\beta \mapsto \kappa\cdot(\alpha,\beta))) = \overline{(p,r)}\cdot\kappa$.

($\wedge^1_E$) We consider the term $\pi_1 p$. If p is a value, it is a pair (V_1,V_2) and $\pi_1(V_1,V_2) \to V_1$. Then, by using Lemma 4.4.1 and Lemma 4.4.4,

$$\begin{aligned}
\overline{\pi_1(V_1,V_2)}\cdot\kappa &= \overline{(V_1,V_2)}\cdot(\gamma \mapsto \kappa\cdot(\pi_1\gamma)) \\
&= \overline{V_1}\cdot\Big(\alpha \mapsto \overline{V_2}\cdot\big(\beta \mapsto \big(\gamma \mapsto \kappa\cdot(\pi_1\gamma)\big)\cdot(\alpha,\beta)\big)\Big) \\
&= \overline{V_2}\cdot\Big(\beta \mapsto \big(\gamma \mapsto \kappa\cdot(\pi_1\gamma)\big)\cdot\Big(\overline{V_1}\cdot\mathrm{id},\beta\Big)\Big) \\
&= \big(\gamma \mapsto \kappa\cdot(\pi_1\gamma)\big)\cdot\Big(\overline{V_1}\cdot\mathrm{id},\overline{V_2}\cdot\mathrm{id}\Big) \\
&= \kappa\cdot\Big(\pi_1\Big(\overline{V_1}\cdot\mathrm{id},\overline{V_2}\cdot\mathrm{id}\Big)\Big) \\
&\to \kappa\cdot\Big(\overline{V_1}\cdot\mathrm{id}\Big) \\
&= \overline{V_1}\cdot\kappa.
\end{aligned}$$

If p is not a value, then $\pi_1 p \to \pi_1 r$ for some r, and

$$\begin{aligned}
\overline{\pi_1 p}\cdot\kappa &= \overline{p}\cdot(\gamma \mapsto \kappa\cdot(\pi_1\gamma)) \\
&\overset{\text{IH}}{\to^\diamond} \overline{r}\cdot(\gamma \mapsto \kappa\cdot(\pi_1\gamma)) \\
&= \overline{\pi_1 r}\cdot\kappa.
\end{aligned}$$

($\vee_I$) We need to consider $\iota_1 p$ when p is neither a value nor of form $P[\mathscr{S}k.p']$, and when there is an r such that $p \to r$.

$$\begin{aligned}
\overline{\iota_1 p}\cdot\kappa &= \overline{p}\cdot(\alpha \mapsto \kappa\cdot(\iota_1\alpha)) \\
&\overset{\text{IH}}{\to^\diamond} \overline{r}\cdot(\alpha \mapsto \kappa\cdot(\iota_1\alpha)) \\
&= \overline{\iota_1 r}\cdot\kappa.
\end{aligned}$$

($\vee_E$) We consider case p of $\big(a_1.q_1 \| a_2.q_2\big)$. If p is a value, we know that

$$\mathsf{case}\ \iota_i V\ \mathsf{of}\ \big(a_1.q_1 \| a_2.q_2\big) \to q_i\{V/a_i\},$$

and then, by using Lemma 4.4.1, Lemma 4.4.2 and Lemma 4.4.4,

$$\begin{aligned}
\overline{\mathsf{case}\ \iota_i V\ \mathsf{of}\ \left(a_1.q_1 \| a_2.q_2\right)} \cdot \kappa &= \overline{\iota_i V} \cdot \left(\gamma \mapsto \left(\mathsf{case}\ \gamma\ \mathsf{of}\ \left(a_1.\overline{q_1} \cdot \kappa \| a_2.\overline{q_2} \cdot \kappa\right)\right)\right) \\
&= \overline{V} \cdot \left(\alpha \mapsto \left(\gamma \mapsto \left(\mathsf{case}\ \gamma\ \mathsf{of}\ \left(a_1.\overline{q_1} \cdot \kappa \| a_2.\overline{q_2} \cdot \kappa\right)\right)\right) \cdot \left(\iota_i \alpha\right)\right) \\
&= \overline{V} \cdot \left(\alpha \mapsto \left(\mathsf{case}\ \iota_i \alpha\ \mathsf{of}\ \left(a_1.\overline{q_1} \cdot \kappa \| a_2.\overline{q_2} \cdot \kappa\right)\right)\right) \\
&= \left(\mathsf{case}\ \iota_i \left(\overline{V} \cdot \mathrm{id}\right)\ \mathsf{of}\ \left(a_1.\overline{q_1} \cdot \kappa \| a_2.\overline{q_2} \cdot \kappa\right)\right) \\
&\to \left(\overline{q_i} \cdot \kappa\right)\{\overline{V} \cdot \mathrm{id} \,/\, a_i\} \\
&= \overline{q_i\{V \,/\, a_i\}} \cdot \kappa.
\end{aligned}$$

Otherwise, $p \to r$ for some r, and

$$\begin{aligned}
\overline{\mathsf{case}\ p\ \mathsf{of}\ \left(a_1.q_1 \| a_2.q_2\right)} \cdot \kappa &= \overline{p} \cdot \left(\gamma \mapsto \left(\mathsf{case}\ \gamma\ \mathsf{of}\ \left(a_1.\overline{q_1} \cdot \kappa \| a_2.\overline{q_2} \cdot \kappa\right)\right)\right) \\
&\overset{\mathrm{IH}}{\to^{\diamond}} \overline{r} \cdot \left(\gamma \mapsto \left(\mathsf{case}\ \gamma\ \mathsf{of}\ \left(a_1.\overline{q_1} \cdot \kappa \| a_2.\overline{q_2} \cdot \kappa\right)\right)\right) \\
&= \overline{\mathsf{case}\ r\ \mathsf{of}\ \left(a_1.q_1 \| a_2.q_2\right)} \cdot \kappa.
\end{aligned}$$

($\Rightarrow_E$) We consider the proof term pq. If p and q are both values, then $p = \lambda a.s$ and $pq \to s\{q/a\}$. Then, by using Lemma 4.4.1 and Lemma 4.4.2,

$$\begin{aligned}
\overline{(\lambda a.s)q} \cdot \kappa &= \overline{\lambda a.s} \cdot \left(\phi \mapsto \overline{q} \cdot \left(\alpha \mapsto \phi\alpha\,(\lambda b.\kappa \cdot b)\right)\right) \\
&= \left(\phi \mapsto \overline{q} \cdot \left(\alpha \mapsto \phi\alpha\,(\lambda b.\kappa \cdot b)\right)\right) \cdot \left(\lambda a.\lambda k.\overline{s} \cdot (b \mapsto kb)\right) \\
&= \overline{q} \cdot \left(\alpha \mapsto \left(\lambda a.\lambda k.\overline{s} \cdot (b \mapsto kb)\right)\alpha\,(\lambda b.\kappa \cdot b)\right) \\
&\to^{+} \overline{q} \cdot \left(\alpha \mapsto \left(\overline{s} \cdot (b \mapsto \kappa \cdot b)\right)\{\alpha/a\}\right) \\
&= \overline{q} \cdot \left(\alpha \mapsto \left(\overline{s} \cdot \kappa\right)\{\alpha/a\}\right) \\
&= (\overline{s} \cdot \kappa)\{\overline{q} \cdot \mathrm{id} \,/\, a\} \\
&= \overline{s\{q/a\}} \cdot \kappa.
\end{aligned}$$

In the $\to^{+}$ step we used the fact that s can have at most a as free variable, which follows from $p = \lambda a.s$ being a closed term.

If p is a value $\lambda a.s$ and q is not, then $q \to r$ for some r, and

$$\begin{aligned}
\overline{(\lambda a.s)q} \cdot \kappa &= \overline{\lambda a.s} \cdot \left(\phi \mapsto \overline{q} \cdot \left(\alpha \mapsto \phi\alpha\,(\lambda b.\kappa \cdot b)\right)\right) \\
&\to^{\diamond} \overline{\lambda a.s} \cdot \left(\phi \mapsto \overline{r} \cdot \left(\alpha \mapsto \phi\alpha\,(\lambda b.\kappa \cdot b)\right)\right) \\
&= \overline{(\lambda a.s)r} \cdot \kappa.
\end{aligned}$$

Otherwise, when p is not a value, and $p \to r$ for some r, we have that

$$\begin{aligned}
\overline{pq} \cdot \kappa &= \overline{p} \cdot \left(\phi \mapsto \overline{q} \cdot \left(\alpha \mapsto \phi\alpha\,(\lambda b.\kappa \cdot b)\right)\right) \\
&\to^{\diamond} \overline{r} \cdot \left(\phi \mapsto \overline{q} \cdot \left(\alpha \mapsto \phi\alpha\,(\lambda b.\kappa \cdot b)\right)\right) \\
&= \overline{rq} \cdot \kappa.
\end{aligned}$$

($\forall_E$) By analogy with $\Rightarrow_E$, but simpler.

($\exists_I$) By analogy with $\vee_I$.

($\exists_E$) By analogy with $\vee_E$.

(#) We consider the proof term $\#p$. If p is a value, then $\#p \to p$ and

$$\overline{\#p} \cdot \kappa = \kappa \cdot (\overline{p} \cdot \mathrm{id}) = \overline{p} \cdot \kappa.$$

If p is of form $P[\mathscr{S}l.q]$, then

$$\begin{aligned}
\overline{\#P[\mathscr{S}l.q]}\cdot\kappa &= \kappa\cdot\left(\overline{P[\mathscr{S}l.q]}\cdot\mathrm{id}\right)\\
\text{(by Lemma 4.4.3)}\quad &= \kappa\cdot\left(\overline{q}\cdot\mathrm{id}\right)\left\{\lambda a.\lambda k.k\left(\overline{P[a]}\cdot\mathrm{id}\right)/l\right\}\\
&= \kappa\cdot\left(\overline{q}\cdot\mathrm{id}\right)\left\{\lambda a.\lambda k.\left(\beta\mapsto k\beta\right)\cdot\left(\overline{P[a]}\cdot\mathrm{id}\right)/l\right\}\\
&= \kappa\cdot\left(\overline{q}\cdot\mathrm{id}\right)\left\{\lambda a.\lambda k.\overline{\#P[a]}\cdot\left(\beta\mapsto k\beta\right)/l\right\}\\
&= \kappa\cdot\left(\overline{q}\cdot\mathrm{id}\right)\left\{\overline{\lambda a.\#P[a]}\cdot\mathrm{id}/l\right\}\\
&= \kappa\cdot\left(\overline{q\{\lambda a.\#P[a]/l\}}\cdot\mathrm{id}\right)\\
&= \overline{\#q\{\lambda a.\#P[a]/l\}}\cdot\kappa.
\end{aligned}$$

Otherwise, when $\#p\to\#r$ because $p\to r$, by the induction hypothesis we have

$$\overline{\#p}\cdot\kappa = \kappa\cdot(\overline{p}\cdot\mathrm{id})\to^{\diamond}\kappa\cdot(\overline{r}\cdot\mathrm{id}) = \overline{\#r}.$$

□

4.4.6. Theorem (Normalisation)**.** *For every closed proof term p_0, such that $\vdash^+ p_0 : A$, there is a finite reduction path $p_0\to p_1\to\ldots\to p_n$ ending with a value p_n.*

PROOF. A step of reduction $p\to r$ in MQC$^+$ translates either into more then one step of reduction $\overline{p}\cdot\kappa\to^+\overline{r}\cdot\kappa$ in MQC, or into definitional equality $\overline{p}\cdot\kappa=\overline{r}\cdot\kappa$, if the reduction was by a rule for reset. However, a reset-reduction "spends" its argument, the # or the $\mathscr{S}$, hence it can occur only a finite number of times *in a row* in a reduction sequence. Therefore, an infinite reduction path in MQC$^+$ would give rise to an infinite reduction path in MQC. □

We are now ready to prove the disjunction and existence properties for the well-delimited derivations of MQC$^+$.

4.4.7. Corollary. *If $\vdash^+ A\vee B$, then $\vdash^+ A$ or $\vdash^+ B$. If $\vdash^+\exists x A(x)$, then there exists a closed term t such that $\vdash^+ A(t)$.*

PROOF. Let $\vdash^+ p : A\vee B$. By Normalisation and Subject reduction, $p\to\cdots\to V$ and $\vdash^+ V : A\vee B$. Since V is a value, V must be of form $\iota_1 V'$ or $\iota_2 V'$, therefore either $\vdash^+ V' : A$ or $\vdash^+ V' : B$. The case for "$\exists$" is analogous. □

4.5. Applications, related and future work

4.5.1. Double negation shift. The first use of a schema equivalent to DNS appears to be in modal logic, by Barcan [**24, 23, 78**], who introduced what is today known as Barcan's formula,

$$\forall x\Box A(x)\to\Box\forall x A(x),$$

or, equivalently,

$$\Diamond\exists x A(x)\to\exists\Diamond A(x).$$

Veldman kindly pointed to us that DNS is also known as Kuroda's Conjecture [**129**]. In [**123**], Kripke showed that Kuroda's Conjecture and Markov's Principle (however, see also [**117**] for criticism of Kripke's argument) are underivable in intuitionistic logic.

In [**118**, Section 2.11], Kreisel used the principle

$$\text{(GMP)} \qquad \neg\forall n A(n) \Rightarrow \exists n \neg A(n),$$

for $A(n)$ an arbitrary formula, to deal with implication while giving a translation of formulae of Analysis into functionals of finite type. In [**148**], Oliva calls this principle the Generalised Markov Principle and remarks that $\mathrm{HA}^\omega \vdash \mathrm{DNS} \leftrightarrow \neg\neg\mathrm{GMP}$. Kreisel does not give a justification of GMP in his paper.

The term "double negation shift" appears for the first time in [**157**] to denote the formula

$$\text{(DNS)} \qquad \forall n \neg\neg A(n) \Rightarrow \neg\neg\forall n A(n).$$

There, Spector builds upon previous works of Gödel [**88, 89, 90**], namely he realises DNS by adding the schema of bar recursion to Gödel's system T. The name "bar recursion" comes from the Bar Principle of Brouwer which is used in justifying it. However, Spector attaches no particular interest to the DNS schema itself; he writes:

> The schema [DNS] is chosen not because we believe it is of intuitionistic significance, but to provide a formal system in which classical analysis is easily interpreted, and whose logical basis is intuitionistic.[**157**]

We treat DNS at the level of predicate logic, not of arithmetic, and we hope to explore in the future the status of this change.

4.5.2. Negative translation of Countable Choice. The Axiom of Countable Choice is,

$$\text{(AC}_0\text{)} \qquad \forall x^0 \exists y^\rho A(x,y) \Rightarrow \exists f^{0\to\rho} \forall x^0 A(x, f(x)),$$

a formula schema of HA^ω, Heyting Arithmetic in all finite types. The type 0 stands for the set of natural numbers $\mathbb{N}$, the type $1 = 0 \to 0$ stands for the functions $\mathbb{N} \to \mathbb{N}$, 2 stands for the functionals $(0 \to 0) \to 0$, and so on. Spector showed that the Kuroda [**114**, p.163] negative translation, $\neg\neg(\mathrm{AC}_0{}^*)$, of AC_0,

$$\text{(AC}_0^{\mathrm{N}}\text{)} \qquad \forall x^0 \neg\neg \exists y^\rho A^*(x,y) \Rightarrow \exists f^{0\to\rho} \forall x^0 \neg\neg A^*(x, f(x)),$$

is provable from DNS and the intuitionistic AC_0. Since AC_0 is realisable in HA^ω, and DNS is realisable by bar recursion, so is $\mathrm{AC}_0^{\mathrm{N}}$. His approach was extended to Dependent Choice (DC) by Luckhardt [**132**] and Howard [**106**]. In recent years, Kohlenbach, Berger, and Oliva gave their own versions of bar recursion (see [**30**] for a comparison).

Since we treat DNS at the level of logic, we are only able to give an *open* proof term deriving the negative translation of AC_0,

$$\text{(AC}_{0\,T}\text{)} \qquad \forall x^0 \neg_T\neg_T \exists y^\rho A_T(x,y) \Rightarrow \neg_T\neg_T \exists f^{0\to\rho} \forall x^0 \neg_T\neg_T A_T(x, f(x)).$$

Given a variable c to denote a proof of the intuitionistic AC_0, we can use a proof term similar to the one of DNS_T for deriving the above schema:

$$\lambda a.\lambda k.\#k(c(\lambda x.\mathscr{S} k'.ax(\lambda d.k'(\nu d)))),$$

where ν is a proof term for $\exists y A_T(x,y) \Rightarrow \exists y A^T(x,y)$.

The proof term being open means that we can not immediately use it for computation. We would have to either develop a realisability semantics for MQC^+,

or add delimited control to a system with strong existential quantifiers, like Martin-Löf type theory, which can derive AC_0. To us, the second approach seems more attractive, but we do not pursue it in any more detail than what is given in the next subsection.

4.5.3. $\mathbf{AC}_0^{\mathbf{N}}$ and strong existential quantifiers. Herbelin has already looked into the behaviour of non-delimited control operators (call/cc) in the context of strong existentials (Σ-types). In [**99**] he shows that: 1) in the absence of universes, Martin-Löf type theory becomes "proof irrelevant" with the addition of call/cc; 2) in the presence of universes, adding call/cc makes it inconsistent.

We explain briefly the reasons for this behaviour. Namely, a proof using call/cc is allowed to "lie" a number of times, before giving a valid proof. For example, to prove $\exists x.2 = x$, using call/cc, one can build a proof term which first gives the number 3 as the witness, and afterwards gives the right witness, 2. Then, if one defines the reduction rules for call/cc to commute with any constructor preceding it, as usually done, we can derive a proof of $2 = 3$.

However, with delimited control, the control operator does not simply commute with every constructor before it; it only does so within a pure evaluation context delimited by a reset. For example, let us imagine that we have a Martin-Löf type theory extended with shift/reset, and let wit and prf denote the first and second projection of Σ-types. The proof term for AC_0 would then be:

$$c := \lambda b.(\lambda x.\mathsf{wit}(bx), \lambda x.\mathsf{prf}(bx)).$$

If we substitute it for the open variable c in the proof term for $\mathrm{AC}_0^{\mathrm{N}}$,

$$\lambda a.\lambda k.\#k(c(\lambda x.\mathcal{S}k'.ax(\lambda d.k'(vd)))),$$

we would get

$$\lambda a.\lambda k.\#k\left(\lambda x.\mathsf{wit}\left(\mathcal{S}k'.ax\left(\lambda d.k'\left(vd\right)\right)\right), \lambda x.\mathsf{prf}\left(\mathcal{S}k'.ax\left(\lambda d.k'\left(vd\right)\right)\right)\right).$$

We see that shift can not by itself immediately commute with wit and prf, regardless of the content supplied for the parameters a and k, as call/cc would do. More work is needed in order to see whether we can add delimited control to Martin-Löf type theory.

4.5.4. Other realisers for $\mathbf{AC}_0^{\mathbf{N}}$. Berardi, Bezem and Coquand [**28**] gave another realisability interpretation of $\mathrm{AC}_0^{\mathrm{N}}$ using Gödel's system T extended with a kind of "store" computational effect. Krivine [**127**] gave an interpretation of the negative translation of dependent choice based on a λ-calculus extended with the undelimited[5] control operator call/cc.

Our work is similar to [**28**] and [**127**] in that it uses a purely functional language extended by some kind of computational effect, however, delimited control is known to be able to simulate computationally both call/cc and store [**74**], while neither call/cc nor store alone can simulate delimited control. Actually, a *combination* of call/cc, store, and exceptions *can* simulate delimited control [**74**].

[5] Strictly speaking, being undelimited is not a property of call/cc itself, but so far all implementations of call/cc inside programing languages treat call/cc as undelimited. It is possible to add a delimiter to call/cc also, see for example [**11, 13**].

4.5.5. Negative translation of Dependent Choice. We mentioned that the approach of Spector had been extended to handling the negative translation of the schema of Dependent Choice,

$$\text{(DC)} \qquad \forall x^0 \forall y^\rho \exists z^\rho . A(x,y,z) \Rightarrow \exists f^{0\to\rho} \forall x^0 A(x, fx, f(x+1)).$$

An axiom schema similar in shape to DNS is used for this purpose (see [**114**, p.208]),

$$\forall x^0 \forall c^\rho \neg\neg \exists a^\rho \forall b^\tau A_0(x,c,a,b) \Rightarrow \neg\neg \exists u^{0\to\rho} \forall y^0 \forall v^\tau A_0(y, uy, u(y+1), v),$$

where A_0 is a decidable formula. We have not yet pursued the goal of treating this schema in our system.

4.5.6. Fan Theorem. Another attractive direction for further work concerns principles arising form Brouwer's Bar Principle. It is folklore that all practical consequences of the Bar Principle are already consequences of the Fan Principle,

$$\text{(FAN)} \qquad \forall \alpha^1 \exists n^0 A_0(\overline{\alpha} n) \Rightarrow \exists N^0 \forall \alpha^1 \exists n \leq N . A_0(\overline{\alpha} n),$$

where $A_0(x,y)$ is a decidable formula, and $\overline{\alpha} n$ denotes the initial segment of length n of the infinite stream α.

We started to work on giving a proof term for FAN, but due to lack of time we had to stop. Our approach was to restate FAN, so that the right hand side of the implication becomes a Σ-formula:

$$\text{(FAN')} \qquad \forall \alpha^1 \exists n^0 A_0(\overline{\alpha} n) \Rightarrow \exists N^0 \forall m^0 < 2^N \exists n \leq N . A_0(\tilde{m} n),$$

where $\tilde{m}n$ turns a natural number m into its binary representation, appending zeros to it if the binary representation has less than n bits. Since the internal quantification over α is bounded, we can work in a bounded arithmetic system like $G_3 A_i^\omega$ (which is strong enough to encode variable-length finite sequences [**114**, p.54]) and consider the right-hand side of the implication as a Σ-formula.

Bauer [**26**] reported to have used, together with Pretnar, delimited control to give an alternative implementation of Escardó's fan functional [**65**] for exhaustive search of Cantor space.

4.5.7. Open Induction and Bar Induction. In the context of Intuitionistic Reverse Mathematics, Veldman considered the relationship between various intuitionistic principles [**177, 176**]. In particular, he recovered a result of Moschovakis and Solovay [**142**], that

$$\text{BI} + \text{MP} \vdash \text{DNS}.$$

Besides other equivalences, he also connected the Open Induction Principle [**50**] on Cantor space (OI($\mathscr{C}$)) to the DNS schema.

Definition (OI($\mathscr{C}$))**.** For any U open subspace of Cantor space $2^{\mathbb{N}}$ and $<$ the lexicographical order on $2^{\mathbb{N}}$, if

$$\forall \alpha . (\forall \beta < \alpha . \beta \in U) \to \alpha \in U$$

then

$$\forall \alpha . \alpha \in U.$$

He shows that

$$\mathrm{MP} \vdash \mathrm{DNS} \Leftrightarrow \mathrm{OI}(\mathscr{C}),$$

hence

$$\mathrm{DNS} + \mathrm{MP} \Leftrightarrow \mathrm{OI}(\mathscr{C}) + \mathrm{MP},$$

and also that FAN is strictly weaker than $\mathrm{OI}(\mathscr{C})$, by giving a proof from $\mathrm{OI}(\mathscr{C})$ of the Paris-Harrington theorem [**150**], which is independent of HA, while it was established by Troelstra [**160**] that FAN is conservative over HA.

4.5.8. Shift and reset. The shift/reset delimited control operators have been introduced in [**58, 59**]. The purpose for introducing them was to exploit the expressive power beyond the traditional CPS translation [**151**]; traditionally, programs written in continuation passing style are of a specific form, all function applications calls are tail applications, but one can write programs in CPS that make non necessarily tail calls[6], and then the question arises what are the original direct style operators that are translated into this extended-CPS form. Danvy and Filinski's answer is shift/reset. They also generalise this procedure, making a CPS translation of the already extended-CPS translated terms, and get a hierarchy of $\mathrm{shift}_n/\mathrm{reset}_n$ operators, level n corresponding to the n-th extended-CPS translation.

A typing system (known as "type-and-effect" system) for shift/reset appears already in [**58**], but it is difficult to connect to traditional logic because implication is a quaternary not a binary connective. A typing system which is a specialisation of Danvy-Filinski's, but again has a ternary implication connective, appears in [**144**]. In his paper on simulating monadic effects [**73**], Filinski uses a typing system similar to ours, where implication is binary and the answer type T is fixed.

Our work is different in that we look at the typing system in the context of logic, with all usual connectives, not in the context of programming languages. Also, our proofs of subject reduction and progress are direct, while for Murthy's and Filinski's system these properties follow [**35**] from the embedding in Danvy-Filinski's system.

Asai-Kameyama [**14**] revisit the type-and-effect system of Danvy-Filinski and extend it to a polymorphic one. Kiselyov-Shan give a typing system expressing a sub-structural logic, and small-step operational semantics (we give a big-step semantics) for shift/reset. Zeilberger[**186**] investigates delimited control in the context of polarised linear logic. Munch-Maccagnoni connects delimited control to co-dereliction of differential linear logic. This list of related work references is far from exhaustive.

4.5.9. Felleisen's control/prompt delimited control operators. There is a variety of delimited control operators in the programming languages literature. One of the earliest ones are Felleisen's control ($\mathscr{F}$) and prompt (#). Their development was not motivated by the study of the extended CPS translation, but directly by the programming need of being able to compose continuations like any other first class object inside a programing language (Scheme). The relationship between control/prompt and shift/reset has been studied by many authors. Although shift/reset and control/prompt are mutually interpretable, and

[6] Indeed, the completeness proof of Chapter 3 uses non-tail calls in the case of disjunction.

their big-step operational semantics is quite similar,

$$\#P[\mathscr{S}k.p] \to \#p\{(\lambda a.\#P[a])/k\}$$
$$\#P[\mathscr{F}k.p] \to \#p\{(\lambda a.P[a])/k\},$$

operationally they behave differently: while shift sets a new control delimiter around its former pure evaluation context, control does not. The PhD thesis of Biernacki [**35**] contains a thorough comparison of shift and control, and further references.

In connection to MQC^+, we have tried to check all the proofs with having $\mathscr{F}$ instead of $\mathscr{S}$. While most of the time the difference does not matter, in the subject reduction theorem, when using control, we do not have to carry out a weakening step as with shift, but for the normalisation theorem it is not clear which two-level CPS translation to take if we had control instead of shift.

4.5.10. Herbelin's calculus for Markov Principle. In [**100**], Herbelin presents $\mathrm{IQC}_{\mathrm{MP}}$, an intuitionistic calculus that derives Markov's Principle. The calculus MQC^+ that we present here has been derived starting from Herbelin's calculus. There are two big differences between the two:

(1) Derivations of $\mathrm{IQC}_{\mathrm{MP}}$ are annotated by a *context* of Σ-formula, not just one formula. This permits, for example, to have a derivation which uses multiple and different instances of Markov's Principle.

We had also tried to have a version of MQC^+ with context annotations, but encountered a problem with the CPS translation. Namely, when multiple Σ-formula annotations are present, one has to have a uniform way to compose the monads $\Gamma_S \vdash A^S, \Gamma_T \vdash A^T, \ldots$ indexed with different Σ-formulae $S, T, \ldots$. We hope to address this problem in the future.

(2) The second difference with $\mathrm{IQC}_{\mathrm{MP}}$ is in the rules for the delimited control operators:

$$\frac{\Gamma \vdash_{\alpha:T,\Delta} p : T}{\Gamma \vdash_\Delta \mathsf{catch}_\alpha\, p : T}\ \textsc{Catch} \qquad \frac{\Gamma \vdash_\Delta p : T \quad (\alpha : T) \in \Delta}{\Gamma \vdash_\Delta \mathsf{throw}_\alpha\, p : A}\ \textsc{Throw}$$

While catch is just $\#$, the proof term $\mathsf{throw}\ p$ is a particular case of $\mathscr{S}k.p$ that does not use the continuation k inside p.

Note that if we had context- and not single-formula-annotations for MQC^+, then we would have the following characterisation of provability of Σ-formulas S:

$$\begin{array}{ccccc} \vdash^+ S & \xrightarrow{4.2.4} & \vdash^i S^\perp & \xleftrightarrow{4.2.8} & \vdash^c S \\ \uparrow & & \Vert\ \text{by def. of } (\cdot)^\perp & & \\ \mathrm{MP}_\perp \vdash^i S & \longleftarrow & \vdash^i \neg\neg S & & \end{array}$$

4.5.11. Completeness for standard Kripke models. We started to investigate the logical meaning of delimited control because of Danvy's program of Section 3.2, that is, because of the possibility of a constructive completeness proof for full intuitionistic logic and *standard* Kripke semantics.

May we now close the circle and use MQC^+ to give such a proof? We believe that further work is necessary: 1) on having multiple Σ-formula annotations, like

in the system of Herbelin described in Subsection 4.5.10; 2) on allowing a certain degree of "polymorphism" in the rules for shift and reset.

If we go back to the the algorithm of page 63, we see that, in a completeness proof having as base the given algorithm, the reset in the implication-reification case would be used for the Σ-formula

$$\exists q.\, \Gamma, a : \tau \vdash q : \sigma,$$

and shift would be used for a Σ-formula of the same form, $\exists r.\, \Gamma \vdash r : \rho$, which means that the continuation bound by the shift would have the type

$$\kappa : \forall \Gamma' \geq \Gamma.\ \left(\Gamma' \Vdash \sigma \text{ or } \Gamma' \Vdash \tau \to \exists r.\, \Gamma' \vdash r : \rho\right).$$

Note that it is not possible to use the simpler typing of κ,

$$\kappa : \Gamma \Vdash \sigma \text{ or } \Gamma \Vdash \tau \to \exists r.\, \Gamma \vdash r : \rho,$$

because the continuation is applied in two contexts, inside the `case` expression, that are enriched by a_1 and a_2; the reset-s are there applied for the Σ-formulae

$$\exists r.\, \Gamma, a_1 : \tau \vdash r : \rho \quad \text{ and } \quad \exists r.\, \Gamma, a_2 : \sigma \vdash r : \rho.$$

Then, it is clear that the shape of the formula that completeness (or reification) is applied to, determines how many Σ-formulae we would need to annotate our derivations with. It is also clear that we have to account somehow for the polymorphic behaviour of the contexts. We thus propose the following rules for a future extension of MQC$^+$,

$$\frac{\Gamma \vdash_{(\alpha:T),\Delta} p : T(w,\phi)}{\Gamma \vdash_\Delta \#_\alpha p : T(w,\phi)}\ \#_\alpha$$

$$\frac{\Gamma, k : \forall w' \geq w.A(w',\phi) \Rightarrow T(w',\phi) \vdash_\Delta p : T(w,\phi) \quad (\alpha : T(w,\phi)) \in \Delta}{\Gamma \vdash_\Delta \mathscr{S}_\alpha k p : A(w,\phi)}\ \mathscr{S}_\alpha,$$

which would be able to prove completeness for standard Kripke models if we let $T(w,\phi)$ encode intuitionistic provability of formula ϕ in the context w, $T(w,\phi) := \exists n.\, \ulcorner w \vdash n : \sigma \urcorner$, and let $A(w,\phi)$ represent Kripke-validity of formula ϕ in the context w, $A(w,\phi) := w \Vdash \phi$.

If the preorder between the individuals is to be avoided, we could use its definability from page 70, to replace

$$k : \forall w' \geq w.A(w',\phi) \Rightarrow T(w',\phi)$$

by

$$k : \forall w'.\left[\forall \psi.T(w,\psi) \Rightarrow T(w',\psi)\right] \Rightarrow A(w',\phi) \Rightarrow T(w',\phi).$$

4.5.12. Normalisation-by-evaluation for MQC$^+$. The pass to the 2-level λ-calculus in order to give a simpler proof of reduction-preservation by the CPS translation, indicates that there should be an NBE algorithm behind, which would express the normalisation proof in two distinct phases (reflection and reification). In future, we hope to be able to develop such a proof, by modifying the notion of model from Chapter 3.

A work-in-progress towards an NBE program for shift/reset has been presented in [**166**].

4.5.13. Monadic computational effects. Finally, we would like to see what specific monadic effects look like when expressed in a direct way, using shift/reset, in our system. The PhD thesis of Filinski [**74**] is a guide on how to do that.

Bibliography

[1] *Proceedings, Sixth Annual IEEE Symposium on Logic in Computer Science, 15-18 July, 1991, Amsterdam, The Netherlands*. IEEE Computer Society, 1991.

[2] *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*. IEEE Computer Society, 2007.

[3] Andreas Abel. Weak beta-theta-normalization and normalization by evaluation for system F. In Cervesato et al. [**45**], pages 497–511.

[4] Andreas Abel. Typed applicative structures and normalization by evaluation for system F^{ω}. In Grädel and Kahle [**91**], pages 40–54.

[5] Andreas Abel. Towards normalization by evaluation for the *beta-eta*-calculus of constructions. In Blume et al. [**36**], pages 224–239.

[6] Andreas Abel, Klaus Aehlig, and Peter Dybjer. Normalization by evaluation for Martin-Löf type theory with one universe. *Electr. Notes Theor. Comput. Sci.*, 173:17–39, 2007.

[7] Andreas Abel, Thierry Coquand, and Peter Dybjer. Normalization by evaluation for Martin-Löf type theory with typed equality judgements. In *LICS* [**2**], pages 3–12.

[8] Andreas Abel, Thierry Coquand, and Peter Dybjer. Normalization by evaluation for Martin-Löf type theory with typed equality judgements. In *LICS* [**2**], pages 3–12.

[9] Andreas Abel, Thierry Coquand, and Peter Dybjer. Verifying a semantic beta-eta-conversion test for Martin-Löf type theory. In Audebaud and Paulin-Mohring [**15**], pages 29–56.

[10] Thorsten Altenkirch, Peter Dybjer, Martin Hofmann, and Philip J. Scott. Normalization by evaluation for typed lambda calculus with coproducts. In *LICS*, pages 303–310, 2001.

[11] Zena M. Ariola and Hugo Herbelin. Minimal classical logic and control operators. In *Thirtieth International Colloquium on Automata, Languages and Programming, ICALP '03, Eindhoven, The Netherlands, June 30 - July 4, 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 871–885. Springer, 2003.

[12] Zena M. Ariola and Hugo Herbelin. Control reduction theories: the benefit of structural substitution. *J. Funct. Program.*, 18(3):373–419, 2008.

[13] Zena M. Ariola, Hugo Herbelin, and Amr Sabry. A type-theoretic foundation of delimited continuations. *Higher Order and Symbolic Computation*, 22(3):233–273, September 2009. online from 2007.

[14] Kenichi Asai and Yukiyoshi Kameyama. Polymorphic delimited continuations. In *APLAS*, pages 239–254, 2007.

[15] Philippe Audebaud and Christine Paulin-Mohring, editors. *Mathematics of Program Construction, 9th International Conference, MPC 2008, Marseille, France, July 15-18, 2008. Proceedings*, volume 5133 of *Lecture Notes in Computer Science*. Springer, 2008.

[16] Jeremy Avigad. Algebraic proofs of cut elimination. *J. Log. Algebr. Program.*, 49(1-2):15–30, 2001.

[17] Jeremy Avigad. A variant of the double-negation translation. Technical report, Carnegie Mellon University, 2006. Technical Report CMU-PHIL 179.

[18] Brian E. Aydemir, Arthur Charguéraud, Benjamin C. Pierce, Randy Pollack, and Stephanie Weirich. Engineering formal metatheory. In Necula and Wadler [**145**], pages 3–15.

[19] Vincent Balat. *Une étude des sommes fortes : isomorphismes et formes normales*. PhD thesis, Université Paris 7, 2002.

[20] Vincent Balat. Keeping sums under control, 2004.

[21] Vincent Balat. Ocaml implementation of tdpe for sums, 2004.

[22] Vincent Balat, Roberto Di Cosmo, and Marcelo P. Fiore. Extensional normalisation and type-directed partial evaluation for typed lambda calculus with sums. In Jones and Leroy [**110**], pages 64–76.

[23] Ruth C. Barcan. The deduction theorem in a functional calculus of first order based on strict implication. *The Journal of Symbolic Logic*, 11(4):115–118, 1946.

[24] Ruth C. Barcan. A functional calculus of first order based on strict implication. *The Journal of Symbolic Logic*, 11(1):1–16, 1946.

[25] Freiric Barral. Exceptional nbe for sums. In Olivier Danvy, editor, *Informal proceedings of the 2009 Workshop on Normalization by Evaluation, August 15th 2009, Los Angeles, California*, pages 21–30, 2009.

[26] Andrej Bauer. personal communication, March 2010.

[27] Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors. *Logical Approaches to Computational Barriers, Second Conference on Computability in Europe, CiE 2006, Swansea, UK, June 30-July 5, 2006, Proceedings*, volume 3988 of *Lecture Notes in Computer Science*. Springer, 2006.

[28] Stefano Berardi, Marc Bezem, and Thierry Coquand. On the computational content of the axiom of choice. *J. Symbolic Logic*, 63(2):600–622, 1998.

[29] Stefano Berardi and Silvio Valentini. Krivine's intuitionistic proof of classical completeness (for countable languages). *Ann. Pure Appl. Logic*, 129(1-3):93–106, 2004.

[30] U. Berger and P. Oliva. Modified bar recursion and classical dependent choice. In M. Baaz, S.D. Friedman, and J. Kraijcek, editors, *Logic Colloquium '01, Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic, held in Vienna, Austria, August 6 - 11, 2001*, volume 20 of *Lecture Notes in Logic*, pages 89–107. Springer, 2005.

[31] Ulrich Berger. A computational interpretation of open induction. In F. Titsworth, editor, *Proceedings of the Nineteenth Annual IEEE Symposium on Logic in Computer Science*, pages 326–334. IEEE Computer Society, 2004.

[32] Ulrich Berger, Stefan Berghofer, Pierre Letouzey, and Helmut Schwichtenberg. Program extraction from normalization proofs. In *Typed Lambda Calculi and Applications, number 664 in Lecture Notes in Computer Science*, pages 91–106. Springer Verlag, 1993.

[33] Ulrich Berger and Helmut Schwichtenberg. An inverse of the evaluation functional for typed lambda-calculus. In *LICS* [**1**], pages 203–211.

[34] Paul Bernays. Axiomatische untersuchung des aussagen-kalkuls der *principia mathematica*. *Mathematische Zeitschrift*, (25):305–320, 1926.

[35] Dariusz Biernacki. *The Theory and Practice of Programming Languages with Delimited Continuations*. PhD thesis, DAIMI, Department of Computer Science, University of Aarhus, Denmark, 12 2005. BRICS DS-05-08.

[36] Matthias Blume, Naoki Kobayashi, and Germán Vidal, editors. *Functional and Logic Programming, 10th International Symposium, FLOPS 2010, Sendai, Japan, April 19-21, 2010. Proceedings*, volume 6009 of *Lecture Notes in Computer Science*. Springer, 2010.

[37] Patrick Braselmann and Peter Koepke. Coincidence lemma and substitution lemma. *Formalized Mathematics*, 13(1):17–26, 2005.

[38] Patrick Braselmann and Peter Koepke. Consequences of the sequent calculus. *Formalized Mathematics*, 13(1):41–44, 2005.

[39] Patrick Braselmann and Peter Koepke. Equivalences of inconsistency and Henkin models. *Formalized Mathematics*, 13(1):45–48, 2005.

[40] Patrick Braselmann and Peter Koepke. Gödel's completeness theorem. *Formalized Mathematics*, 13(1):49–53, 2005.

[41] Patrick Braselmann and Peter Koepke. A sequent calculus for first-order logic. *Formalized Mathematics*, 13(1):33–39, 2005.

[42] Patrick Braselmann and Peter Koepke. Substitution in first-order formulas: Elementary properties. *Formalized Mathematics*, 13(1):5–15, 2005.

[43] Patrick Braselmann and Peter Koepke. Substitution in first-order formulas. part ii. the construction of first-order formulas. *Formalized Mathematics*, 13(1):27–32, 2005.

[44] Stanley Burris and Simon Lee. Tarski's high school identities. *The American Mathematical Monthly*, 100(3):231–236, 1993.

[45] Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors. *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*, volume 5330 of *Lecture Notes in Computer Science*. Springer, 2008.

[46] Alonzo Church. An unsolvable problem of elementary number theory. *American journal of mathematics*, (58):345–363, 1936.

[47] Mario Coppo, Elena Lodi, and G. Michele Pinna, editors. *Theoretical Computer Science, 9th Italian Conference, ICTCS 2005, Siena, Italy, October 12-14, 2005, Proceedings*, volume 3701 of *Lecture Notes in Computer Science*. Springer, 2005.

[48] Catarina Coquand. From semantics to rules: A machine assisted analysis. In *CSL '93*, volume 832 of *Lecture Notes in Computer Science*, pages 91–105. Springer, 1993.

[49] Catarina Coquand. A formalised proof of the soundness and completeness of a simply typed lambda-calculus with explicit substitutions. *Higher Order Symbol. Comput.*, 15(1):57–90, 2002.

[50] Thierry Coquand. A note on the open induction principle, 1997.

[51] Thierry Coquand and Peter Dybjer. Intuitionistic model constructions and normalization proofs. *Mathematical Structures in Computer Science*, 7(1):75–94, 1997.

[52] John Corcoran, editor. *Logic, Semantics, Metamathematics, papers from 1923 to 1938*. Hackett Publishing Company, Indianapolis, 1983.

[53] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In *ICFP*, pages 233–243, 2000.

[54] Bernd I. Dahn. Constructions of classical models by means of Kripke models (survey). *Studia Logica*, 38(4):401–405, 1979.

[55] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. Computational isomorphisms in classical logic. *Theor. Comput. Sci.*, 294(3):353–378, 2003.

[56] Olivier Danvy. Pragmatics of type-directed partial evaluation. In *Selected Papers from the Internaltional Seminar on Partial Evaluation*, pages 73–94, London, UK, 1996. Springer-Verlag.

[57] Olivier Danvy. Type-directed partial evaluation. In *POPL*, pages 242–257, 1996.

[58] Olivier Danvy and Andrzej Filinski. A functional abstraction of typed contexts. Technical report, Computer Science Department, University of Copenhagen, 1989. DIKU Rapport 89/12.

[59] Olivier Danvy and Andrzej Filinski. Abstracting control. In *LISP and Functional Programming*, pages 151–160, 1990.

[60] Olivier Danvy and Andrzej Filinski. Representing control: A study of the CPS transformation. *Mathematical Structures in Computer Science*, 2(4):361–391, 1992.

[61] Martin Davis, editor. *The undecidable: basic papers on undecidable propositions, unsolvable problems, and computable functions*. Raven Press, Hewlett, N.Y., 1965.

[62] The Coq development team. The Coq proof assistant v8.2 reference manual. available at `http://coq.inria.fr`, 2009.

[63] Peter Dybjer and Andrzej Filinski. Normalization and partial evaluation, 2002.

[64] V. H. Dyson and G. Kreisel. Analysis of Beth's semantic construction of intuitionistic logic. Technical report, Applied mathematics and statistical laboratories, Stanford University, 1961. Technical Report 3.

[65] Martín Hötzel Escardó. Infinite sets that admit fast exhaustive search. In *LICS*, pages 443–452, 2007.

[66] Solomon Feferman, editor. *Collected works. Publications 1929–1936*, volume 1. The Clarendon Press Oxford University Press, New York, 1986.

[67] Solomon Feferman, editor. *Collected works. Publications 1938–1974*, volume II. The Clarendon Press Oxford University Press, New York, 1990.

[68] Solomon Feferman, Jr. John W. Dawson, Warren Goldfarb, Charles Parsons, and Robert M. Solovay, editors. *Collected works. Unpublished essays and lectures*, volume III. The Clarendon Press Oxford University Press, New York, 1995.

[69] Matthias Felleisen. The theory and practice of first-class prompts. In *POPL*, pages 180–190, 1988.

[70] Matthias Felleisen, Daniel P. Friedman, Eugene E. Kohlbecker, and Bruce F. Duba. Reasoning with continuations. In *LICS*, pages 131–141, 1986.

[71] Matthias Felleisen, Daniel P. Friedman, Eugene E. Kohlbecker, and Bruce F. Duba. Beyond continuations. Technical report, Indiana University, 1987.

[72] Matthias Felleisen, Daniel P. Friedman, Eugene E. Kohlbecker, and Bruce F. Duba. A syntactic theory of sequential control. *Theor. Comput. Sci.*, 52:205–237, 1987.

[73] Andrzej Filinski. Representing monads. In *Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 446–457, 1994.

[74] Andrzej Filinski. *Controlling Effects*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1996. Technical Report CMU-CS-96-119 (144pp.).

[75] Andrzej Filinski and Henning Korsholm Rohde. A denotational account of untyped normalization by evaluation. In Walukiewicz [**178**], pages 167–181.

[76] Marcelo P. Fiore, Roberto Di Cosmo, and Vincent Balat. Remarks on isomorphisms in typed lambda calculi with empty and sum types. *Ann. Pure Appl. Logic*, 141(1-2):35–50, 2006.

[77] Melvin Fitting. *Intuitionistic Logic, Model Theory, and Forcing*. North-Holland Publishing Co., 1969.

[78] Melvin Fitting. Barcan both ways, 1997.

[79] Gerhard Gentzen. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Math. Ann.*, 112(1):493–565, 1936.

[80] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50(1):1–102, 1987.

[81] Valery Ivanovich Glivenko. Sur quelques points de la logique de M. Brouwer. In *Bulletins de la classe des sciences*, volume 15 of *5*, pages 183–188. Academie Royale de Belgique, 1929.

[82] Kurt Gödel. *On the completeness of the calculus of logic*, pages 61–101. Volume 1 of Feferman [**66**], 1929. PhD thesis.

[83] Kurt Gödel. *The completeness of the axioms of the functional calculus of logic*, pages 103–123. Volume 1 of Feferman [**66**], 1930.

[84] Kurt Gödel. Lecture on completeness of the functional calculus. In Feferman et al. [**68**], pages 17–29.

[85] Kurt Gödel. On formally undecidable propositions of *principia mathematica* and related systems. In Feferman [**66**], pages yyy–xxx.

[86] Kurt Gödel. *An interpretation of the intuitionistic propositional calculus*, pages 301–303. Volume 1 of Feferman [**66**], 1933.

[87] Kurt Gödel. Zur intuitionistischen Arithmetik und Zahlentheorie. *Ergebnisse eines mathematischen Kolloquiums*, 4:34–38, 1933.

[88] Kurt Gödel. *In what sense is intuitionistic logic constructive*, pages 189–200. Volume III of Feferman et al. [**68**], 1941. early lecture on the Dialectica interpretation.

[89] Kurt Gödel. *On a hitherto unutilized extension of the finitary standpoint*, pages 241–251. Volume II of Feferman [**67**], 1958.

[90] Kurt Gödel. *On an extension of finitary mathematics which has not yet been used*, pages 271–280. Volume II of Feferman [**67**], 1972.

[91] Erich Grädel and Reinhard Kahle, editors. *Computer Science Logic, 23rd international Workshop, CSL 2009, 18th Annual Conference of the EACSL, Coimbra, Portugal, September 7-11, 2009. Proceedings*, volume 5771 of *Lecture Notes in Computer Science*. Springer, 2009.

[92] Timothy Griffin. A formulae-as-types notion of control. In *POPL*, pages 47–58, 1990.

[93] R. Gurevič. Equational theory of positive numbers with exponentiation. *Proceedings of the American Mathematical Society*, 94(1):135–141, 1985.

[94] Leon Henkin. The completeness of the first-order functional calculus. *Journal of symbolic logic*, (14):159–166, 1949.

[95] Leon Henkin. The discovery of my completeness proofs. *Bulletin of Symbolic Logic*, 2(2):127–158, 1996.

[96] Hugo Herbelin. A lambda-calculus structure isomorphic to Gentzen-style sequent calculus structure. In *CSL '94*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 1994.

[97] Hugo Herbelin. *Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*. Ph.D. thesis, Université Paris 7, Jan. 1995.

[98] Hugo Herbelin. *C'est maintenant qu'on calcule: au coeur de la dualité*. Habilitation thesis, University Paris 11, Dec. 2005.

[99] Hugo Herbelin. On the degeneracy of sigma-types in presence of computational classical logic. In Urzyczyn [**170**], pages 209–220.

[100] Hugo Herbelin. An intuitionistic logic that proves Markov's principle. In *Proceedings, 25th Annual IEEE Symposium on Logic in Computer Science (LICS '10), Edinburgh, UK, 11-14 July 2010*, page N/A. IEEE Computer Society Press, 2010.

[101] Hugo Herbelin and Silvia Ghilezan. An approach to call-by-name delimited continuations. In Necula and Wadler [**145**], pages 383–394.

[102] Hugo Herbelin and Gyesik Lee. Forcing-based cut-elimination for Gentzen-style intuitionistic sequent calculus. In Hiroakira Ono, Makoto Kanazawa, and Ruy J. G. B. de Queiroz, editors, *WoLLIC*, volume 5514 of *Lecture Notes in Computer Science*, pages 209–217. Springer, 2009.

[103] Arend Heyting, editor. *Constructivity in Mathematics, Proceedings of the colloqium held at Amsterdam, 1957*, Studies in Logic and The Foundations of Mathematics. North-Holland Publishing Company Amsterdam, 1959.

[104] David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik*. Springer, Berlin, 1928.

[105] Wilfrid Hodges. Tarski's truth definitions. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition, 2008.

[106] W. A. Howard. Functional interpretation of bar induction by bar recursion. *Compositio Math.*, 20:107–124 (1968), 1968.

[107] W. A. Howard and G. Kreisel. Transfinite induction and bar induction of types zero and one, and the role of continuity in intuitionistic analysis. *The Journal of Symbolic Logic*, 31(3):325–358, 1966.

[108] Joe Hurd and Thomas F. Melham, editors. *Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005, Oxford, UK, August 22-25, 2005, Proceedings*, volume 3603 of *Lecture Notes in Computer Science*. Springer, 2005.

[109] Danko Ilik, Gyesik Lee, and Hugo Herbelin. Kripke models for classical logic. *Annals of Pure and Applied Logic*, 161(11):1367 – 1378, 2010. Special Issue: Classical Logic and Computation (2008).

[110] Neil D. Jones and Xavier Leroy, editors. *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004, Venice, Italy, January 14-16, 2004*. ACM, 2004.

[111] Yukiyoshi Kameyama. Axioms for control operators in the CPS hierarchy. *Higher-Order and Symbolic Computation*, 20(4):339–369, 2007.

[112] Oleg Kiselyov. Call-by-name linguistic side effects. Slides from ESSLLI 2008 Workshop on Symmetric calculi and Ludics for the semantic interpretation. 4-7 August, 2008. Hamburg, Germany., 2008.

[113] S. C. Kleene. On the interpretation of intuitionistic number theory. *The Journal of Symbolic Logic*, 10(4):109–124, 1945.

[114] U. Kohlenbach. *Applied proof theory: proof interpretations and their use in mathematics*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2008.

[115] Andrey Nikolayevich Kolmogorov. O principe tertium non datur. *Matematiceskij Sbornik*, 32:646–667, 1925. English translation in van Heijenoort 1967, pp. 416–437.

[116] G. Kreisel. A remark on free choice sequences and the topological completeness proofs. *The Journal of Symbolic Logic*, 23(4):369–388, 1958.

[117] G. Kreisel. Review: [Semantical analysis of intuitionistic logic I. by Saul A. Kripke]. *The Journal of Symbolic Logic*, 35(2):330–332, 1970.

[118] Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In Heyting [**103**], pages 101–127.

[119] Georg Kreisel. Elementary completeness properties of intuitionistic logic with a note on negations of prenex formulae. *Journal of Symbolic Logic*, 23(3):317–330, 1958.

[120] Georg Kreisel. On weak completeness of intuitionistic predicate logic. *J. Symb. Log.*, 27(2):139–158, 1962.

[121] Saul Kripke. A completeness theorem in modal logic. *J. Symb. Log.*, 24(1):1–14, 1959.

[122] Saul Kripke. Semantical considerations on modal and intuitionistic logic. *Acta Philos. Fennica*, 16:83–94, 1963.

[123] Saul A. Kripke. Semantical analysis of intuitionistic logic i. In *Formal Systems and Recursive Functions*, pages 92–130. North Holland, 1965.

[124] J. L. Krivine. *Lambda-calculus, types and models*. Ellis Horwood, Upper Saddle River, NJ, USA, 1993.

[125] Jean-Louis Krivine. Une preuve formelle et intuitionniste du théorème de complétude de la logique classique. *Bulletin of Symbolic Logic*, 2(4):405–421, 1996.

[126] Jean-Louis Krivine. Typed lambda-calculus in classical zermelo-frænkel set theory. *Arch. Math. Log.*, 40(3):189–205, 2001.

[127] Jean-Louis Krivine. Dependent choice, 'quote' and the clock. *Theor. Comput. Sci.*, 308(1-3):259–276, 2003.

[128] Jean-Louis Krivine. Algèbres de réalisabilité: un programme pour bien ordonner r. *CoRR*, abs/1002.3438, 2010.

[129] Sigekatu Kuroda. Intuitionistische untersuchungen der formalistischen logik. *Nagoya Mathematical Journal*, (2):35–47, 1951.

[130] Saunders M. Lane. *Categories for the Working Mathematician (Graduate Texts in Mathematics)*. Springer, 2nd edition, 1998.

[131] Stéphane Lengrand. Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. *Electr. Notes Theor. Comput. Sci.*, 86(4), 2003.

[132] Horst Luckhardt. *Extensional Gödel functional interpretation. A consistency proof of classical analysis*. Lecture Notes in Mathematics, Vol. 306. Springer-Verlag, Berlin, 1973.

[133] Damiano Macedonio and Giovanni Sambin. From meta-level to semantics via reflection: a model for basic logic and its extensions. available from the authors.

[134] Damiano Macedonio and Giovanni Sambin. personal communication, April 2009.

[135] D. C. McCarthy. Undecidability and intuitionistic incompleteness. *Journal of Philosophical Logic*, (25):559–565, 1996.

[136] David Charles McCarty. On theorems of Gödel and Kreisel: Completeness and Markov's principle. *Notre Dame Journal of Formal Logic*, 35(1):99–107, 1994.

[137] David Charles McCarty. Intuitionistic completeness and classical logic. *Notre Dame Journal of Formal Logic*, 43(4):243–248, 2002.

[138] Eugenio Moggi. Notions of computation and monads. *Inform. and Comput.*, 93(1):55–92, 1991. Selections from the 1989 IEEE Symposium on Logic in Computer Science.

[139] Joan Rand Moschovakis. Review: [A remark on free choice sequences and the topological completeness proofs. by Georg Kreisel]. *The Journal of Symbolic Logic*, 32(2):283, 1967.

[140] Joan Rand Moschovakis. Review: [Elementary completeness properties of intuitionistic logic with a note on negations of prenex formulae. by Georg Kreisel]. *The Journal of Symbolic Logic*, 32(2):282–283, 1967.

[141] Joan Rand Moschovakis. Review: [On weak completeness of intuitionistic predicate logic. by Georg Kreisel]. *The Journal of Symbolic Logic*, 34(1):119–120, 1969.

[142] Joan Rand Moschovakis. Classical and constructive hierarchies in extended intuitionistic analysis. *The Journal of Symbolic Logic*, 68(3):1015–1043, 2003.

[143] Chetan Murthy. *Extracting Classical Content from Classical Proofs*. PhD thesis, Department of Computer Science, Cornell University, 1990.

[144] Chetan R. Murthy. Control operators, hierarchies, and pseudo-classical type systems: A-translation at work. In *Proceedings of the ACM SIGPLAN Workshop on Continuations CW92*, pages 49–72. Stanford University, 1992. Technical Report STAN-CS-92-1426.

[145] George C. Necula and Philip Wadler, editors. *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*. ACM, 2008.

[146] Russell O'Connor. Essential incompleteness of arithmetic verified by Coq. In Hurd and Melham [**108**], pages 245–260.

[147] Mitsuhiro Okada. A uniform semantic proof for cut-elimination and completeness of various first and higher order logics. *Theor. Comput. Sci.*, 281(1-2):471–498, 2002.

[148] Paulo Oliva. Understanding and using Spector's bar recursive interpretation of classical analysis. In Beckmann et al. [**27**], pages 423–434.

[149] Michel Parigot. Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In *Logic Programming and Automated Reasoning: International Conference LPAR '92 Proceedings, St. Petersburg, Russia*, pages 190–201. Springer-Verlag, 1992.

[150] Jeff Paris and Leo Harrington. A mathematical incompleteness in peano arithmetic. In Jon Barwise, editor, *HANDBOOK OF MATHEMATICAL LOGIC*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 1133 – 1142. Elsevier, 1977.

[151] G. D. Plotkin. Call-by-name, call-by-value and the [lambda]-calculus. *Theoretical Computer Science*, 1(2):125–159, 1975.

[152] Christophe Raffalli and Paul Rozière. Phox. In Wiedijk [**179**], pages 67–71.

[153] Giovanni Sambin. Pretopologies and completeness proofs. *J. Symb. Log.*, 60(3):861–878, 1995.

[154] Alexis Saurin. A hierarchy for delimited continuations in call-by-name. In *Foundations of Software Science and Computational Structures*, pages 374–388.

[155] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard Isomorphism, Volume 149 (Studies in Logic and the Foundations of Mathematics)*. Elsevier Science Inc., New York, NY, USA, 2006.

[156] Matthieu Sozeau and Thorsten Altenkirch. Kripke semantics for simply-typed lambda calculus, 2008. http://mattam.org/research/coq.en.html.

[157] Clifford Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics. In *Proc. Sympos. Pure Math., Vol. V*, pages 1–27. American Mathematical Society, Providence, R.I., 1962.

[158] Alfred Tarski. The concept of truth in the languages of the deductive sciences. In Corcoran [**52**], pages 152–278.

[159] Alfred Tarski and Robert L. Vaught. Arithmetical extensions of relational systems. *Compositio Mathematica*, (13):81–102, 1956.

[160] A. S. Troelstra. Note on the fan theorem. *The Journal of Symbolic Logic*, 39(3):584–596, 1974.

[161] A. S. Troelstra and D. van Dalen. *Constructivism in mathematics. Vol. I*, volume 121 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1988. An introduction.

[162] A. S. Troelstra and D. van Dalen. *Constructivism in mathematics. Vol. II*, volume 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1988. An introduction.

[163] Anne Sjerp Troelstra and Paul van Ulsen. The discovery of E.W. Beth's semantics for intuitionistic logic. available online.

[164] A.S. Troelstra. History of constructivism in the 20th century.

[165] Andrzej Trybulec. Mizar. In Wiedijk [**179**], pages 20–23.

[166] Kanae Tsushima and Kenichi Asai. Towards type-directed partial evaluation for shift and reset. In *Informal proceedings of the 2009 Workshop on Normalization by Evaluation*, 2009. `http://www.brics.dk/~danvy/NBE09/informal-proceedings/`.

[167] Alan M. Turing. *On computable numbers, with an application to the Entscheidungsproblem*, volume 2, pages 230–265. 1937. correction in ibid. 43, 544–546.

[168] Christian Urban. *Classical Logic and Computation*. Ph.D. thesis, University of Cambridge, October 2000.

[169] Christian Urban and Diana Ratiu. Classical logic is better than intuitionistic logic: A conjecture about double-negation translations, 2006. http://www.doc.ic.ac.uk/~svb/CLaC06/programme.html.

[170] Pawel Urzyczyn, editor. *Typed Lambda Calculi and Applications, 7th International Conference, TLCA 2005, Nara, Japan, April 21-23, 2005, Proceedings*, volume 3461 of *Lecture Notes in Computer Science*. Springer, 2005.

[171] Mark van Atten. The development of intuitionistic logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2009 edition, 2009.

[172] Steffen van Bakel, Stéphane Lengrand, and Pierre Lescanne. The language $\mathscr{X}$: Circuits, computations and classical logic. In Coppo et al. [**47**], pages 81–96.

[173] Jean van Heijenoort. *From Frege to Gödel. A source book in mathematical logic, 1879–1931*. Harvard University Press, Cambridge, Mass., 1967.

[174] Jaap van Oosten. Realizability: A historical essay. *Mathematical Structures in Computer Science*, 12(3):239–263, 2002.

[175] Wim Veldman. An intuitionistic completeness theorem for intuitionistic predicate logic. *J. Symb. Log.*, 41(1):159–166, 1976.

[176] Wim Veldman. Brouwer's fan theorem as an axiom and as a contrast to Kleene's alternative. Technical report, Department of Mathematics, Radboud University Nijmegen, 2005. Report No. 0509.

[177] Wim Veldman. The principle of open induction on the unit interval [0,1] and some of its equivalents. Slides, May 2010.

[178] Igor Walukiewicz, editor. *Foundations of Software Science and Computation Structures, 7th International Conference, FOSSACS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings*, volume 2987 of *Lecture Notes in Computer Science*. Springer, 2004.

[179] Freek Wiedijk, editor. *The Seventeen Provers of the World, Foreword by Dana S. Scott*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.

[180] Wikipedia. Abstract rewriting system — Wikipedia, the free encyclopedia, 2009. [Online; accessed 20-June-2010].

[181] Wikipedia. Glivenko's theorem — Wikipedia, the free encyclopedia, 2009. [Online; accessed 1-July-2010].

[182] Wikipedia. Pairing function — Wikipedia, the free encyclopedia, 2010. [Online; accessed 11-June-2010].

[183] Wikipedia. POPLmark challenge — Wikipedia, the free encyclopedia, 2010. [Online; accessed 11-June-2010].

[184] A. J. Wilkie. On exponentiation - a solution to tarski's high school algebra problem. Technical report, 2001.

[185] Andrew K. Wright and Matthias Felleisen. A syntactic approach to type soundness. *Inform. and Comput.*, 115(1):38–94, 1994.

[186] Noam Zeilberger. Polarity and the logic of delimited continuations. In *Proceedings, 25th Annual IEEE Symposium on Logic in Computer Science (LICS '10), Edinburgh, UK, 11-14 July 2010*, page N/A. IEEE Computer Society Press, 2010.

APPENDIX A

Additional material for MQC$^+$

A.1. Call-by-name translation for MQC$^+$

The CPS translation and reduction rules of Chapter 4 were based on the call-by-value version of the shift-reset calculus. Using the monadic operations from Lemma 4.2.3, we can define an alternative, call-by-name, CPS translation,

$$
\begin{aligned}
\overline{(\cdot)} &: \Gamma \vdash^+ A \to \Gamma^T \vdash^m A^T \\
\overline{a} &= a \\
\overline{\lambda a.p} &= \eta(\lambda a.\overline{p}) \\
\overline{pq} &= (\alpha \mapsto \lambda k.\overline{p}(\lambda f.f(\eta\alpha)k))^*\overline{q} \\
\overline{(p,q)} &= \eta(\overline{p},\overline{q}) \\
\overline{\pi_1 p} &= (\alpha \mapsto \pi_1\alpha)^*\overline{p} \\
\overline{\iota_1 p} &= \eta(\iota_1\overline{p}) \\
\overline{\mathsf{case}\ p\ \mathsf{of}\ \left(a_1.q_1 \| a_2.q_2\right)} &= (\alpha \mapsto \mathsf{case}\ \alpha\ \mathsf{of}\ \left(a_1.\overline{q_1} \| a_2.\overline{q_2}\right))^*\overline{p} \\
\overline{\lambda x.p} &= \eta(\lambda x.\overline{p}) \\
\overline{pt} &= (\alpha \mapsto \alpha t)^*\overline{p} \\
\overline{(t,p)} &= \eta(t,\overline{p}) \\
\overline{\mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q} &= (\alpha \mapsto \mathsf{dest}\ \alpha\ \mathsf{as}\ (x.a)\ \mathsf{in}\ \overline{q})^*\overline{p} \\
\overline{\#p} &= \overline{p} \\
\overline{\mathscr{S}k.p} &= \lambda k'.\mu\left(\overline{p}\left\{\eta(\lambda a.\eta(k'a))\middle/ k\right\}\right),
\end{aligned}
$$

which can be used together with a double-negation translation similar to Kolmogorov's,

$$
\begin{aligned}
A^T &:= (A_T \Rightarrow T) \Rightarrow T \\
A_T &:= A && \text{if } A \text{ is a } \Sigma\text{-formula} \\
(A \square B)_T &:= A^T \square B^T && \text{for } \square = \vee, \wedge, \Rightarrow \\
(\square A)_T &:= \square A^T && \text{for } \square = \exists, \forall,
\end{aligned}
$$

to give a call-by-name version of Theorem 4.2.4. The only difference is that the target for the translation is not $\Gamma_T \vdash^m A^T$, but $\Gamma^T \vdash^m A^T$. We give the case for shift also as a proof tree:

$$\dfrac{\dfrac{\dfrac{\text{IH}}{\Gamma^T \vdash (A \Rightarrow T)^T \Rightarrow T^T}\Rightarrow_I}{\Gamma^T, A_T \Rightarrow T \vdash (A \Rightarrow T)^T \Rightarrow T^T} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\cdots}{\Gamma^T, A_T \Rightarrow T, A^T \vdash T}\Rightarrow_E}{\Gamma^T, A_T \Rightarrow T, A^T \vdash T_T}\eta}{\Gamma^T, A_T \Rightarrow T, A^T \vdash T^T}\Rightarrow_I}{\Gamma^T, A_T \Rightarrow T \vdash A^T \Rightarrow T^T}\eta}{\Gamma^T, A_T \Rightarrow T \vdash (A \Rightarrow T)^T}\Rightarrow_E}{\dfrac{\dfrac{\Gamma^T, A_T \Rightarrow T \vdash T^T}{\Gamma^T, A_T \Rightarrow T \vdash T}\mu}{\Gamma^T \vdash A^T}\Rightarrow_I}$$

Note, however, that in the translation A_T we now stop the recursive descent on the structure of A, once A has the form of a Σ-formula, *not* of an atomic formula.

This might appear as an arbitrary choice, however, it allows us to easily define the $\mu(\cdot)$ ("run") operator for the monads of form $\Gamma \vdash T^T$; a standard descent down to atomic formulae would have forced us to make as the target of the $(\cdot)^T$ translation, second order logic, because in that case we would have needed (predicative) polymorphism over the formula T in order to define "run". For the call-by-value translation, $S_T = S$ by definition. (S, T are Σ-formulae)

It would be challenging to carry out the entire development of Chapter 4 for call-by-name delimited control, since such calculi have only recently started to be studied, notably by Ghilezan, Herbelin, Kiselyov, and Saurin [**101, 112, 154**].

A.2. Explicit version of the two-level CPS transform

This section contains the explicit versions of the two-level CPS-translation and the associated lemmas, from Section 4.4.

$$\begin{aligned}
\overline{\Gamma \vdash^+ p : A} &: \forall C.\ (\Gamma_T \vdash C \to \Gamma_T \vdash T) \to (\Gamma_T \vdash A_T \to \Gamma_T \vdash C) \to \Gamma_T \vdash C \\
\overline{a} &= C \mapsto L \mapsto \kappa \mapsto \kappa \cdot a \\
\overline{\lambda a.p} &= C \mapsto L \mapsto \kappa \mapsto \kappa \cdot \left(\lambda a.\lambda k.\overline{p} \cdot T \cdot \mathrm{id} \cdot \left(\beta \mapsto k\beta\right)\right) \\
\overline{pq} &= C \mapsto L \mapsto \kappa \mapsto \overline{p} \cdot C \cdot L \cdot \left(\phi \mapsto \overline{q} \cdot C \cdot L \cdot \left(\alpha \mapsto \phi\alpha\,(\lambda b.\kappa \cdot b)\right)\right) \\
\overline{(p,q)} &= C \mapsto L \mapsto \kappa \mapsto \overline{p} \cdot C \cdot L \cdot \left(\alpha \mapsto \overline{q} \cdot C \cdot L \cdot \left(\beta \mapsto \kappa \cdot \left(\alpha, \beta\right)\right)\right) \\
\overline{\pi_1 p} &= C \mapsto L \mapsto \kappa \mapsto \overline{p} \cdot C \cdot L \cdot \left(\gamma \mapsto \kappa \cdot \left(\pi_1 \gamma\right)\right) \\
\overline{\iota_1 p} &= C \mapsto L \mapsto \kappa \mapsto \overline{p} \cdot C \cdot L \cdot (\alpha \mapsto \kappa \cdot (\iota_1 \alpha)) \\
\overline{\mathsf{case}\ p\ \mathsf{of}\ \left(a_1.q_1 \| a_2.q_2\right)} &= C \mapsto L \mapsto \kappa \mapsto \overline{p} \cdot C \cdot L \cdot \left(\gamma \mapsto \left(\mathsf{case}\ \gamma\ \mathsf{of}\ \left(a_1.\overline{q_1} \cdot C \cdot L \cdot \kappa \| a_2.\overline{q_2} \cdot C \cdot L \cdot \kappa\right)\right)\right) \\
\overline{\lambda x.p} &= C \mapsto L \mapsto \kappa \mapsto \kappa \cdot \left(\lambda x.\lambda k.\overline{p} \cdot T \cdot \mathrm{id} \cdot \left(\beta \mapsto kx\beta\right)\right) \\
\overline{pt} &= C \mapsto L \mapsto \kappa \mapsto \overline{p} \cdot C \cdot L \cdot \left(\phi \mapsto \phi t\,(\lambda b.\kappa \cdot b)\right) \\
\overline{(t,p)} &= C \mapsto L \mapsto \kappa \mapsto \overline{p} \cdot C \cdot L \cdot (\alpha \mapsto \kappa \cdot (t, \alpha)) \\
\overline{\mathsf{dest}\ p\ \mathsf{as}\ (x.a)\ \mathsf{in}\ q} &= C \mapsto L \mapsto \kappa \mapsto \overline{p} \cdot C \cdot L \cdot \left(\gamma \mapsto \left(\mathsf{dest}\ \gamma\ \mathsf{as}\ (x.a)\ \mathsf{in}\ \overline{q} \cdot C \cdot L \cdot \kappa\right)\right) \\
\overline{\#p} &= C \mapsto L \mapsto \kappa \mapsto \kappa \cdot \left(\overline{p} \cdot T \cdot (\alpha \mapsto L \cdot \mathrm{id} \cdot (\kappa \cdot \alpha))\right) \\
\overline{\mathscr{S} l.p} &= C \mapsto L \mapsto \kappa \mapsto \left(\overline{p} \cdot T \cdot \mathrm{id} \cdot (\alpha \mapsto L \cdot (\kappa \cdot \alpha))\right)\left\{\lambda a.\lambda k.k\,(L \cdot (\kappa \cdot a))\,/\,l\right\}
\end{aligned}$$

Lemma (Explicit version of 4.4.1)**.** *If* $\vdash^+ V : A$ *for* V *a value, then, for any* C, L *and* κ*, such that*

$$L : \vdash C \to \vdash T \quad \kappa : \vdash A_T \to \vdash C,$$

we have that

$$\overline{V} \cdot C \cdot L \cdot \kappa = \kappa \cdot \left(\overline{V} \cdot A_T \cdot (\alpha \mapsto L \cdot (\kappa \cdot \alpha)) \cdot \mathrm{id} \right).$$

Lemma (Explicit version of 4.4.2)**.** *If* $\Gamma \vdash^+ q : A$ *and* V *is a closed value, then for any* C, L, κ *and* a*, such that*

$$L : \vdash C \to \vdash T \quad \kappa : \vdash A_T \to \vdash C,$$

we have that

$$\overline{q\{V/a\}} \cdot C \cdot L \cdot \kappa = \left(\overline{q} \cdot C \cdot L \cdot \kappa \right) \left\{ \overline{V} \cdot A_T \cdot (\alpha \mapsto L \cdot (\kappa \cdot \alpha)) \cdot \mathrm{id} \Big/ a \right\}.$$

Lemma (Explicit version of 4.4.3)**.** *If* P *is a pure evaluation context s.t.* $\Gamma \vdash^+ P : A$*, then, for every* C, L *and* κ*, such that*

$$L : \vdash C \to \vdash T \quad \kappa : \vdash A_T \to \vdash C,$$

we have that

$$\overline{P[\mathscr{S}\, l.s]} \cdot C \cdot L \cdot \kappa = \left(\overline{s} \cdot A_T \cdot (\alpha \mapsto L \cdot (\kappa \cdot \alpha)) \cdot \mathrm{id} \right) \left\{ \lambda a.\lambda k.k(L \cdot (\overline{P[a]} \cdot C \cdot L \cdot \kappa)) \Big/ l \right\}.$$

Lemma (Explicit version of 4.4.4)**.** *If* V *is a value, then so is* $\overline{V} \cdot A_T \cdot \kappa \cdot \mathrm{id}$*, for any* $\kappa : \Gamma_T \vdash A_T \to \Gamma_T \vdash T$*.*

Lemma (Explicit version of 4.4.5)**.** *If* $\vdash^+_\diamond p : A$*,* p *is not a value, and* p *is not of form* $P[\mathscr{S}\, k.p']$*, then, for some* r*,* $p \to^1 r$ *and, for any* C, L *and* κ*, such that*

$$L : \vdash C \to \vdash T \quad \kappa : \vdash A_T \to \vdash C,$$

we have that

$$\overline{p} \cdot C \cdot L \cdot \kappa \to^\diamond \overline{r} \cdot C \cdot L \cdot \kappa.$$