Lecture 5
From Formulas to Programs, Part 2
Krivine's approach to classical logic

Davide Barbarossa

Contents

1 Second order logic in a nutshell

Fix a *second order signature*, i.e. a set $\mathcal{V}_1$ of *first order variables*, a set $\mathcal{V}_2$ of *second order variables*, each coming with its own arity, denoted X^k, and sets $\mathcal{S}_{k_1}, \dots, \mathcal{S}_{k_n}$ of *functions* $f : \mathbb{N}^{k_i} \to \mathbb{N}$, respectively[1].

Expressions For each $n \in \mathbb{N}$, the set $\mathcal{E}_n$ contains judgments $a_1, \dots, a_n \vdash e$ (where the context is a finite list), called *expressions in context*, and is defined by the following rules:

$$\frac{}{\vec{a} \vdash n}(n \in \mathbb{N}) \qquad \frac{}{\vec{a} \vdash a_i}(a_1, \dots, a_n \in \mathcal{V}_1) \qquad \frac{\vec{a} \vdash e_1 \quad \cdots \quad \vec{a} \vdash e_k}{\vec{a} \vdash f(e_1, \dots, e_k)}(f \in \mathcal{S}_k)$$

[1] Instead of actual numerical functions, one usually one considers in the signature symbols for relations (and calls "constants" the 0-ary relation symbols), that are to be interpreted. But, typically, one designs the language with a particular intended interpretation in mind for the relation symbols, so that one is actually only interested in the interpretations which choose the intended one for them. For instance, targeting, say, graph theory, one adds to the language a binary relation symbol E with the intended interpretation of "adjacency"; or targeting the theory of real numbers one adds a binary relation symbol $\leq$ with the intended interpretation of "less-than-or-equal-to". So, the only "real" interpretation of the signature is that of the free first and second order variables. Here we target the theory of arithmetic, so we design our language for that. In particular, our constants are the natural numbers (0-ary functions), and the functions in the $\mathcal{S}_i$'s represent the intended interpretation of the function/relation symbols that one would add in the approach described above. The choice of functions instead of relations is inessential, as they can be encoded within each other. For instance, if one has a language with a relation R – giving thus rise to formulas $R(\vec{e})$ – with the intended interpretation of R as a relation f_R, we encode this by introducing f_R as a characteristic function and then formula $f_R(\vec{e}) = 1$ encodes the intended interpretation of $R(\vec{e})$. Anyway, design choices are inessential: we just want a simple language for arithmetic.

Formulas For all $k, n_0, \dots, n_k \in \mathbb{N}$, the sets $\mathcal{F}_{n_0,\dots,n_k}$, respectively containing judgments $A\{a_1, \dots, a_{n_0} \mid X^{n_1}, \dots, X^{n_k}\}$ (where the context is a pair of finite lists) called *formulas in context*, are defined by the following rules:

$$\frac{\vec{a} \vdash e_1 \quad \cdots \quad \vec{a} \vdash e_{n_i}}{X_i^{n_i}(e_1, \dots, e_{n_i})\{\vec{a} \mid \vec{X}\}} \qquad \frac{A\{\vec{a} \mid \vec{X}\} \quad B\{\vec{a} \mid \vec{X}\}}{(A \to B)\{\vec{a} \mid \vec{X}\}} \qquad \frac{A\{\vec{a}, c \mid \vec{X}\}}{\forall c.A\{\vec{a} \mid \vec{X}\}} \qquad \frac{A\{\vec{a} \mid \vec{X}, Y^k\}}{\forall^k Y.A\{\vec{a} \mid \vec{X}\}}$$

which we take modulo α-equivalence on quantified first and second order variables.

For example, $\forall b.\forall^3 X.(X(a, f(b), c) \to \forall d.Y(g(d, d), f(a), b))$ is a formula in context $\{c, b, a \mid Y\}$.

Remark 1.1. *We can encode the formula $\bot := \forall^0 X.X$, formula constructors $\neg(\_) := (\_) \to \bot$, $\wedge, \vee, \exists, \leftrightarrow$ and a formula constructor $=$ from expressions, in the usual way, and for which we can derive the expected formula formation rules. For example, $e_1 = e_2$ is sugar for Leibniz $\forall^1 X.(X(e_1) \leftrightarrow X(e_2))$ and the formula formation rule*

$$\frac{\vec{a} \vdash e_1 \qquad \vec{a} \vdash e_2}{(e = e')\{\vec{a} \mid \vec{X}\}}$$

is derivable. Similarly, $A \vee B$ is sugar for $\forall^0 X.\ (A \to X) \to (B \to X) \to X$, and we have the associated formula formation derivable rule.

Therefore, we can encode the second order theories of interest, typically, Peano's PA2.

Example 1.2. *We call induction axiom the formula $\forall^1 X.\,(X(0) \wedge \forall a.(X(a) \to X(\mathrm{succ}(a))) \to \forall a.X(a))$ (in the context $\{|\}$), where $\mathrm{succ} \in \mathcal{S}_1$ is the successor function. Another example: for a second order variable F^2, one can easily write a formula in context $\mathrm{injfun}_F\{ \mid F\}$ that expresses the fact that (a future interpretation of) F is an injective function (instead of just a binary relation). Now given $\chi_A \in \mathcal{S}_1$ the characteristic function of a fixed set $A \subseteq \mathbb{N}$, we have the formula $\exists^2 F.\ \mathrm{injfun}_F \wedge \forall x.(\chi_A(x) = 1 \to \forall z.(F(x, z) \to Y(z)))$ in context $\{ \mid Y^1\}$. This expresses the fact that the cardinality of A is less than or equal to that of (a future interpretation of) Y.*

Remark 1.3. *Let Y^k be a second order variable, $\vec{X}$ a list of second order variables (may or may not including Y) and $b_1, \dots, b_k$ first order variables. Let $B\{Y^k := \langle C, b_1, \dots, b_k\rangle\}$ be the formula obtained from B by replacing each $Y(e_1, \dots, e_k)$ in B (if any) with $C\{b_1 := e_1, \dots, b_k := e_k\}$. Then the following formulas formation rule is admissible:*

$$\frac{B\{\vec{a} \mid \vec{X}\} \qquad C\{\vec{a}, \vec{c}, b_1, \dots, b_k \mid \vec{X}, \vec{Z}\}}{(B\{Y^k := \langle C, b_1, \dots, b_k\rangle\})\{\vec{a}, \vec{c} \mid \vec{X} - \{Y\}, \vec{Z}\}}$$

Derivations A derivation is a tree with nodes judgments of shape $\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{M} : B$, where $\underline{\mathtt{x}} : \underline{A}$ is a finite *set* of declared program variables, $\vec{a}, \vec{X}$ are finite lists of first and second order formula variables such that $A_i\{\vec{a} \mid \vec{X}\}$ and $B\{\vec{a} \mid \vec{X}\}$, and $\mathtt{M}$ is a term in the following grammar (we will later give an operational semantics):

$$\mathtt{M} ::= \mathtt{x} \mid \lambda \mathtt{y}.\,\mathtt{M} \mid \mathtt{M}\,\mathtt{M} \mid \mathtt{callcc}.$$

Derivations are defined by the rules in Figure 1. The meaning of `callcc`, and its link with classical logic was introduced by Griffin (regardless of realisability) in [Griffin, 1990]. We will quickly discuss it when an operational semantics for the language is provided.

The $\mathtt{M}$ in $\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{M} : B$ is said to be the *proof-term* for such derivation.

An informal proof of a theorem in Peano's arithmetic is then formally encoded as a derivation of a $\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{M} : B$, where the $A_i \in \mathrm{PA2}$.

$$\frac{(A_i\{\vec{a} \mid \vec{X}\})_{i=1}^n \qquad B\{\vec{a} \mid \vec{X}\}}{\underline{\mathtt{x}} : \underline{A}, \mathtt{y} : B \vdash_{\vec{a}|\vec{X}} \mathtt{y} : B}\,\mathrm{proj}_{\mathtt{y}}^{\mathtt{x},\mathtt{y}} \qquad \frac{(A_i\{\vec{a} \mid \vec{X}\})_{i=1}^n \qquad B\{\vec{a} \mid \vec{X}\} \qquad C\{\vec{a} \mid \vec{X}\}}{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{callcc} : ((B \to C) \to B) \to B}\,\mathrm{pl}$$

$$\frac{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{M} : B \to C \qquad \underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{N} : B}{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{MN} : C}\,@ \qquad \frac{\underline{\mathtt{x}} : \underline{A}, \mathtt{y} : B \vdash_{\vec{a}|\vec{X}} \mathtt{M} : C}{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \lambda \mathtt{y}.\mathtt{M} : B \to C}\,\lambda$$

$$\frac{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a},c|\vec{X}} \mathtt{M} : B \qquad (A_i\{\vec{a} \mid \vec{X}\})_{i=1}^n}{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{M} : \forall c.B}\,\forall_i^1 \qquad \frac{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X},Y^k} \mathtt{M} : B \qquad (A_i\{\vec{a} \mid \vec{X}\})_{i=1}^n}{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{M} : \forall^k Y.B}\,\forall_i^2$$

$$\frac{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{M} : \forall c.B \qquad \vec{a}, \vec{b} \vdash e}{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a},\vec{b}|\vec{X}} \mathtt{M} : B\{c := e\}}\,\forall_e^2 \qquad \frac{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{M} : \forall^k Y.B \qquad C\{\vec{a}, \vec{c}, b_1, \ldots, b_k \mid \vec{X}, \vec{Z}\}}{\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a},\vec{c}|\vec{X},\vec{Z}} \mathtt{M} : B\{Y^k := \langle C, b_1, \ldots, b_k\rangle\}}\,\forall_e^2$$

Figure 1: Rules of second-order classical logic.

$$\dfrac{\dfrac{\dfrac{}{\mathtt{x} : \neg\neg A \vdash_{\vec{a}|\vec{Z}} \mathtt{callcc} : (\neg A \to A) \to A}\,\mathrm{pl} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathtt{x} : \neg\neg A, \mathtt{y} : \neg A \vdash_{\vec{a}|\vec{Z}} \mathtt{x} : \neg\neg A}\,\mathrm{proj} \qquad \dfrac{}{\mathtt{x} : \neg\neg A, \mathtt{y} : \neg A \vdash_{\vec{a}|\vec{Z}} \mathtt{y} : \neg A}\,\mathrm{proj}}{\mathtt{x} : \neg\neg A, \mathtt{y} : \neg A \vdash_{\vec{a}|\vec{Z}} \mathtt{xy} : \bot}\,@}{\mathtt{x} : \neg\neg A, \mathtt{y} : \neg A \vdash_{\vec{a}|\vec{Z}} \mathtt{xy} : A}\,\forall_e^2}{\mathtt{x} : \neg\neg A \vdash_{\vec{a}|\vec{Z}} \lambda \mathtt{y}.\mathtt{xy} : \neg A \to A}\,\lambda}{\mathtt{x} : \neg\neg A \vdash_{\vec{a}|\vec{Z}} \mathtt{callcc}(\lambda \mathtt{y}.xy) : A}\,@}{\vdash_{\vec{a}|\vec{Z}} \lambda \mathtt{x}.\,\mathtt{callcc}(\lambda \mathtt{y}.\mathtt{xy}) : \neg\neg A \to A}\,\lambda$$

Figure 2: Derivation and proof-term for the double negation elimination for $A\{\vec{a} \mid \vec{Z}\}$, using that clearly $\bot\{\vec{a} \mid \vec{Z}\}$, $\neg\neg A\{\vec{a} \mid \vec{Z}\}$ and $\neg A\{\vec{a} \mid \vec{Z}\}$.

Example 1.4. *Suppose that $A\{\vec{a} \mid \vec{Z}\}$. Then in Figure 2 there is a derivation of the double negation elimination, showing that we are indeed dealing with classical logic.*

Ex. 1 — Remember the 2nd order encoding of $\vee$ given in Remark 1.1. Show that indeed the usual introduction rules for $\vee$ are admissible from the rules of Figure 1.

Answer (Ex. 1) — See Section 3. □

Ex. 2 — Consider the following proof of the excluded middle for A from Peirce's law:

> *Remark that an instance of Peirce's law for a formula C is $(notC \Rightarrow C) \Rightarrow C$ (this instance of Pierce's law is called* consequentia mirabilis*). So, taking $C = (A$ or $notA)$, in order to establish the excluded middle for A it suffices to show $not(A$ or $notA) \Rightarrow (A$ or $notA)$. So we have to prove $(A$ or $notA)$ under the hypothesis $not(A$ or $notA)$, call this hypothesis* $\mathtt{y}$*. We claim that we can prove $notA$, i.e. let us prove a contradiction under hypothesis A, call this hypothesis* $\mathtt{x}$*. But from* $\mathtt{x}$ *we obtain $(A$ or $notA)$. So from* $\mathtt{y}$ *we obtain the desiered contradiction.*

Formalise the proof above as a derivation from the rules system of Figure 1.

Answer (Ex. 2) — See Section 3. □

2 Classical Realisability: From Tarski to Krivine

Define the function $[\![\_]\!] : \mathcal{E}_h \times \mathbb{N}^h \to \mathbb{N}$ by

$$[\![\vec{a} := \vec{n} \vdash m]\!] := m \quad [\![\vec{a} := \vec{n} \vdash a_i]\!] := n_i \quad [\![\vec{a} := \vec{n} \vdash f(e_1, \dots, e_k)]\!] := f([\![\vec{a} := \vec{n} \vdash e_1]\!], \dots, [\![\vec{a} := \vec{n} \vdash e_k]\!])$$

where we write $\vec{a} := \vec{n} \vdash e$ for the tuple $(\vec{a} \vdash e, \vec{n}) \in \mathcal{E}_h \times \mathbb{N}^h$, and we call it a *parametric expression*, $\vec{n}$ being the parameters.

2.1 Tarski semantics

Tarski's semantics sees a formula as being exactly either true or false, once an interpretation for its free variables is fixed. We define below the usual second order (full) Tarski semantics by putting it in the conceptual shape that allows us to see what place the assumption above takes.

In Tarski, any formula exactly has either one counterwitness or it has none. Let us call † the only possible counterwitness to formulas, and □ the only possible witness to formulas. These represent the fact that a proof exists (□), or that a refutation exists (†), and they are called the *truth values*. We obviously want some notion of duality between them, defining how they interact; we do that by an orthogonality relation $\perp\!\!\!\perp$ between the set $\{\square\}$ of witnesses and the one $\{\dagger\}$ of counters. With this, we can define as usual the witnesses orthogonal to a set of counters C, as in Figure 3a.

In Tarski we take the trivial interaction: there is a counter iff there is no witness, i.e. the duality $\perp\!\!\!\perp \subseteq \{\square\} \times \{\dagger\}$ is the trivial one $\perp\!\!\!\perp := \emptyset$. Figure 3b shows a summary.

Thus, for $C \in \wp(\{\dagger\}) = \{\emptyset, \{\dagger\}\}$, we have

$$\{\emptyset, \{\square\}\} = \wp(\{\square\}) \ni C^{\perp\!\!\!\perp} = \begin{cases} \emptyset & \text{if } C \neq \emptyset \text{ (i.e. } C = \{\dagger\}\text{)} \\ \{\square\} & \text{if } C = \emptyset \end{cases}$$

We thus see that the set of witnesses and of counters indeed model truth values, and $(\_)^{\perp\!\!\!\perp}$ is the negation: "True" is encoded as both the element $\{\square\}$ of $\wp(\{\square\})$ and $\emptyset$ of $\wp(\{\dagger\})$, and "False" is encoded as both the element $\emptyset$ of $\wp(\{\square\})$ and $\{\dagger\}$ of $\wp(\{\dagger\})$.

Let us now define the interpretation of formulas. For the change of perspective that we will do with Krivine's realisability, it is better to start by focusing on the idea of a formula being disproved, so we first define the interpretation in terms of counterwitnesses.

A second order variable X^n is interpreted as a n-predicate, i.e. a set $P \subseteq \mathbb{N}^n$; now, this can be equivalently thought of as a function $P : \mathbb{N}^n \to \wp(\{\dagger\})$, associating $n \in \mathbb{N}^n$ with the set $P(\vec{n})$ of all the counterwitnesss to the formula $X(\vec{n})$. Since witness and counters are, in this case, boolean truth values, $P(\vec{n})$ is the set $\emptyset \in \wp(\{\dagger\})$ when there is no counterwitness to the formula $X(\vec{n})$, i.e. when it is true (i.e. when we have the only witness □), and the set $\{\dagger\}$ when there is a counterwitness (i.e. when the formula is false), the only possible one being †.

Analogously to the case of Tarski semantics for first order logic, the semantics of formulas can be then presented as the data of functions

$$\mathcal{C} : \mathcal{F}_{h_0,\dots,h_k} \times \mathbb{N}^{h_0} \times \wp(\{\dagger\})^{\mathbb{N}^{h_1}} \times \cdots \times \wp(\{\dagger\})^{\mathbb{N}^{h_k}} \to \wp(\{\dagger\})$$

$$\mathcal{W} : \mathcal{F}_{h_0,\dots,h_k} \times \mathbb{N}^{h_0} \times \wp(\{\dagger\})^{\mathbb{N}^{h_1}} \times \cdots \times \wp(\{\dagger\})^{\mathbb{N}^{h_k}} \to \wp(\{\square\})$$

defined in Figure 3c, where

$$* : \wp(\{\square\}) \times \wp(\{\dagger\}) \to \wp(\{\dagger\})$$

$$\boxed{C^{\perp\!\!\!\perp} := \{t \in \mathbb{W} \mid t \perp\!\!\!\perp \pi \text{ for all } \pi \in C\}}$$

(a) Orthogonal $C^{\perp\!\!\!\perp} \subseteq \mathbb{W}$ of a $C \subseteq \mathbb{C}$.

	$\mathbb{W}$	$\mathbb{C}$	$\perp\!\!\!\perp \subseteq \mathbb{W} \times \mathbb{C}$	$* : \mathcal{P}(\mathbb{W}) \times \mathcal{P}(\mathbb{C}) \to \mathcal{P}(\mathbb{C})$
Tarski	$\{\Box\}$	$\{\dagger\}$	$\emptyset$	$\Rightarrow$
Krivine	Λ	Λ^*	*pole*	*cons*

(b) Witnesses, Counterwitnesses, Orthogonality and "merging", for Tarski and Krivine's semantics.

$$\begin{array}{rcl}
\mathcal{C}(X_i^{h_i}(e_1,\dots,e_{h_i})\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}) & = & P_i([\![\vec{a} := \vec{n} \vdash e_1]\!], \dots, [\![\vec{a} := \vec{n} \vdash e_{h_i}]\!]) \\
\mathcal{C}((A \to B)\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}) & = & \mathcal{W}(A\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}) * \mathcal{C}(B\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}) \\
\mathcal{C}(\forall c.A\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}) & = & \bigcup_{m \in \mathbb{N}} \mathcal{C}(A\{\vec{a} := \vec{n}, c := m \mid \vec{X} := \vec{P}\}) \\
\mathcal{C}(\forall^m Y.A\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}) & = & \bigcup_{Q : \mathbb{N}^m \to \wp(\mathbb{C})} \mathcal{C}(A\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}, Y^m := Q\}) \\
\mathcal{W}(_) & := & \mathcal{C}(_)^{\perp\!\!\!\perp}
\end{array}$$

(c) Semantics of formulas (both Tarski and Realisability). We write $A\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}$ for the tuple $(A\{\vec{a} \mid \vec{X}\}, n_1, \dots, n_{h_0}, P_1, \dots, P_k) \in \mathcal{F}_{h_0,\dots,h_k} \times \mathbb{N}^{h_0} \times \wp(\mathbb{C})^{\mathbb{N}^{h_1}} \times \dots \times \wp(\mathbb{C})^{\mathbb{N}^{h_k}}$, and we call it a *parametric formula*, $\vec{n}$ and $\vec{P}$ being the *parameters*.

Figure 3: Semantics of formulas, in a unified version for both Tarski and Krivine.

is the *entailment* wrt the natural encoding of "True" and False mentioned above, i.e.

$$\begin{array}{ccccc}
\emptyset & * & \emptyset & = & \emptyset \\
\emptyset & * & \{\dagger\} & = & \emptyset \\
\{\Box\} & * & \emptyset & = & \emptyset \\
\{\Box\} & * & \{\dagger\} & = & \{\dagger\}
\end{array}$$

Remark 2.1. *It is important to remark that the above is* literally *the usual definition of Tarski truth-semantics, we are just phrasing it with different words: given a Tarski structure $\mathfrak{M}$ for A, i.e. parameters $(\vec{n}, \vec{P}) \in \mathbb{N}^{h_0} \times \wp(\{\dagger\})^{\mathbb{N}^{h_1}} \times \dots \times \wp(\{\dagger\})^{\mathbb{N}^{h_k}}$ for A (i.e. an interpretation of its free first and second order variables), $\mathcal{W}(A)$ is the usual Tarski truth value of A, as we have:*

$$\mathfrak{M} \vDash A \text{ iff } \mathcal{W}(A)_{\vec{n}|\vec{P}} = \{\Box\} \text{ iff } \mathcal{C}(A)_{\vec{n}|\vec{P}} = \emptyset \text{ iff } \mathcal{C}(\neg A)_{\vec{n}|\vec{P}} = \{\dagger\} \text{ iff } \mathfrak{M} \nvDash \neg A.$$

One can prove the following adequacy result, which in the case of Tarski is called soundness:

Theorem 2.2 (Soundness)**.** *Let $\mathtt{x}_1 : A_1, \dots, \mathtt{x}_m : A_m \vdash_{\vec{a}|\vec{X}} \mathtt{M} : B$ be provable and let $(\vec{n}, \vec{P})$ be parameters for $A_1, \dots, A_m, B$.*

If $\Box \in \mathcal{W}(A_i\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\})$ for all $i = 1, \dots, m$, then $\Box \in \mathcal{W}(B\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\})$.

Therefore, a proof of $\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} B$ trivially defines, for each parameter $(\vec{n}, \vec{P})$, a (trivially computable) function $\prod_{\mathtt{x} \in \underline{\mathtt{x}}} \mathcal{W}(A_{\mathtt{x}}\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}) \to \mathcal{W}(B\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\})$, namely: if

the domain is non-empty – thus, it is $\{(\square, \cdots, \square)\}$ – then it returns $\square$; otherwise, the domain is empty and the function is the empty function. This is simply saying that a proof transports evidence from the hypotheses to the conclusion, i.e., as standard, it preserves truth.

2.2 Realisability semantics

Just like Tarski semantics, also realisability is a semantics of formulas (as opposed to a semantics of proofs/programs, that we mentioned in the previous lectures and that is more typical in proof and/or program theory). Just like Tarski, it is obtained via the exact same construction as above. But it takes a computational viewpoint: while Tarski defines the semantics of formulas in terms of boolean truth valued witnesses and counterwitnesses, realisability defines a semantics of formulas in terms of computationally relevant witnesses and counters. In particular, contrarily to Tarski, a formula typically admits both witnesses and counters. Of course, also realisability provides a semantics that is sound for proofs; but since it is computation oriented, it is able to distinguish between different proofs (while in Tarski a proof has no structure, as all proofs behave the same – see Theorem 2.2 and the lines just after it): thanks to the Curry-Howard correspondence we know that proofs are (at least) non-trivial computational objects and realisability is compatible with such understanding of proofs – see Theorem 2.9 and the lines just after it.

More specifically, we move from Tarski semantics' understanding of proofs:

> *a proof is the trivial function* $\{(\square, \ldots, , \square)\} \to \{\square\}$ *sending the truth of the hypotheses to the truth of the conclusion*

to the BHK understanding:

> *a proof is a computable function sending witnesses (there may be many) of the hypotheses to witnesses (there may be many) of the conclusion.*

In fact, realisability (in all its variants, Kleene, Kreisler's modified, Krivine) can be precisely seen as a formalisation of BHK. Some other versions of realisabilities, like Gödel's Dialectica, can be seen as refinements of it.

Actually, as mentioned above, in realisability one even takes the finer viewpoint given by the Curry-Howard/proof-program theoretic understanding of proofs:

> *a proof defines of a program, in some fixed programming language, computing the BHK function above.*

That is precisely the meaning of the Adequacy Theorem 2.9 below. Krivine's and Kreisler's realisability[2] fully take a Curry-Howard approach to logic, so a proof will *directly* define a program. In other frameworks – such as the original Kleene's realisability – it would only define an encoding of a program, typically a number n encoding the n-th (program for a) Turing machine.

The computational model Let us introduce the Turing-complete, untyped programming language that we work with. Since we want proofs-terms to act on witnesses, a natural choice is to take a "monistic" approach, in the sense that both proofs and witnesses will be the same kind of objects. Since proof-terms are already $\lambda + \texttt{callcc}$-terms, we take witnesses to be the same, and the action of the former on the latter to be the usual application of λ-terms. In order to get a meaningful OPerational Semantics (OPS), we actually need to extend the language:

[2] See [Oliva and Streicher, 2008] for the relationship between the intuitionistic and classical realisability: a slight variant of Krivine's realisability (obtained via a CPS-formulation) corresponds to first performing a (slight variant of the) Krivine-Streicher-Reus negative translation of formulas, and then applying intuitionistic modified realisability to them.

Definition 2.3. *We define a langueage with following syntax:*

$$\begin{array}{lllll} \textit{(terms)} & \Lambda_{\mathtt{callcc}} \ni & \mathtt{M} ::= & \mathtt{x} \mid \lambda\mathtt{x}.\,\mathtt{M} \mid \mathtt{M}\,\mathtt{M} \mid \mathtt{callcc} \mid \mathtt{k}_\pi \\ \textit{(stacks)} & \Lambda^*_{\mathtt{callcc}} \ni & \pi ::= & [] \mid \mathtt{M} :: \pi \end{array}$$

The terms $\mathtt{k}_\pi$ *are called* continuations*. In* $\lambda\mathtt{x}.\,\mathtt{M}$*, we let* $\lambda\mathtt{x}$ *only bind the* $\mathtt{x}$ *of* $\mathtt{M}$ *which are* not *in the stack of a continuation. E.g.,* $\lambda\mathtt{x}.\,\mathtt{x}\,\mathtt{k}_{\mathtt{x}::[]} = \lambda\mathtt{y}.\,\mathtt{y}\,\mathtt{k}_{\mathtt{x}::[]}$*.*

We inductively define when a term is closed*: this is the usual definition, plus stating that* $\mathtt{callcc}$ *is closed and* $\mathtt{k}_\pi$ *is closed whenever all the terms in* π *are. The* closed *terms of* $\Lambda_{\mathtt{callcc}}$ *are called* λ_c*-terms, and the set of those is denoted* Λ_c*. The set* Λ^*_c *is the set of lists of* λ_c*-terms.*

The λ_c*-terms which are continuation-free are said to be* proof-like *terms.Observe that all closed proof-terms are proof-like and, more generally, all proof-terms are continuation-free. Moreover, a derivation is intuitionistic (i.e. not using Pierce's law) iff its proof-term is* $\mathtt{callcc}$*-free.*

We denote $\Lambda_c \times \Lambda^*_c$ *by* Proc*. Its elements are called* processes *and denoted* $\mathtt{M} \star \pi$*.*

Definition 2.4. *The OPS of* $\Lambda_{\mathtt{callcc}}$ *and* Λ_c *is defined in terms of the following simplified KAM, given by the rewriting relation* $\succ \subseteq (\Lambda_{\mathtt{callcc}} \times \Lambda^*_{\mathtt{callcc}}) \times (\Lambda_{\mathtt{callcc}} \times \Lambda^*_{\mathtt{callcc}})$ *which is the reflexive and transitive closure of the following one-step reduction* $\to$*:*

$$\begin{array}{lrcrcrcr} \textbf{\textit{(push)}} & \mathtt{M}\,\mathtt{N} & \star & \pi & \to & \mathtt{M} & \star & \mathtt{N} :: \pi \\ \textbf{\textit{(pop)}} & \lambda\mathtt{x}.\,\mathtt{M} & \star & \mathtt{N} :: \pi & \to & \mathtt{M}\{\mathtt{x} := \mathtt{N}\} & \star & \pi \\ \textbf{\textit{(save)}} & \mathtt{callcc} & \star & \mathtt{M} :: \pi & \to & \mathtt{M} & \star & \mathtt{k}_\pi :: \pi \\ \textbf{\textit{(restore)}} & \mathtt{k}_\pi & \star & \mathtt{M} :: \rho & \to & \mathtt{M} & \star & \pi \end{array}$$

In particular, remark that $\succ$ *restricts to closed terms, i.e. if* $(\mathtt{M}, \pi) \in \mathrm{Proc} = \Lambda_c \times \Lambda^*_c$ *and* $\mathtt{M} \star \pi \succ \mathtt{M}' \star \pi'$*, then* $(\mathtt{M}', \pi') \in \Lambda_c \times \Lambda^*_c$*. Therefore, we have a reduction* $\succ \subseteq \mathrm{Proc} \times \mathrm{Proc}$*.*

Remark 2.5. *One could, of course, add a new program primitive symbol for any axiom that we like; but in order for such an assignment to be interesting it should be non-ad-hoc, namely it should at least have a computationally meaningful semantics. This is precisely what one does with* $\mathtt{callcc}$ *for Peirce's law. Finding interesting programming primitives that allow us to realise interesting axioms is, in a sense, the whole point of realisability.*

Example 2.6. *A typical run in this KAM is shown by that of the proof term for the double negation elimination found in Figure 2:*

$$\begin{array}{lrcl} & \lambda\mathtt{x}.\,\mathtt{callcc}(\lambda\mathtt{y}.\,\mathtt{x}\,\mathtt{y}) & \star & \mathtt{M} :: \pi \\ \succ & \mathtt{callcc}(\lambda\mathtt{y}.\,\mathtt{M}\,\mathtt{y}) & \star & \pi \\ \succ & \lambda\mathtt{y}.\,\mathtt{M}\,\mathtt{y} & \star & \mathtt{k}_\pi :: \pi \\ \succ & \mathtt{M}\,\mathtt{k}_\pi & \star & \pi \\ \succ & \mathtt{M} & \star & \mathtt{k}_\pi :: \pi \end{array}$$

The realisability relation and its adequacy The construction of the realisability semantics is exactly as the Tarski's, but for the fact that we make the set of possible witnesses and counters non-trivial. Let us call them $\mathbb{W}$ and $\mathbb{C}$, respectively. Now all the construction is the same by replacing $\{\Box\}$ by $\mathbb{W}$ and $\{\dagger\}$ by Π, and by choosing the opportune operation $*$ and duality $\perp\!\!\!\perp$.

We take

$$\mathbb{W} := \Lambda_c.$$

Remembering their OPS and that we need a function $* : \wp(\mathbb{W}) \times \wp(\mathbb{C}) \to \wp(\mathbb{C})$ it is natural to take

$$\mathbb{C} := \mathbb{W}^* = \Lambda^*_c$$

and $*$ the extension of the cons operation on sets:

$$W * C := \{\mathtt{M} :: \pi \mid \mathtt{M} \in W \text{ and } \pi \in C\}.$$

Finally, we need a duality relation $\perp\!\!\!\perp \subseteq \mathbb{W} \times \mathbb{C}$. We define it via the OPS of λ_c-terms: fix a $\succ$-downward closed $\perp\!\!\!\perp \subseteq$ Proc, called a *pole* (which is thus a parameter of all the construction; also, remember that $\succ$-downward closed means that, for all $p, q \in$ Proc, we have: $p \succ q \in \perp\!\!\!\perp \Rightarrow p \in \perp\!\!\!\perp$). We set

$$\mathtt{M} \perp\!\!\!\perp \pi \quad \textit{whenever} \quad \mathtt{M} \star \pi \in \perp\!\!\!\perp .$$

The orthogonal $C^{\perp\!\!\!\perp} \subseteq \mathbb{W}$ of $C \subseteq \mathbb{C}$ is defined as before in Figure 3a.

Now the definition of

$$\mathcal{C} : \mathcal{F}_{h_0,\dots,h_k} \times \mathbb{N}^{h_0} \times \wp(\Pi_c)^{\mathbb{N}^{h_1}} \times \cdots \times \wp(\Pi_c)^{\mathbb{N}^{h_k}} \to \wp(\Pi_c)$$

and

$$\mathcal{W} : \mathcal{F}_{h_0,\dots,h_k} \times \mathbb{N}^{h_0} \times \wp(\Pi_c)^{\mathbb{N}^{h_1}} \times \cdots \times \wp(\Pi_c)^{\mathbb{N}^{h_k}} \to \wp(\Lambda_c)$$

is given in Figure 3c, exactly as before.

Example 2.7. *Fix a pole $\perp\!\!\!\perp$. Let us show that, for all formulas A, B and $\pi \in \mathcal{C}(A)$, we have $\mathtt{k}_\pi \in \mathcal{W}(A \to B)$. In particular, $\mathtt{k}_\pi \in \mathcal{W}(\neg A)$.*

*We have to show that for $\mathtt{M} \in \mathcal{W}(A)$, $\rho \in \mathcal{C}(B)$, we have $\mathtt{k}_\pi \star \mathtt{M} :: \rho \in \perp\!\!\!\perp$. This is immediate by **(restore)**.*

Ex. 3 — Show that the proof-term of Example 1.4 witnesses the conclusion of its proof. I.e., for all A and pole $\perp\!\!\!\perp$, we have: $\lambda \mathtt{x}.\, \mathtt{callcc}(\lambda \mathtt{y}.\, \mathtt{x}\, \mathtt{y}) \in \mathcal{W}(\neg\neg A \to A)$.

Answer (Ex. 3) — See Section 3. □

Remark 2.8. *Remark that, setting $U \to V := \{\mathtt{M} \in \Lambda_c \mid \mathtt{M}\mathtt{N} \in V \text{ for all } \mathtt{N} \in U\}$, we have*

$$\mathcal{W}(A \to B) = \mathcal{W}(A) \to \mathcal{W}(B).$$

Realisability witnesses thus enjoy the BHK understanding of an implication, in the sense that a witness for $A \to B$ is operationally indistinguishable from a program of "type" $\mathcal{W}(A) \to \mathcal{W}(B)$.

The situation of the previous exercise 3 is not a coincidence, as the following adequacy Theorem shows. It is analogous to the one in Tarski's case, but now it makes formal the BHK understanding of proofs. See e.g. [Krivine, 2003] for its proof.

Theorem 2.9 (Adequacy)**.** *Let $\mathtt{x}_1 : A_1, \dots, \mathtt{x}_m : A_m \vdash_{\vec{a}|\vec{X}} \mathtt{M} : B$ be provable, let $(\vec{n}, \vec{P})$ be parameters for $A_1, \dots, A_m, B$ and let $\mathtt{N}_1, \dots, \mathtt{N}_m$ be λ_c-terms. Let $\perp\!\!\!\perp$ be a pole. We have:*

If

$$\mathtt{N}_1 \in \mathcal{W}(A_1\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}), \dots, \mathtt{N}_m \in \mathcal{W}(A_m\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\})$$

then

$$\mathtt{M}\{\vec{\mathtt{x}} := \vec{\mathtt{N}}\} \in \mathcal{W}(B\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}).$$

Therefore, $\underline{\mathtt{x}} : \underline{A} \vdash_{\vec{a}|\vec{X}} \mathtt{M} : B$ defines a program $\lambda\vec{\mathtt{x}}.\mathtt{M}$ in λ_c-calculus which, for each parameter $(\vec{n}, \vec{P})$ and each pole $\perp\!\!\!\perp$, computes the function

$$\{\mathtt{N}_\mathtt{x}\}_{\mathtt{x}\in\underline{\mathtt{x}}} \in \prod_{\mathtt{x}\in\underline{\mathtt{x}}} \mathcal{W}(A_\mathtt{x}\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}) \quad \mapsto \quad \mathtt{M}\{\underline{\mathtt{x}} := \underline{\mathtt{N}}\} \in \quad \mathcal{W}(B\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\})$$

Observe how this precisely formalises the BHK "definition": a proof transports evidences from the hypotheses to the conclusion.

In particular, for the case of a derivation $\vdash_{\vec{a}|\vec{X}} \mathtt{M} : B$ under no hypotheses, the above means that for each pole $\bot\!\!\!\bot$ we have:

$$\mathtt{M} \in \mathcal{W}(B\{\vec{a} := \vec{n} \mid \vec{X} := \vec{P}\}).$$

Remember that Example 2.7 shows that continuations allow us to "cheat" and witness the negation of virtually all formulas. Whence the "proof-like clause" in the following:

Definition 2.10. *A witness* $\mathtt{M}$ *of* A *which is a proof-like term is called a* $\bot\!\!\!\bot$-realiser *of* A*, written* $\mathtt{M} \Vdash_{\bot\!\!\!\bot} A$*. A proof-like term* $\mathtt{M}$ *which* $\bot\!\!\!\bot$*-realises* A *for all pole* $\bot\!\!\!\bot$ *is said to be a realiser of* A *(sometimes a "universal" realiser), written* $\mathtt{M} \Vdash A$*.*

Realisers of A capture then the *computational content* of A. Remark that, from the Adequacy result above, we have in particular that $\vdash_{\vec{a}|\vec{X}} \mathtt{M} : B \Rightarrow \mathtt{M} \Vdash B$, as expected. For example, $\lambda\mathtt{x}.\,\mathtt{callcc}(\lambda\mathtt{y}.\,\mathtt{x}\,\mathtt{y}) \Vdash \neg\neg A \to A$.

Ex. 4 — Consider the derivation from Exercise 2, where for simplicity we take $A\{\,|\,\}$ (i.e. no first or second order free variables). Without using the Adequacy Theorem but just following the definitions, show that its proof-term realises indeed $A \vee \neg A$.

Answer (Ex. 4) — See Section 3. □

* **Ex. 5** — Set $\mathtt{M} \approx_n \mathtt{N}$ iff for all $\mathtt{N}_1, \ldots, \mathtt{N}_n \in \Lambda_c$ and $\pi \in \Pi_c$ we have $\mathtt{M} \star \mathtt{N}_1 :: \cdots :: \mathtt{N}_n :: \pi \succ p$ whenever $\mathtt{N} \star \mathtt{N}_1 :: \cdots :: \mathtt{N}_n :: \pi \to p$.
Prove that, for a proof-like term $\mathtt{M}$, we have: $\mathtt{M} \Vdash \forall X^0.\, X \to X$ iff $\mathtt{M} \approx_1 \mathtt{I}$.

Answer (Ex. 5) — See Section 3. □

Beyond proofs, toward axioms In these notes, we have defined the realisability semantics by formalising and refining the BHK understanding of proofs, we compared it with Tarski's one, and we stated the Adequacy Theorem, which says that realisability is a sound semantics (wrt provability) of formulas in terms of programs. We have roughly followed the first 5 sections of [Krivine, 2003].

But actually, the main point of realisability is to start from here in order to go beyond proofs, and to realise (i.e. find computational witnesses for) formulas that do *not* have a proof but that we may want to take as axioms! In fact, this is what we with for Peirce's law, yielding realisability for classical logic through `callcc`. In the case of Krivine's realisability, this has been brought to spectacular levels by realising the axioms of ZFC, for instance.

So let us conclude by simply summarising the two main aspects of Krivine's realisability:

- A proof/program theoretic pov: it is a technique to realise axioms, i.e. to witness/justify them from a computational point of view. From this perspective, it belongs to the family of functional interpretations, and in the particular framework of Krivine's realisability, we can realise highly non-constructive axioms. For instance, one can realise CH and, recently, Krivine realised AC. The challenge here is to add interesting and natural programming instructions to the computational model, such that with them one can realise non-constructive non-trivial axioms via interesting programs. The ideal goal would be, therefore, to obtain a concrete realiser (so, a λ-term plus other instructions), with an interesting computational behaviour that we are able to precisely describe, or maybe even

to characterize the computational behaviour of *any* realiser of a given axiom[3]. However, in many cases, one is only able to show by model theoretic ways (see below) that a realiser exists[4].

- A model theoretic pov: Under a mild hypothesis on the pole, if we fix the interpretation of all first and second order variables, and then we collect all formulas whose parametrisation (with the parameters from the fixed interpretation) admits a realiser (i.e. a proof-like term which witnesses the formula regardless of the choice of the pole), one can see that we get a (deductively closed) non-contradictory theory. If we are first order, this means by completeness that the theory has a Tarski-model. If we are in second-order, we can take Henkin-like models. They are called *realisability models* (and remark that the adequacy Theorem entails that realisability models are sound wrt provability). An important part of the active research in classical realisability is devoted to the study of such models, especially in the case of ZF, and particularly in relation with the Axiom of Choice [Krivine, 2021], or weaker variants of it [Fontanella and Geoffroy, 2020] (as well as other kind of set theoretical axioms).

In both cases, Krivine's realisability appears to be very much related to set theory's forcing[5]. On the one hand, in some cases the techniques in use are a syntactic adaptation of Forcing (see [Miquel, 2011, Krivine, 2011], based on the draft [Krivine, 2008][6]). On the other hand, it turns out that all forcing models of ZF are realisability models of ZF [**?**]. Moreover, there are ZF realisability models which cannot be obtained by forcing methods [**?**]: therefore, realisability for ZF is actually a stronger technique than forcing!

Let us conclude by mentioning two other important directions of research in realisability: there are categorical formulations of the various variants of it [van Oosten, 2008] (not only Krivine's version [Streicher, 2013]), and more algebraic formulations in terms of so-called implicative algebras [Miquel, 2020]. These are both important and central topics of research in realisability, and they are actually strictly related to each other.

Finally, keep in mind that while here we only talked about Krivine's realisability for classical logic (started after the '00), realisability is half a century older than that! In particular, it was invented by Kleene [Kleene, 1945] and much work in realisability happened (and still happens) in an intuitionistic setting; see also [van Oosten, 2002] for a historical presentation before Krivine's one existed.

References

[Fontanella and Geoffroy, 2020] Fontanella, L. and Geoffroy, G. (2020). Preserving cardinals and weak forms of zorn's lemma in realizability models. *Math. Struct. Comput. Sci.*, 30(9):976–996.

[Griffin, 1990] Griffin, T. (1990). A formulae-as-types notion of control. In Allen, F. E., editor, *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California, USA, January 1990*, pages 47–58. ACM Press.

[3] This is what we did in Exercise 5... for a very trivial formula, namely the polymorphic identity!

[4] This is an important weakness of today's formulations of classical realisability. Not only sometimes one is not able to show a realiser, but when one is able to do it, the realiser of a highly non-trivial axiom often happens to be quite trivial, which is not satisfying...

[5] Notice that we also use the same symbol "$\Vdash$".

[6] A hopefully more accessible and English version of which can be read in Chapter 3 of Barbarossa, Master thesis 2018

[Kleene, 1945] Kleene, S. C. (1945). On the interpretation of intuitionistic number theory. *The Journal of Symbolic Logic*, 10(4):109–124.

[Krivine, 2003] Krivine, J. (2003). Dependent choice, 'quote' and the clock. *Theor. Comput. Sci.*, 308(1-3):259–276.

[Krivine, 2008] Krivine, J. (2008). Structures de réalisabilité, RAM et ultrafiltre sur N. *CoRR*, abs/0809.2394.

[Krivine, 2011] Krivine, J. (2011). Realizability algebras: a program to well order R. *Log. Methods Comput. Sci.*, 7(3).

[Krivine, 2021] Krivine, J. (2021). A program for the full axiom of choice. *Log. Methods Comput. Sci.*, 17(3).

[Miquel, 2011] Miquel, A. (2011). Forcing as a program transformation. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 197–206. IEEE Computer Society.

[Miquel, 2020] Miquel, A. (2020). Implicative algebras: a new foundation for realizability and forcing. *Math. Struct. Comput. Sci.*, 30(5):458–510.

[Oliva and Streicher, 2008] Oliva, P. and Streicher, T. (2008). On krivine's realizability interpretation of classical second-order arithmetic. *Fundam. Informaticae*, 84(2):207–220.

[Streicher, 2013] Streicher, T. (2013). Krivine's classical realisability from a categorical perspective. *Math. Struct. Comput. Sci.*, 23(6):1234–1256.

[van Oosten, 2002] van Oosten, J. (2002). Realizability: a historical essay. *Mathematical Structures in Computer Science*, 12(3).

[van Oosten, 2008] van Oosten, J. (2008). *Realizability: An Introduction to its Categorical Side*, volume 152 of *Studies in Logic and the Foundations of Mathematics*. Elsevier.

3 Solutions to the exercises

Answer (Ex. 1) — The usual introduction rules for $\vee$ are admissible from Figure 1 in the following form:

$$\frac{\Gamma \vdash \mathtt{M} : L}{\Gamma \vdash \lambda \mathtt{lr}.\, \mathtt{lM} : L \vee R}\vee_l \qquad \frac{\Gamma \vdash \mathtt{M} : R}{\Gamma \vdash \lambda \mathtt{lr}.\, \mathtt{rM} : L \vee R}\vee_r$$

In order to show this, we have to show that, given a derivation $\mathfrak{q}$ of $\Gamma \vdash \mathtt{M} : L$, we have a derivation $\mathfrak{q}^\vee_l$ of $\Gamma \vdash \lambda \mathtt{lr}.\mathtt{lM} : L \vee R$, and similarly for the other. This derivation is defined in Figure 4, the other is similar. □

Answer (Ex. 2) — The derivation is:

$$\frac{\dfrac{}{\vdash \mathtt{callcc} : (\neg(A \vee \neg A) \to (A \vee \neg A)) \to (A \vee \neg A)}\,pl \qquad \dfrac{\dfrac{\mathfrak{p}}{\vdash \lambda \mathtt{yvh}.\, \mathtt{h}(\lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{zw}.\mathtt{zx})) : \neg(A \vee \neg A) \to (A \vee \neg A)}}{\vdash \mathtt{callcc}(\lambda \mathtt{yvh}.\, \mathtt{h}(\lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{zw}.\mathtt{zx}))) : A \vee \neg A}\,@$$

where $\mathfrak{p}$ is the derivation given in Figure 5. □

$$\dfrac{\dfrac{}{\Gamma, \mathtt{l} : L \to X, \mathtt{r} : R \to X \vdash \mathtt{l} : L \to X}\, proj \qquad \dfrac{\mathtt{q}^{Weak}_{(\mathtt{l}:L\to X, \mathtt{r}:R\to X)}}{\Gamma, \mathtt{l} : L \to X, \mathtt{r} : R \to X \vdash \mathtt{M} : L}}{\dfrac{\dfrac{\Gamma, \mathtt{l} : L \to X, \mathtt{r} : R \to X \vdash \mathtt{l}\mathtt{M} : X}{\Gamma \vdash \lambda \mathtt{l}\mathtt{r}.\, \mathtt{l}\mathtt{M} : (L \to X) \to (R \to X) \to X}\, \lambda, \lambda}{\Gamma \vdash \lambda \mathtt{l}\mathtt{r}.\, \mathtt{l}\mathtt{M} : L \vee R}\, \forall^0}\, @$$

Figure 4: Definition of the derivation $\mathtt{q}^{\vee}_{l}$ of $\Gamma \vdash \lambda \mathtt{l}\mathtt{r}.\mathtt{l}\mathtt{M} : L \vee R$, given $\mathtt{q}$ of $\Gamma \vdash \mathtt{M} : L$. For fresh variables $\mathtt{l}, \mathtt{r}$ in a derivation $\mathtt{q}$, the derivation $\mathtt{q}^{Weak}_{(\mathtt{l}:H_1, \mathtt{r}:H_2)}$ is defined as $\mathtt{q}$ but where, in each of its rules, we add at the end of the context the two new variable declarations $\mathtt{l} : H_1, \mathtt{r} : H_2$. Remark that the need of weakening $\mathtt{q}$ is the only reason why the rules for $\vee$ are, strictly speaking, only *admissible* and not *derivable*. But they are "morally" derivable.

$$\dfrac{\dfrac{\dfrac{}{\mathtt{y} : \neg(A \vee \neg A), \mathtt{x} : A \vdash \mathtt{y} : \neg(A \vee \neg A)}\, proj \qquad \dfrac{}{\mathtt{y} : \neg(A \vee \neg A), \mathtt{x} : A \vdash \mathtt{x} : A}\, proj}{\mathtt{y} : \neg(A \vee \neg A), \mathtt{x} : A \vdash \lambda \mathtt{z}\mathtt{w}.\mathtt{z}\mathtt{x} : A \vee \neg A}\, \vee_l}{\dfrac{\dfrac{\mathtt{y} : \neg(A \vee \neg A), \mathtt{x} : A \vdash \mathtt{y}(\lambda \mathtt{z}\mathtt{w}.\mathtt{z}\mathtt{x}) : \bot}{\mathtt{y} : \neg(A \vee \neg A) \vdash \lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{z}\mathtt{w}.\mathtt{z}\mathtt{x}) : \neg A}\, \lambda}{\mathtt{y} : \neg(A \vee \neg A) \vdash \lambda \mathtt{v}\mathtt{h}.\, \mathtt{h}(\lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{z}\mathtt{w}.\mathtt{z}\mathtt{x})) : A \vee \neg A}\, \vee_r}{\vdash \lambda \mathtt{y}\mathtt{v}\mathtt{h}.\, \mathtt{h}(\lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{z}\mathtt{w}.\mathtt{z}\mathtt{x})) : \neg(A \vee \neg A) \to (A \vee \neg A)}\, \lambda\, @$$

Figure 5: Derivation $\mathtt{p}$ from the solution of Exercise 1. We use (in dashed lines) the admissible rules of Exercise 1, so this actually stands for the derivation which is obtained by replacing the subderivations above the admissible rules by their respective derivations (this just decodes $\vee$ and puts the appropriate variable declarations in the context).

Answer (Ex. 3) — We have to show that $\lambda \mathtt{x}.\, \mathtt{callcc}(\lambda \mathtt{y}.\, \mathtt{x}\, \mathtt{y}) \star \mathtt{M} :: \pi \in \perp\!\!\!\perp$ for all $M \in \mathcal{W}(\neg\neg A)$ and $\pi \in \mathcal{C}(A)$. From Example 2.6 it suffices to show that $\mathtt{M} \star \mathtt{k}_{\pi} :: \pi \in \perp\!\!\!\perp$. But Example 2.7 tells us that $\mathtt{k}_{\pi} \in \mathcal{W}(\neg A)$. So the desired result follows from $M \in \mathcal{W}(\neg A \to \bot)$ and $\pi \in \Pi_c = \mathcal{C}(\bot)$. □

Answer (Ex. 4) — We have to show that

$$\mathtt{callcc}(\lambda \mathtt{y}\mathtt{v}\mathtt{h}.\, \mathtt{h}(\lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{z}\mathtt{w}.\, \mathtt{z}\mathtt{x}))) \Vdash A \vee \neg A.$$

Remark that the proof-term is closed. We have to show that, for each pole $\perp\!\!\!\perp$ and each $P \subseteq \Pi_c$, it witnesses $((A \to X) \to (\neg A \to X) \to X)\{ \mid X := P\}$. That is, that for all $\mathtt{M} \in \mathcal{W}((A \to X)\{ \mid X := P\})$, $\mathtt{N} \in \mathcal{W}((\neg A \to X)\{ \mid X := P\})$ and $\pi \in \mathcal{C}(X\{ \mid X := P\})$, we show that $\mathtt{callcc}(\lambda \mathtt{y}\mathtt{v}\mathtt{h}.\, \mathtt{h}(\lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{z}\mathtt{w}.\, \mathtt{z}\mathtt{x}))) \star \mathtt{M} :: \mathtt{N} :: \pi \in \perp\!\!\!\perp$. In order to show it, let us run the process in the KAM:

$$\begin{array}{rccccc}
 & \mathtt{callcc}(\lambda \mathtt{y}\mathtt{v}\mathtt{h}.\, \mathtt{h}(\lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{z}\mathtt{w}.\, \mathtt{z}\mathtt{x}))) & \star & \mathtt{M} & :: \mathtt{N} :: \pi \\
\succ & \mathtt{callcc} & \star & \lambda \mathtt{y}\mathtt{v}\mathtt{h}.\, \mathtt{h}(\lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{z}\mathtt{w}.\, \mathtt{z}\mathtt{x}))) & :: \mathtt{M} :: \mathtt{N} :: \pi \\
\succ & \lambda \mathtt{y}\mathtt{v}\mathtt{h}.\, \mathtt{h}(\lambda \mathtt{x}.\, \mathtt{y}(\lambda \mathtt{z}\mathtt{w}.\, \mathtt{z}\mathtt{x}))) & \star & \mathtt{k}_{\mathtt{M}::\mathtt{N}::\pi} & :: \mathtt{M} :: \mathtt{N} :: \pi \\
\succ & \mathtt{N}(\lambda \mathtt{x}.\, \mathtt{k}_{\mathtt{M}::\mathtt{N}::\pi}(\lambda \mathtt{z}\mathtt{w}.\, \mathtt{z}\mathtt{x})) & \star & & \pi \\
\succ & \mathtt{N} & \star & \lambda \mathtt{x}.\, \mathtt{k}_{\mathtt{M}::\mathtt{N}::\pi}(\lambda \mathtt{z}\mathtt{w}.\, \mathtt{z}\mathtt{x}) & :: \pi
\end{array}$$

Since the pole is closed by anti-reduction, it suffices to show that this last process is in it. But by hypothesis $\mathtt{N} \in \mathcal{W}((\neg A \to X)\{ \mid X := P\})$, so it suffices for that to show that

$\lambda \mathtt{x}.\, \mathtt{k}_{\mathtt{M}::\mathtt{N}::\pi}(\lambda \mathtt{zw}.\, \mathtt{zx}) \in \mathcal{W}(\neg A)$. In order to show this, let $\mathtt{H} \in \mathcal{W}(A)$ and $\rho \in \mathcal{C}(\bot) = \Pi_c$, and we show that $\lambda \mathtt{x}.\, \mathtt{k}_{\mathtt{M}::\mathtt{N}::\pi}(\lambda \mathtt{zw}.\, \mathtt{zx}) \star \mathtt{H} :: \rho \in \perp\!\!\!\perp$. For this, let us run the process in the KAM:

$$\begin{array}{rccccc}
 & \lambda \mathtt{x}.\, \mathtt{k}_{\mathtt{M}::\mathtt{N}::\pi}(\lambda \mathtt{zw}.\, \mathtt{zx}) & \star & \mathtt{H} & :: \rho \\
\succ & \mathtt{k}_{\mathtt{M}::\mathtt{N}::\pi}(\lambda \mathtt{zw}.\, \mathtt{zH}) & \star & & \rho \\
\succ & \mathtt{k}_{\mathtt{M}::\mathtt{N}::\pi} & \star & \lambda \mathtt{zw}.\, \mathtt{zH} & :: \rho \\
\succ & \lambda \mathtt{zw}.\, \mathtt{zH} & \star & \mathtt{M} & :: \mathtt{N} :: \pi \\
\succ & \mathtt{MH} & \star & & \pi \\
\succ & \mathtt{M} & \star & \mathtt{H} & :: \pi
\end{array}$$

Now, by hypothesis we have $\mathtt{M} \in \mathcal{W}((A \to X)\{ \mid X := P\})$ and also $\mathtt{H} \in \mathcal{W}(A)$, so indeed $\mathtt{M} \star \mathtt{H} :: \pi \in \perp\!\!\!\perp$ by definition of the witnesses of an implication. □

Answer (Ex. 5) — Suppose that $\mathtt{M} \approx_1 \mathtt{I}$. Fix a pole $\perp\!\!\!\perp$ and $P \subseteq \Pi_c$. We have to show that $\mathtt{M} \in \mathcal{W}((X \to X)\{X := P\})$, i.e. $\mathtt{M} \star \mathtt{N} :: \pi \in \perp\!\!\!\perp$ for all $\mathtt{N} \in \mathcal{W}(X\{X := P\})$ and $\pi \in \mathcal{C}(X\{X := P\})$. But by hypothesis we have $\mathtt{M} \star \mathtt{N} :: \pi \succ \mathtt{N} :: \pi$, and the latter belongs to $\perp\!\!\!\perp$ by construction. Since $\perp\!\!\!\perp$ is $\succ$-downward closed, we are done.
For the converse, suppose that $\mathtt{M} \Vdash \forall X^0.\, X \to X$. For $\mathtt{N} \in \Lambda_c, \pi \in \Pi_c$, we have to show that $\mathtt{M} \star \mathtt{N} :: \pi \succ \mathtt{N} \star \pi$. But the set $\perp\!\!\!\perp := \{p \in \mathrm{Proc} \mid p \succ \mathtt{N} \star \pi\}$ is trivially $\succ$-downward closed, so we can take it as a pole: the hypothesis says then that $\mathtt{M} \in \mathcal{W}((X \to X)\{X := P\})$ (for the $\mathcal{W}$ relative to that pole) for all $P \subseteq \Pi_c$. In particular, choosing $P := \{\pi\}$, we have that $\mathtt{M} \perp\!\!\!\perp S :: \rho$ for all $\rho \in \mathcal{C}(X\{X := \{\pi\}\}) = \{\pi\}$ and $S \in \mathcal{W}(X\{X := \{\pi\}\}) = \{\pi\}^{\perp\!\!\!\perp}$. Now, we remark that $\mathtt{N} \perp\!\!\!\perp \pi$, because $\mathtt{N} \star \pi \in \perp\!\!\!\perp$ by definition of $\perp\!\!\!\perp$. Therefore $\mathtt{M} \perp\!\!\!\perp \mathtt{N} :: \pi$ and, by definition of $\perp\!\!\!\perp$, we have the desired $\mathtt{M} \star \mathtt{N} :: \pi \succ \mathtt{N} \star \pi$. □