Constructive Logic and Realisability

Dirk Pattinson

ANU Logic Summer School, December 2013

1 Introduction

This is a (very?) rough write-up of a series of five lectures given at the ANU logic summer school 2013. Largely, these notes are based on Thomas Streicher’s notes ‘Introduction to Constructive Logics and Mathematics’ that I recommend for further reading. More references are Troelstra 1973, Metamathematical Investigations into Intuitionistic Arithmetic’, Schwichtenberg and Wainer, Proof and Computation (not for the faint-hearted!), and the pointers on Jaap van Oosten’s web page on realizability: http://www.staff.science.uu.nl/~ooste110/realizability.html.

2 Constructive Reasoning

2.1 Examples and Motivation

Lemma 2.1. There are two irrational numbers $a$ and $b$ such that $a^b$ is rational.

Proof. By case analysis on whether $\sqrt{2}^\sqrt{2}$ is rational or not. \qed

Write $\{\cdot\}$ for Kleene application, i.e. $\{n\}(m)$ is the result of running the $m$-th Turing machine on input $m$. Write $x\downarrow$ to say that $x$ is defined.

Lemma 2.2. For all $n, m$: either $\{n\}(m)\downarrow$ or $\{n\}(m)\uparrow$.

Lemma 2.3. The formula $\exists x(D(x) \rightarrow \forall yD(y))$ is a tautology in classical first-order logic.

We take the point of view here that this is counter-intuitive, as the proof of either of these lemmas does not give any evidence regarding the asserted statement.
2.2 Brower-Heyting-Kolmogorov Interpretation

The BHK interpretation is an informal (yet helpful) device that explains what we mean by a constructive proof.

1. A proof of $A \land B$ is a pair $(\pi_1, \pi_2)$ so that $\pi_1$ is a proof of $A$ and $\pi_2$ is a proof of $B$.

2. A proof of $A \rightarrow B$ is a function $f$ on proofs such that $f(\pi)$ is a proof of $B$ whenever $\pi$ is a proof of $A$.

3. A proof of $A \lor B$ is a pair $(i, \pi)$ so that $\pi$ is a proof of $A$ if $i = 0$ and $\pi$ is a proof of $B$ if $i > 0$.

4. A proof of $\forall x A(x)$ is a function $f$ such that $f(d)$ is a proof of $A(d)$ for all $d \in D$, a domain of discourse.

5. A proof of $\exists x. A(x)$ is a pair $(d, \pi)$ where $d \in D$, some domain of discourse, and $\pi$ is a proof of $A(d)$.

6. There is no proof of $\bot$.

It is the goal of these notes to present some ideas on how to make the BHK-interpretation more formal.

2.3 Constructive Proof, Formally

We present a natural deduction calculus of standard first-order logic. Our basic signature consists of function symbols and relation symbols, the propositional connectives (excluding $\neg$) and universal/existential quantification. We write $\neg A$ for $A \rightarrow \bot$. We present natural deduction in sequent style. A sequent is a structure $\Gamma \vdash A$ where $\Gamma$ is a multiset of formulae, and $A$ is a single formula. We read $\Gamma \vdash A$ as “the assumptions in $\Gamma$ entail (the conclusion) $B$.” Below we have introduction rules on the left, and elimination rules on the right.

Conjunction

\[
\begin{array}{c}
\Gamma \vdash A \\
\Gamma \vdash B
\end{array}
\quad
\begin{array}{c}
\Gamma \vdash A \land B
\end{array}
\]

Implication

\[
\begin{array}{c}
\Gamma, A \vdash B
\end{array}
\quad
\begin{array}{c}
\Gamma \vdash A \rightarrow B
\end{array}
\]

\[
\begin{array}{c}
\Gamma \vdash A
\end{array}
\quad
\begin{array}{c}
\Gamma \vdash A
\end{array}
\quad
\begin{array}{c}
\Gamma \vdash B
\end{array}
\]
Disjunction

\[
\begin{align*}
\Gamma &\vdash A & &\Gamma &\vdash B & &\Gamma &\vdash A \lor B & &\Gamma, A &\vdash C & &\Gamma, B &\vdash C \\
\hline
\Gamma &\vdash A \lor B & &\Gamma &\vdash A \lor B & &\Gamma &\vdash C
\end{align*}
\]

Falsum

\[
\Gamma \vdash \bot \\
\hline
\Gamma \vdash A
\]

(We write \(\text{FV}(\Gamma)\) for the set of free variables that occur in \(\Gamma\)).

Universal Quantification

\[
\begin{align*}
\Gamma &\vdash A(x) & &\Gamma &\vdash \forall x. A(x) (x \notin \text{FV}(\Gamma)) & &\Gamma &\vdash A(t)
\end{align*}
\]

Existential Quantification

\[
\begin{align*}
\Gamma &\vdash A(t) & &\Gamma &\vdash \exists x. A(x) & &\Gamma, A(x) &\vdash C (x \notin \text{FV}(\Gamma, C))
\end{align*}
\]

Others: Axioms, Weakening, Contraction

\[
\begin{align*}
\Gamma, A &\vdash A & &\Gamma &\vdash B & &\Gamma &\vdash C & &\Gamma, A &\vdash B
\end{align*}
\]

Exercise 2.4.  
1. Convince yourself that the variable conditions are necessary: exhibit a derivation where the variable condition has not been obeyed and argue that the (fraudulently) derived sequent should not be derivable.

2. Argue that this calculus is consistent with the BHK interpretation by constructing proofs as required. What is the role of the variable condition here?

3. Show that the cut rule

\[
\begin{align*}
\Gamma &\vdash A & &\Gamma, A &\vdash B \\
\hline
\Gamma &\vdash B
\end{align*}
\]

is admissible in the natural deduction calculus for constructive first-order logic.
2.4 The Babelfish: From Classical to Constructive

Our interpretation of constructive first-order logic is that it adds two connectives to classical logic: $\exists$ and $\lor$. The homonymous connectives in classical logic are just translations. This is the Goedel-Gentzen translation:

$$
\bot^G = \bot \\
P^G = \neg P \\
(A \land B)^G = A^G \land B^G \\
(A \lor B)^G = \neg (\neg A^G \land \neg B^G) \\
(A \to B)^G = A^G \to B^G \\
(\forall x.A)^G = \forall x.A^G \\
(\exists x.A)^G = \neg \forall x.\neg A^G
$$

In the sequel, we say that $\Gamma \vdash A$ is classically derivable if it is derivable with the help of reductio ad absurdum:

$$
\frac{\Gamma \vdash \neg \neg A}{\Gamma \vdash A}
$$

We show how reductio ad absurdum proves the law of excluded middle. First, as an exercise, you are asked to give a derivation of

$$
A \to C \land B \to C \vdash A \lor B \to C
$$

Instantiating, we obtain

$$
\neg A \land \neg A \vdash \neg (A \lor \neg A)
$$

Using that $\neg (\neg A \land \neg A)$ is derivable, we obtain $\neg \neg (A \lor \neg A)$, and reductio ad absurdum gives $\vdash A \lor \neg A$.

Exercise 2.5. 1. Make the details of the previous derivation explicit!

2. show that $\neg \bot \to \bot$ is constructively derivable. I.e. we could have defined $\bot^G$ as $\neg \bot$ but it would have made no difference. (One possible tree is the following:

```
  ¬¬⊥ ⊢ ¬¬⊥  ¬¬⊥ ⊢ ⊥  ¬¬⊥, ⊥ ⊢ ⊥
  ¬¬⊥ ⊢ ¬⊥ → ⊥  ¬¬⊥ ⊢ ⊥
  ¬¬⊥ ⊢ ⊥
```
3. Show that \( A \vdash \neg
\neg A \) is derivable for every formula \( A \). In other words, every formula implies its double negation, but not vice versa.

**Theorem 2.6.** 1. In classical propositional calculus, we have that \( A \) is derivable if and only if \( A^G \) is derivable.

2. We have that \( \Gamma \vdash A \) classically if and only if \( \Gamma^G \vdash A^G \) constructively.

The proof of the theorem uses the fact that every formula of the form \( A^G \) is double negation closed: We can derive \( \neg
\neg A^G \vdash A^G \) constructively.

**Exercise 2.7.** Give an example of a formula \( A \) such that \( A \) and \( A^G \) are not equivalent in constructive first-order logic (i.e. one of \( A \rightarrow A^G \) or \( A^G \rightarrow A \)) is not constructively derivable. Which?

### 3 Heyting Arithmetic

#### 3.1 Primitive Recursive Functions

We recall the definition of the set of primitive recursive functions of type \( \mathbb{N}^n \to \mathbb{N} \) (\( n \geq 0 \)).

1. \( 0 : \mathbb{N}^0 \to \mathbb{N} \) is primitive recursive (the constant 0)

2. \( s : \mathbb{N} \to \mathbb{N} \) is primitive recursive. (the successor function)

3. all functions \( p^n_k : \mathbb{N}^n \to \mathbb{N} \), defined by \( p^n_k(x_1, \ldots, x_n) = x_k \), are primitive recursive.

4. If \( f : \mathbb{N}^n \) and \( g_1, \ldots, g_n : \mathbb{N}^k \to \mathbb{N} \) are primitive recursive, then so is the function \( h : \mathbb{N}^k \to \mathbb{N} \), defined by \( h(x_1, \ldots, x_k) = f(g(x_1, \ldots, x_k), \ldots, g_n(x_1, \ldots, x_k)) \).

5. if \( f : \mathbb{N}^{n+2} \to \mathbb{N} \) and \( g : \mathbb{N}^n \to \mathbb{N} \) are primitive recursive, then so is the function \( h : \mathbb{N}^{n+1} \to \mathbb{N} \), defined by \( h(0, x_1, \ldots, x_n) = g(x_1, \ldots, x_n) \) and \( h(n+1, x_1, \ldots, x_n) = f(n, h(n, x_1, \ldots, x_n), x_1, \ldots, x_n) \).

Primitive recursive functions allow pairing:

**Proposition 3.1.** There are primitive recursive functions \( p_0, p_1 : \mathbb{N} \to \mathbb{N} \) and \( \langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N} \) such that

\[
p_0(\langle n, m \rangle) = n \quad p_1(\langle n, m \rangle) = m \quad \langle p_0(k), p_1(k) \rangle = k
\]

for all \( n, m, k \in \mathbb{N} \).
It is a result of Kleene that the class of all recursive functions is obtained by adding minimisation to the definition of primitive recursive functions. More precisely, the class of recursive functions contains the class of all primitive recursive functions, together with the clause

6. If $f : \mathbb{N}^{n+1} \to \mathbb{N}$ is recursive, then so is $\mu(f) : \mathbb{N}^k \to \mathbb{N}$ defined by $\mu(f)(x_1, \ldots, x_n) = \min\{k \in \mathbb{N} \mid f(k, x_1, \ldots, x_n) = 0\}$ if the latter set is non-empty, and $\mu(f)(x_1, \ldots, x_n)$ is undefined, otherwise.

What is important for us later is the Kleene normal form theorem. In a nutshell, there are two main parts:

1. There is a primitive recursive $T : \mathbb{N}^3 \to \mathbb{N}$ such that $T(n, m, k) = 0$ iff $k$ represents a terminating computation of the $n$-th Turing machine applied to $m$.

2. There is a primitive recursive function $U : \mathbb{N} \to \mathbb{N}$ such that $U(k)$ is the result (tape content) of the computation coded by $k$.

Kleene’s normal form theorem then says that every (partial) recursive function can be represented as

$$f(n) = U(\mu k. T(n, m, k))$$

using primitive recursive functions. I.e. one application of minimisation on the outer level is enough. (Kleene’s $T$ predicate is sometimes defined as $T(n, m, k) = 0$.)

The other ingredient that we need from recursive function theory is the so-called parametrisation theorem:

**Proposition 3.2.** For every partial recursive function $f : \mathbb{N}^{k+1} \to \mathbb{N}$ there is a primitive recursive(!) function $g : \mathbb{N}^k \to \mathbb{N}$ such that $f(x_1, \ldots, x_n, y) = \{g(x_1, \ldots, x_n)\}(y)$ for all $x_1, \ldots, x_n, y \in \mathbb{N}^k$.

We write the function $g$ above as $\Lambda f$ such that $\{\Lambda f(x_1, \ldots, x_n)\}(y) = f(x_1, \ldots, x_n, y)$. In particular, if $t$ is a primitive recursive term mentioning variables $x_1, \ldots, x_n, y$ then there is a primitive recursive term $\Lambda t.y$ such that $\{\Lambda t.y\}(y) = t$.

Moreover, Kleene-Application, i.e. computing the result $\{n\}(m)$ of running the $n$-th Turing machine on input $m$ is recursive in $n$ and $m$ jointly. Precisely:

**Proposition 3.3.** The function $\{\cdot\}(\cdot) : \mathbb{N}^2 \to \mathbb{N}$, i.e. $(n, m) \mapsto \{n\}(m)$ is recursive.
3.2 Definition of Heyting Arithmetic

Heyting Arithmetic is the language containing a function symbol for every primitive recursive function and equality. The axioms of Heyting arithmetic are the defining equations for primitive recursive functions (as above: not all equalities that hold under some interpretation) and the equality axioms

\[ t = t \quad A(t) \land t = s \rightarrow A(s) \]

and the assertion that \( 1 \neq 0 \) plus induction, i.e.

\[ \neg(0 = s(0)) \quad A(0) \land (\forall x (A(x) \rightarrow A(s(x)))) \rightarrow \forall x (A(x)) \]

where \( A \) is any formula of Heyting arithmetic. We write \( \text{HA} \vdash A \) if \( A \) is a formula of Heyting arithmetic and \( A \) can be derived in natural deduction using the above additional axioms.

The language of Peano Arithmetic is the same as that of Heyting Arithmetic, but based on classical first-order logic. We say that \( A \) is derivable in Peano arithmetic (\( \text{PA} \vdash A \)) if it is derivable using the axioms and rules of Heyting arithmetic, plus additionally reductio ad absurdum

\[
(\text{RAA}) \quad \Gamma \vdash \neg
\]

A question that puzzled logicians for many years is whether constructive logic is ‘safer’ than classical logic. For derivability, the double-negation theorem says that the answer is ‘no’: If \( \bot \) is derivable classically, then \( \bot \) is derivable constructively.

This does not answer the question about whether or not Heyting arithmetic is ‘safer’ than peano arithmetic, as the above translation does not immediately translate to theories. But we have the following:

**Proposition 3.4.** If \( S \) is a set of sentences such that \( A^G \) is constructively derivable from \( T \), for every \( A \in S \), then \( T \vdash A^G \) constructively whenever \( T \vdash A \) classically.

The proof of that is simple: if \( T \vdash A \) classically, then there are finitely many axioms \( \Gamma = A_1, \ldots, A_n \in T \) such that \( \Gamma \vdash A \) classically whence \( \Gamma^G \vdash A^G \) constructively, and using (cut) it follows that \( A^G \) is derivable from \( T \) constructively. \( \Gamma \vdash A \) classically whence \( \Gamma^G \vdash A^G \) constructively, and using (cut) it follows that \( A^G \) is derivable from \( T \) constructively.

In particular, Heyting arithmetic is not safer than Peano arithmetic as all axioms of Heyting arithmetic are left unchanged by \((\cdot)^G\).
3.3 Informal Realisability

Informal realisability is a relation between numbers and formulae of Heyting Arithmetic defined as follows:

\( n \text{ ri } s = t \iff s = t \)
\( n \text{ ri } A \land B \iff p_0(n) \text{ ri } A \text{ and } p_1(n) \text{ ri } B \)
\( n \text{ ri } A \lor B \iff p_0(n) = 0 \text{ and } p_1(n) \text{ ri } A \text{ or } p_0(n) \neq 0 \text{ and } p_1(n) \text{ ri } B \)
\( n \text{ ri } \forall x. A(x) \iff \text{ for all } m \in \mathbb{N} \text{ for which } m \text{ ri } A \text{ we have that } \{n\}(m) \downarrow \text{ and } \{n\}(m) \text{ ri } B \)
\( n \text{ ri } \exists x. A(x) \iff p_1(n) \text{ ri } A(p_0(n)) \)

Note that \( \bot \) does not have a realiser.

Exercise 3.5. Discuss the above definition. What does it mean to say \( s = t \) in the first clause? In what sense can \( s = t \) be ‘true’? In what meta-theory can the definition be embedded?

To get a feeling for realisability, consider the statement

\( n \text{ ri } \forall x. \exists y. P(x, y) \)

where \( P \) is a primitive recursive predicate. Unravelling the definitions, we have a recursive function, \( \phi(x) = \{n\}(x) \) such that \( \phi(x) \text{ ri } \exists y. P(x, y) \) for all \( x \).

This is to say that \( p_1(\phi(x)) \text{ ri } P(x, p_0(\phi(x))) \) for all \( x \), hence \( P(x, p_0(\phi(x))) \) holds for all \( x \): we have extracted a function that witnesses the truth of the predicate.

We write \( H(n) \) for the predicate \( \{n\}(n) \downarrow \), i.e.

\( H(n) = \exists k. T(n, n, k) = 0 \)

so that \( H(n) \) if and only if the \( n \)-th Turing machine halts on input \( n \).

Exercise 3.6. Show that the halting statement \( \forall x.(H(x) \lor \neg H(x)) \) cannot have a realiser by showing that, if \( n \text{ ri } \forall x. H(x) \lor \neg H(x) \) then the function \( n \mapsto p_0(\{x\}(n)) \) would decide the halting problem. Note that this statement is classically true. Does the negation of the halting statement, viz \( \neg \forall n(H(n) \lor \neg H(n)) \) have a realiser?

Now consider the proposition commonly known as Church’s Thesis (all functions are recursive):

\[ (CT) \quad \forall n \exists m P(n, m) \rightarrow \exists k \forall n. P(n, \{k\}(n)) \]
We can re-write

\[ P(n, \{k\}(n)) \equiv \exists e. T(k, n, e) \land P(n, U(e)) \]

so that Church’s thesis is actually a formula of Heyring arithmetic.

**Exercise 3.7.** Show that (CT) has a realiser, and that CT is in fact classically false.

One application of informal realisability is to show consistency of Heyting arithmetic. We will see later that every provable formula in fact has a realiser whence e.g. the formula \( \bot \) cannot be provable. Moreover, this extends if we adopt principles that are realisable, but classically false, such as e.g. (CT0).

Another axiom we may wish to add is Markov’s principle: if we can prove that it is absurd that a computation diverges, then it must terminate. As a formula, this reads as

\[(MP) (\forall x. A(x) \lor \neg A(x)) \to \neg \forall x. \neg A(x) \to \exists x. A(x)\]

The crucial observation is that the property \( A(x) \) is required to be decidable, i.e. we must have constructive means to see whether or not \( A(x) \) actually holds.

Consider e.g. the property \( A(k) = T(n, n, k) = 0 \) that expresses that \( \{n\}(n) \downarrow \). We know that \( T \) is a primitive recursive function so that \( T(n, n, k) = 0 \) is decidable, simply by evaluation. Markov’s principle then allows us to state that \( \{n\}(n) \) terminates if we can show that it is absurd that it doesn’t. A realiser for Markov’s principle is found by using minimisation (see the definition of partial recursive functions above).

### 3.4 Number Realisability

We address the question of meta-theory by formalising realisability within Heyting arithmetic: this is the notion of realisability that we study in the sequel. We define formulae \( n \text{ rn } A \) of Heyting arithmetic.

\[
\begin{align*}
\text{n rn } P & \equiv P \\
\text{n rn } A \land B & \equiv p_0(n) \text{ rn } A \land p_1(n) \text{ rn } B \\
\text{n rn } A \lor B & \equiv (p_0(n) = 0 \leftrightarrow p_1(n) \text{ rn } A) \land (p_0(n) \neq 0 \leftrightarrow p_1(n) \text{ rn } B) \\
\text{n rn } A \rightarrow B & \equiv \forall m. m \text{ rn } A \rightarrow \{n\}(m) \text{ rn } B \\
\text{n rn } \forall x. A(x) & \equiv \forall m. \{n\}(m) \text{ rn } A(m) \\
\text{n rn } \exists x. A(x) & \equiv p_0(n) \text{ rn } A(p_1(n)) \\
\end{align*}
\]

where \( P \) is primitive, i.e. \( P = \bot \) or \( P = s = t \).
Theorem 3.8. Suppose that $A_1, \ldots, A_k \vdash A_0$ is derivable in Heyting arithmetic. Then there exists a term $t$, possibly containing Kleene application, such that $x_1 \text{rn } A_1, \ldots, x_k \text{rn } A_k \vdash t \text{rn } A$ is derivable in Heyting arithmetic, where $x_1, \ldots, x_k$ are new variables and $\text{FV}(t) \subseteq \{x_1, \ldots, x_n\} \cup \text{FV}(A_0, \ldots, A_n)$.

We do not give a full proof of this theorem but instead discuss some of the cases. One interesting case is the rule of $\rightarrow$-introduction, i.e. the conclusion has been derived using

\[
\Gamma, A \vdash B \\
\Gamma \vdash A \rightarrow B
\]

We assume that $\Gamma = A_1, \ldots, A_n$, $\bar{x} = x_1, \ldots, x_n$ and write $\bar{x} \text{rn } \Gamma$ to mean $x_1 \text{rn } A_1, \ldots, x_n \text{rn } A_n$ where $x_1, \ldots, x_n$ are fresh variables.

We also assume that the free variables of $A_1, \ldots, A_n, A, B$ are $\bar{y} = y_1, \ldots, y_k$. Using this notation, we have, by induction hypothesis, that there exists a term $s$ such that

\[
\bar{x} \text{rn } \Gamma, z \text{rn } A \vdash s \text{rn } B.
\]

Using the parametrisation theorem, there exists a primitive recursive term $t$ such that

\[
\{t(\bar{x}, \bar{y})\}(z) = s(\bar{x}, \bar{y}, z)
\]

and we can therefore re-write the induction hypothesis to obtain the premise of the above derivation:

\[
x \text{rn } \Gamma \vdash z \text{rn } A \rightarrow \{t(\bar{x}, \bar{y})\}(z) \text{rn } B \\
x \text{rn } \Gamma \vdash \forall z. (z \text{rn } A \rightarrow \{t(\bar{x}, \bar{y})\}(z) \text{rn } B) \\
x \text{rn } \Gamma \vdash \{t(\bar{x}, \bar{y})\} \text{rn } A \rightarrow B
\]

where the last double line indicates unfolding of the definition of $\text{rnt}$. Note that the variable condition in $\forall$-introduction is satisfied as $z$ is a fresh variable by assumption.

The case of $\rightarrow$-elimination is a lot easier. Assuming that the conclusion of the proof has been derived via

\[
\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A \\
\Gamma \vdash B
\]

and using notation similar as above, we have terms $s, t$ such that

\[
\bar{x} \text{rn } \Gamma \vdash s \text{rn } A \rightarrow B \quad \bar{x} \text{rn } \Gamma \vdash t \text{rn } A
\]

is derivable. This means that the partial derivation that uses the above two judgements as premises
can be completed to a proof that shows that we have a realiser for $B$.

Now let's consider the case of $\land$-introduction, i.e. the last step of the proof is

$$\Gamma \vdash A \quad \Gamma \vdash B$$

hypothesis, we have terms $s, t$ such that

$$x \Gamma \vdash s \Gamma A \quad x \Gamma \vdash t \Gamma B$$

and thus the following derivation.

$$p_0(s, t) = s$$

$$x \Gamma \vdash p_0(s, t) = s \quad x \Gamma \vdash s \Gamma A \quad p_0(s, t) = s \land s \Gamma A \rightarrow p_0(s, t) \Gamma A$$

$$x \Gamma \vdash p_0(s, t) = s \land s \Gamma A \rightarrow p_0(s, t) \Gamma A$$

$$x \Gamma \vdash p_0(s, t) \Gamma A$$

We have an analogous derivation that ends in

$$x \Gamma \vdash p_1(s, t) \Gamma B$$

and putting these together we obtain

$$x \Gamma \vdash p_0(s, t) \Gamma A \quad x \Gamma \vdash p_1(s, t) \Gamma B$$

$$x \Gamma \vdash \langle p_0(s, t) \Gamma A \land p_1(s, t) \Gamma A$$

$$x \Gamma \vdash \langle s, t \rangle \Gamma A \land B$$

where the double line indicates unfolding of the definition of $\Gamma$.

To complete the proof, we also need to produce realisers for all axioms of Heyting arithmetic. It is instructive to construct a realiser, in particular for the induction axiom.

**Exercise 3.9.** Show that the induction axiom of HA has a realiser, and fill in the remaining details of the proof.

We have not discussed the role of the meta-theory (and will refrain from doing so) but mention the following consequence:
Proposition 3.10. Heyting arithmetic is consistent.

For if Heyting arithmetic were not consistent, we would obtain a term \( t \) such that \( \text{HA} \vdash t \text{rn} \bot \) from which we conclude that \( t \text{ri} \bot \) which is impossible as \( \bot \) has no realiser.

The intricacies of this depend on what we mean by ‘true’ in the definition of informal realisability: technically we assume that we can map formulae in Heyting arithmetic to a stronger (and hopefully consistent) meta-theory so that consistency can then be formalised as relative consistency with regards to the chosen meta-theory.

3.5 Disjunction and Existence Property

It may be considered as a defect of formalised realisability that the statement \( n \text{rn} A \) does not imply that \( A \) is derivable.

This is remedied if we modify realisability slightly and move to realisability with truth. We defined, as in formalised realisability, the formula \( n \text{rn} A \) by the same clauses, the only difference being the clause for implication:

\[
 n \text{rn} A \rightarrow B \equiv (\forall m. m \text{rn} A \rightarrow \{n\}(m) \text{rn} B) \land (A \rightarrow B)
\]

Using this definition, we can still prove that every formula has a realiser, and the disjunction / existence property follows. We have the following:

**Theorem 3.11.** Let \( A_1, \ldots, A_k \vdash A_0 \) be derivable in Heyting arithmetic. Then there exists a term \( t \), possibly containing Kleene application, such that \( x_1 \text{rnt} A_1, \ldots, x_k \text{rnt} A_k \vdash t \text{rnt} A \) is derivable in Heyting arithmetic, where \( x_1, \ldots, x_k \) are new variables and \( \text{FV}(t) \subseteq \{x_1, \ldots, x_n\} \cup \text{FV}(A_0, \ldots, A_n) \).

To prove this theorem, we only need to adapt the proof of the soundness theorem in case of implication introduction – but this is trivial. In particular, this allows us to show:

**Corollary 3.12.** If \( \text{HA} \vdash A \), and \( A \) is a sentence (i.e. does not contain any free variables), then there is a natural number \( n \) such that \( \text{HA} \vdash n \text{rnt} A \).

We can show the disjunction and existence property once we convince ourselves that all realisable formulae are in fact true. This is an easy induction on the structure of proofs.

**Theorem 3.13.** In Heyting arithmetic, the formula \( (\exists x. x \text{rnt} A) \rightarrow A \) is derivable for all formulae \( A \) with \( x \notin \text{FV}(A) \).
The variable condition in the theorem above is induced by the base case of a proof that proceeds on the structure of $A$. So assume that $A \equiv s = t$ and we have the proof tree

\[ \exists x. s = t \vdash \exists x. s = t, \quad \exists x. s = t, s = t \vdash s = t \]

\[ \vdash (\exists x. s = t) \rightarrow s = t \]

where we need to require that $x \notin \text{FV}(s = t)$ to apply $\exists$-elimination. Also note that by definition, $x \text{ rnt } s = t \equiv s = t$ so we have proved the base case.

(Also note that $(\exists x. x \text{ rnt } x = 5) \rightarrow x = 5$ cannot be provable, for instance.)

For $A \land B$ we have the following derivation:

\[
\begin{array}{c}
p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \vdash p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \\
p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \vdash p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } A \\
p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \vdash \exists x. x \text{ rnt } A \\
p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \vdash (\exists x. x \text{ rnt } A) \rightarrow A \\
p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \vdash A
\end{array}
\]

where we use the induction hypothesis on the left. A similar derivation shows that we can derive

\[ p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \vdash B \]

in Heyting arithmetic. Applying $\land$-introduction to these two judgements we have that the top of the proof tree below is actually derivable:

\[
\begin{array}{c}
p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \vdash A \land B \\
\exists x. p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \vdash A \land B \\
\exists x. p_0(x) \text{ rnt } A \land p_1(x) \text{ rnt } B \vdash A \land B
\end{array}
\]

Note that the introduction of the existential quantifier in this derivation in fact obeys the variable condition as $x \notin \text{FV}(A \land B)$.

We can now state disjunction and existence property as follows.

**Theorem 3.14.** Heyting arithmetic has the disjunction property and the existence property, viz:

1. if $HA \vdash A \lor B$ for sentences $A, B$ then $HA \vdash A$ or $HA \vdash B$.

2. if $HA \vdash \exists x. A(x)$ where $\exists x. A(x)$ is a sentence, then there is a term $t$ such that $HA \vdash A(t)$.  

13
The disjunction property is established as follows: If $\text{HA} \vdash A \rightarrow B$, then we have a realiser $n \ \text{rnt} \ A \lor B$. As $A$ is a sentence, $n$ is a term without free variables, i.e. a number. If $p_0(n) = 0$ then $\text{HA} \vdash p_1(n) \ \text{rnt} \ A$ from which we conclude that $\text{HA} \vdash A$. The same argument applies if $p_0(n) \neq 0$.

For the existence property, we have a realiser $\text{HA} \ \text{rnt} \ \exists x. A(x)$ so that $\text{HA} \vdash p_1(n) \ \text{rnt} \ A(p_0(n))$. As $p_0(n)$ is (closed) term, we have that $p_0(n)$ is in fact a number and $\text{HA} \vdash A(p_0(n))$ as desired.

4 Variations of the Theme

We have looked at Kleene’s number realisability in some detail. Realisability as a method is much more powerful. In a nutshell, we have a relation

$$n \ \text{realises} \ A$$

where $A$ is a formula of some logic, and $n$ is an object that represents the computational content of $A$. Here, we have used Heyting arithmetic and natural numbers (representing Turing machines), but both can be varied.

For the logic, we can consider e.g. Heyting arithmetic in higher types, where types are given by

$$T \ni \sigma, \tau ::= N \mid \sigma \times \tau \mid \sigma \rightarrow \tau$$

for a base type $N$ that represents numbers. We define functions using $\lambda$-abstractions, applications and a recursor of type

$$R : \sigma \rightarrow (N \rightarrow \sigma \rightarrow \Sigma) \rightarrow N \rightarrow \sigma$$

subject to the (usual) equations of the typed $\lambda$-calculus, together with

$$Rf0 = f \ \text{and} \ Rfgx(sx) = gx(Rfx)$$

which gives rise to Goedel’s system $T$. Realisability is then a typed notion, and we can use terms of this system directly as realisers. In particular, $R$ realises the induction axiom!

This language is strictly more powerful than Heyting arithmetic at base type, as we can e.g. define the Ackermann function.

We may also equip a given logic (at higher type or at type 0) with a different notion of realisers – as long as they embody computation. Possible candidates here are the set of untyped $\lambda$-terms, the second Kleene algebra (streams of natural numbers), or even programs in your favourite language!