Fundamentals of the CTT Approach

Shahid Rahman, Nicolas Clerbout

THE FOLLOWING TEXT IS AN OVERVIEW ON EXISTING LITERATURE: THERE IS NO CLAIM OF ORIGINALITY

Fundamentals of the CTT Approach[1]

By Shahid Rahman and Nicolas Clerbout

Within Per Martin-Löf's Constructive Type Theory (CTT for short) the logical constants are interpreted through the Curry-Howard correspondence between propositions and sets. A proposition is interpreted as a set whose elements represent the proofs of the proposition. It is also possible to view a set as a problem description in a way similar to Kolmogorov's explanation of the intuitionistic propositional calculus. In particular, a set can be seen as a specification of a programming problem: the elements of the set are then the programs that satisfy the specification (Martin-Löf 1984, p. 7). Furthermore in CTT sets are also understood as types so that propositions can be seen as data (or proof)-types.[2] We will start by a quick over-view of the principles of the CTT approach. We will then recall the rules of intuitionistic predicate logic in CTT.

The general philosophical idea is linked to what has been called the full interpreted[3] approach in which special care is taken, in Martin-Löf's own words (1984, p.2), to "avoid keeping content and form apart. Instead we will at the same time display certain forms of judgement and inference that are used in mathematical proofs and explain them semantically. Thus, we make explicit what is usually implicitly taken for granted". The *explicitation* task involves bringing into the object language level features that determine meaning and that are usually formulated at the metalevel.

According to the CTT view, premises and conclusion of a logical inference are not propositions but judgments.
A rule of inference is justified by explaining the conclusion on the assumption that the premises are known. Hence, before a rule of inference can be justified, it must be explained what it is that we must know in order to have the right to make a judgment of any one of the various forms that the premises and conclusion can have (Martin-Löf 1984, p.2). Two further basic tenets of CTT are the following:

1. No entity without type
2. No type without identity

Accordingly, we can take the assertion that an individual is an element of the set *A* as the assertion that that individual instantiates or exemplifies type *A*. But what is a type, and how do we differentiate between its examples and those objects that are not of a given type? Or, more fundamentally, what must we know in order to have the right to judge something as a type?
In CTT, a set is defined by specifying its *canonical elements*, the elements that "directly" exemplify the type, and its non-canonical ones, the elements that can be shown, using some

---

1 Translated from the Spanish by the Hanna Karpenko from the chapter on an overview of CTT in the book *Las Raíces Dialógicas de la Teoría Constructiva de Tipos, by* Rahman/Clerbout, London : College Publications, in preparation.
2 Cf. Nordström et al. (1990) and Granström (2011).
3 For a thorough discussion on this issue, see Sundholm (1997, 2001).

prescribed method of transformation, that they are equal (in the type) to a canonical one. Moreover, it is required that the equality between objects of a type be an equivalence relation.[4] This is what the second tenet is about: the introduction of an equivalence relation in a set (an object of the type '*set*').
When we have a type we know, from the meaning explanation of what it means to be a type, what the conditions are for being an object of that type. So if *A* is a type and we have an object *b* that satisfies the corresponding conditions, then *b* is an object of type *A*, which we formally write $b : A$.[5] In such a frame

$b : A$	*A true*

can be read as

b is an element of the set *A*	*A* has an element
b is a proof of the proposition *A*	*A* is true
b fulfills the expectation *A*	*A* is fulfilled
b is a solution to the problem *A*	*A* has a solution

'*Set*' itself does not instantiate the type *set*, since we do not have a general method for generating all possible ways of building a set. However, given the type '*set*' we can build the objects that instantiate it by the means described above.

Hypotheticals: The judgements we have introduced so far do not depend on any assumptions. They are *categorical* judgements. The CTT language has also *hypothetical* judgements of the form:

$$B \; type \; (x : A)$$

Where *A* is a type which does not depend on any assumptions and *B* is a type when $x : A$ (the *hypothesis* for *B*). In the case of sets we have that *b* is an element of the set *B*, under the assumption that *x* is an element of the *A*:

$b : B \; (x : A)$ (more precisely: $b : el \; (B) \; (x : el \; (A))$)

The explicit introduction of hypotheticals carries with it the explicit introduction of appropriate substitution rules, but we will deal with them further on in our text, when we present the CTT view on equality.

As pointed out by Granström (2011, p. 112) the form of assertion $b : B \; (x : A)$ ($b : el \; (B) \; (x : el \; (A))$) can be generalized in three directions:

1. Any number of assumptions will be allowed, not just one;
2. The set over which a variable ranges may depend on previously introduced variables;
3. The set *B* may depend on all introduced variables

4 For a thorough discussion see Granström (2011, pp. 54–76).
5 Martin-Löf used the sign "∈" in order to indicate that something, say *a*, is of some type, say *B*. He even suggests to understand it as a the copula 'is'. Nordström et al. (1990) also uses this notation while other authors such as Ranta (1994) use the colon. Granström (2011) distinguishes the colon from the epsilon, where the first applies to non-canonical elements and the latter to canonical ones. We will use the colon.

Such a list of assumptions will be called a *context*. Thus we might need the forms of assertion

$b : B\ (\Gamma)$ – where Γ is a context (i.e., a list of assumptions)

Γ : *context*

In general, a hypothetical judgement is made in the context of the form:

$$x_1 : A_1, x_2 : A_2, \ldots x_n : A_n$$

where we already know that A_1 *is a type,* A_2 *is a type in the context* $x_1 : A_1$*, ..., and* A_n *is a type in the context* $x_1 : A_1, x_2 : A_2, \ldots x_{n-1} : A_{n-1}$:

A_1 *type* [depending on no assumption]
A_2 *type* $(x_1 : A_1)$
...
A_n *type* $(x_1 : A_1, x_2 : A_2, \ldots x_{n-1} : A_{n-1})$
A *type* $(x_1 : A_1, x_2 : A_2, \ldots x_n : A_n)$

$x : A\ (x_1 : A_1, x_2 : A_2, \ldots x_n : A_n)$

The rules for substitution and equality are generalized accordingly:

Hypothetical judgements introduce functions from A to B:

$$f(x) : B\ (x : A)$$

It can be read in several ways, for example:

$f(x) : B$ *for arbitrary* $x : A$
$f(x) : B$ *under the hypothesis* $x : A$
$f(x) : B$ *provided* $x : A$
$f(x) : B$ *given* $x : A$
$f(x) : B$ *if* $x : A$
$f(x) : B$ *in the context* $x : A$

It is crucial to notice that the notion of function is intensional rather than extensional. Indeed, the meaning of an hypothetical function that introduces a function is that whatever element a is substituted for x in $f(x)$, an element $f(a)$ of B results. Moreover, the equality of two functions defined by establishing that substitutions of equal elements of A result in equal elements of B as regulated by the rules of substitution given above – where $b(x)$ is interpreted as function from A to B.[6]

In addition to domains of individuals, an interpreted scientific language requires propositions. They are introduced in CTT by laying down what counts as proof of a proposition. Accordingly, a proposition is true if there is such a proof. We write

$$A : prop$$

to formalize the judgement that A is a proposition. Propositional functions are introduced by hypothetical judgements. The hypothetical judgement required to introduce propositional functions is of the form:

---

6 Cf. Ranta (1994, p. 21), Nordström/Petersson/Smith (1990, chapter 3.3), Primiero (2008, pp. 47-55) and Granström (2011, pp. 77-102).

$$B(x) : prop\ (x : A)$$

that reads, *B*(*x*) *is of the type proposition, provided it is applied to elements of the* (*type-*)*set A.* The rule by which we produce propositions from propositional functions is the following:

$$\frac{a : A \qquad B(x) : prop\ (x : A)}{Ba : prop}$$

And it requires also of the formulation of an appropriate rule that defines the equivalence relation within the type prop:

$$\frac{a=b : A \qquad B(x) : prop\ (x : A)}{Ba=Bb : prop}$$

The notion of propositional function as hypothetical judgement allows the (intensional) introduction of subsets by separation:

$$\frac{A : set \qquad B(x) : prop\ (x : A)}{\{x : A \mid B(x)\} : set} \qquad \frac{b : A \quad B(b)\ \ true}{b : \{x : A \mid B(x)\}}$$

This explanation of subsets also justifies the following rules:

$$\frac{b : \{x : A \mid B(x)\}}{b : A} \qquad \frac{b : \{x : A \mid B(x)\}}{B(b)\ true}$$

Since this method is based on pre-existent sets that have been constructed by description of their canonical elements, the standard paradoxes of set theory do not appear (such paradoxes also appear in some early formulations of the Lorenzen's method for the construction of sets).[7]

What *is given in a context*, the *given contextually-dependent knowledge*, is whatever can be derived from the hypotheses constituting the context. Actually, it is usually distinguished between what is *actually given* in the context (*actual knowledge*), namely the variables themselves and the judgements involving these variables and what is potentially given (*potential knowledge*), namely what can be derived by the rules of type theory from what is actually given. Now, actual and potential knowledge can be increased by extending a given context in ways to be described below.

1.4 The fundaments of the intuitionistic first-order logic in the frame of CTT

7 Cf. Siegwart (1993).

In this section we are going to briefly introduce the basic rules of intuitionistic logic in the CTT frame. Our presentation is inspired by the brief description given in Ranta (1991, section 3) of the system of rules introduced by Martin-Löf (1984, pp. 24-25).

1.4.1 Four types of rules

Because of the fact that in this frame propositions are considered as sets, logical operators are defined as set-theoretical ones. Meaning of these operators is established through four types of different rules, namely: formation rules, introduction rules, elimination rules and equality rules :

- *Formation rules*. The explicit inclusion of the formation rules in the inferential system is one of the most distinctive properties of the CTT. Formation rules simultaneously establish the syntax and the basic types, to which logical and non-logical constants of the language in question correspond. More precisely, the formation rules specify under which conditions can we infer that something is a set (a type), and under which conditions can we affirm that two sets are equal. Therefore, since well-formedness includes not only modes of syntactic composition, but, equally, identification of basic corresponding types, we could say that formation rules display at the same time rules of syntactic well-formedness and the meaning specific to a language in question.
  In fact, any CTT demonstration[8] begins by verifying that expression of judgement, which needs to be proved, results from the application of corresponding formation rules. This is a way to implement inside the CTT frame the idea of fully interpreted language: when we read CTT demonstration bottom-up, it displays syntactic and semantic elements of the demonstrated judgement.
- *Introduction Rules.* Introduction rules define the types of the system, establishing the way to form canonical elements and the way to determine if two canonical elements are equal.
- *Elimination rules.* They establish how to define functions (called selectors) on the sets defined by the rules of introduction.
- *Equality rules*. As we mentioned in the previous section, equality rules are linked to the "harmony" between introduction and elimination rules. More precisely, equality rules specify the way in which selectors, defined by elimination rules, work and how to execute their computation in relation to the canonical elements, generated by the introduction rules.

8 We remind to the reader that we use the expression *CTT demonstration* as an abbreviation of the expression *a demonstration in the frame of the constructive type theory*.

1.4.2 The intuitionistic logic in CTT

The Π-operator
(cartesian product of a family of set):

$$\frac{A : set \qquad \begin{array}{c}(x : A)\\ \dots \\ B(x) : set\end{array}}{(\Pi x : A)B(x) : set}\ \Pi F \qquad\qquad \frac{\begin{array}{c}(x : A)\\ \dots \\ b(x) : B(x)\end{array}}{(\lambda x)b(x) : (\Pi x : A)B(x)}\ \Pi I$$

$$\frac{c : (\Pi x : A)B(x) \qquad a : A}{Ap(c, a) : B(a)}\ \Pi E$$

$$\frac{a : A \qquad \begin{array}{c}(x : A)\\ \dots \\ b(x) : B(x)\end{array}}{Ap((\lambda x)b(x), a) = b(a) : B(a)}\ \Pi Eq^1$$

$$\frac{c : (\Pi x : A)B(x)}{c = (\lambda x)Ap(c, x) : (\Pi x : A)B(x)}\ \Pi Eq^2$$

The introduction rule assumes, as usual, that there is no other occurrence of x in any assumption besides the ones of the form $(x : A)$. The binary function $Ap(x, y)$ (the selector *Application*) is defined by both, the way it is introduced (by the elimination rule) and by the specification of its computation (using equality rules). One can read it as "application of x to y". In fact, it is a method to obtain a canonical object $B(a)$, provided $a : A$ (see Martin-Löf (1984, pp.28-29).

In other words, the computation rule for $Ap((\lambda x)b(x), a)$ establishes that if $b(x)$ is a proof-object for $B(x)$ under the assumption that $x : A$, and given that a is such a proof of A, then its execution yields a proof object $b(a) : B(a)$, such that a replaces x throughout in $b(x)$. For short:

$$Ap((\lambda x)b(x), a) \rightarrow b(a/x)$$

The Π-operator allows to define the universal quantifier and the material implication in the following way:

- $(\forall x : A)B(x) = (\Pi x : A)B(x) : prop$, provided $A : set$ and $B(x) : prop\ (x : A)$.
- $A \rightarrow B = (\Pi x : A)B : prop$, provided $A : prop$ and $B : prop$.

The Σ operator
(disjoint union of a family of sets)

$$\begin{array}{ll} & (x : A) \\ & \dots \\ A : set \quad B(x) : set & \\ \hline (\Sigma x : A)B(x) : set \ \Sigma F & \end{array} \qquad \begin{array}{ll} & \\ & \\ a : A \quad b : B(a) & \\ \hline (a, b) : (\Sigma x : A)B(x) \ \Sigma I & \end{array}$$

$$\begin{array}{ll} & (x : A, y : B(x)) \\ & \dots \\ c : (\Sigma x : A)B(x) \quad (x, y) : C((x, y)) & \\ \hline \mathbf{E}(c, (x, y)d(x, y)) : C(c) \ \Sigma E & \end{array}$$

$$\begin{array}{lll} & (x : A, y : B(x)) & \\ & \dots & \\ a : A \quad b : B(a) \quad d(x, y) : C((x, y)) & & \\ \hline \mathbf{E}(a, b, (x, y)d(x, y)) = d(a, b) : C((a, b)) \ \Sigma Eq & & \end{array}$$

The expression $\mathbf{E}(c, (x, y)d(x, y))$ which occurs as the conclusion of Σ-elimination rule, is informally read as the following computational instruction:

> Execute c.
> The result of the execution is a canonical element which has the form of the couple (a, b) such that $a : A$ and $b : B$.
> Now substitute a and b in the right premise, for x and y respectively.
> Thus obtain: $d(a, b) : C((a, b))$.
> The execution of $d(a, b)$ will give for result a canonical element e of $C((a, b))$ – it is not difficult to deduce, therefore, that e is a canonical element of $C(c)$ (see Martin-Löf (1984, p. 40)).

The Σ-operator allows to define the existential quantifier and the conjunction in the following way:

- $(\exists x : A)B(x) = (\Sigma x : A)B(x) : prop$, provided $A : set$ and $B(x) : prop$ $(x : A)$.
- $A \wedge B = (\Sigma x : A)B : prop$, provided $A : prop$ and $B : prop$.

In the case of conjunction, we obtain the standard elimination rules from the elimination rules of Σ,

1. if we decide that C is either A or B, and
2. if we define the projection rules $p(c)$ and $q(c)$, mentioned in the previous section, in the following way: $p(c) \equiv \mathbf{E}(c,(x,y)x)$ and $q(c) \equiv \mathbf{E}(c,(x,y)y)$.

That is, if we carry out steps 1 and 2, from ΣE we get:

$$c : A \wedge B \qquad\qquad c : A \wedge B$$

$$\frac{}{p(c) : A}\ \wedge E1 \qquad \frac{}{q(c) : B}\ \wedge E2$$

Recalling the equality rules, we come to the following computational rules for the execution of $p(c)$ and $q(c)$, where c is constituted by the pair (a, b) such that $a : A$, $b : B$:

$$p(a, b) \rightarrow a \qquad q(a, b) \rightarrow b$$

The operator +
(the disjoint union or the co-product of two sets):

$$\frac{A : set \quad B : set}{A + B : set}\ +F \qquad \frac{a : A}{i(a) : A + B}\ +I1$$

$$\frac{b : B}{j(b) : A + B}\ +I2$$

$$\frac{c : A + B \qquad \begin{matrix}(x : A)\\ \dots\\ d(x) : C(i(x))\end{matrix} \qquad \begin{matrix}(y : B)\\ \dots\\ e(y) : C(j(y))\end{matrix}}{\mathbf{D}(c, (x)d(x), (y)e(y)) : C(c)}\ +E$$

$$\frac{a : A \qquad \begin{matrix}(x : A)\\ \dots\\ d(x) : C(i(x))\end{matrix} \qquad \begin{matrix}(y : B)\\ \dots\\ e(y) : C(j(y))\end{matrix}}{\mathbf{D}(i(a), (x)d(x), (y)e(y)) = d(a) : C(i(a))}\ +Eq^1$$

$$\frac{b : B \qquad \begin{matrix}(x : A)\\ \dots\\ d(x) : C(i(x))\end{matrix} \qquad \begin{matrix}(y : B)\\ \dots\\ e(y) : C(j(y))\end{matrix}}{\mathbf{D}(j(b), (x)d(x), (y)e(y)) = e(b) : C(j(b))}\ +Eq^2$$

The expressions i and j are two new primitive constants (injections), that provide the information that an element of $A+B$ belongs to A or to B, and which of both is actually the case.

The expression $\mathbf{D}(c, (x)d(x), (y)e(y))$ that occurs in the conclusion of the elimination rule is informally read as the following computational instruction:

Execute c;

if as result of the execution the canonical elements *i(a)* results, then substitute *a* for *x* in *d*(*x*);
if as result of the execution the canonical elements *j(b)* results, then substitute *b* for *y* in *e(y)*

If we recall the equality rules, we can also express this in the following way: the execution of **D**(*c*, (*x*)*d*(*x*), (*y*)*e*(*y*)), yields a proof-object for *C*(*c*) built from proof-object *c* for *A* + *B*, and the proof-objects *d*(*x*) and *e*(*y*) for *C*(*i*(*x*)) and ;*C*(*j*(*y*)) respectively, such that *x* : *A* and *y* : *B*. If we display this with two computational rules we obtain:

D(*i*(*a*), (*x*)*d*(*x*), (*y*)*e*(*y*)) → *d*(*a*) (where *d*(*a*) : *C*(*i*(*a*)))
D(*j*(*b*), (*x*)*d*(*x*), (*y*)*e*(*y*)) → *e*(*b*) (where *e*(*b*) : *C*(*j*(*b*)))

The operator + allows to define the disjunction in the following way:

$A \vee B = A + B$: *prop*, provided *A* : *prop* and *B* : *prop*.

The absurdum ⊥

	x : ⊥
--------- ⊥F	-------- ⊥E (for arbitrary *x*)
⊥ : *set*	*a* : *A*

The symbol ⊥ is nothing else than an empty set, and since its formation rule establishes that it is always a set, the same rule also establishes that ⊥ is always a proposition. The absurdum does not require any introduction rule. In fact, the elimination rule for $A \rightarrow \perp$ introduces ⊥. Clearly, it follows from the standard perspective of intuitionistic logic in which ~*A* is seen as an abbreviation for $A \rightarrow \perp$. Since there is no specific rule for ⊥ ,there is no specific equality rule which would establish the relationship between the introduction and the elimination rules. Finally, the elimination rule expresses the so-called rule of *ex falso sequitur quodlibet.* In fact, a judgement that an *x* is of type ⊥ is contradictory because ⊥ is an empty set. So, from such a judgement we could conclude *a* : *A* (see Martin-Löf (1984, pp.65-67)).

Because of the fact that one of the main theses of our work is based on the idea that the copy-cat move in dialogical logic is closely linked to the dynamic version of the notion of definitional equality as deployed by the Π- and Σ-equality rules, we should present systematically CTT-take on definitional and propositional equality.

Due to space limitations, we will not be able to provide a detailed description of the CTT here. In the next section we will limit ourselves by exhibiting the issues, relevant for the objectives of the present work . For more information on this perspective, the reader should examine Martin-Löf (1984), Ranta (1988, 1994), Nordström et al. (1990), Primiero (2008) and Granström (2011).

II.2 Assertions and equality

Whenever a new expression is introduced in CTT, this introduction is deployed by means of what is called a *meaning explanation*. In the case of the introduction of a new type, the meaning explanation consists in (1) describing its canonical elements, (2) providing an algorithm for deciding whether a non-canonical object is of the given type or not, and (3) give the conditions that allow establishing (or not) the identity of two objects with respect to the

given type. The point (3) consists in defining an appropriate equivalence relation. In this way, assertions of the form $a = b : A$, affirm that two objects a and b satisfy the equivalence relation defined for a type A. The assertion $a = b : A$ is also called an *assertion of definitional equality*, since by means of this assertion explicit definitions are This kind of equality is transmitted by reflexivity, symmetry, transitivity, and by substitution of definitionally equals. (Ranta 1994, 52). Even more, real definitions as pointed out by Frege while criticizing Hilbert, might provide the truth of a proposition based on those definitions, but they are *not* bearers of truth. For short, definitional equality is *not* a predicate. The written form $a = b$ is ambiguous unless we make fully explicit the assertion in which it is embedded: at the left of the colon it expresses definitional equality and the right of it expresses a predicate.

When the type is a proposition constituted by logical constants such as conjunction, disjunction and so on, the inferential role of the equalities at the left of the colon is to harmonize the process of synthesis and analysis of an assertion involving the logical constant at stake. Let us briefly discuss this point in the following section – for a thorough discussion see Rahman/Redmond (2015b).

II.2.1 Proof-objects, equality and their inferential role

In his landmark 1935 article on natural deduction, Gerhard Gentzen points out:

> *The introductions represent, as it were; the 'definitions' of the symbols concerned, and the eliminations are no more, in the final analysis; than the consequence of these definitions.* (Gentzen 1969, 80)

The idea behind Gentzen's observation is that an introduction rule in the system of natural deduction exhibits the basis for the assertion of a proposition which contains the logical constant in question, and that the result of using the correspondent elimination rule shows exactly those elements of proposition, that introduction rule characterized as sufficient for its assertion. Dummett calls such form of coordination *harmony*. Briefly, an elimination rule is harmonic if this rule does no more and no less than displaying the consequences of the meaning provided by the introduction rule.[9] If we begin with the introduction rules, one way to see the contribution of the concept of harmony is that while the introduction rules show how to *synthesize*, from certain premises, an expression that contains a given logical constant, the elimination rule shows how to *analyze* the expressions under consideration into exactly those components required by the introduction rule.
The explicit language of the CTT allows to express the condition of harmony by means of definitional equalities. Let us take as an example the case of the conjunction. The proposition $A \wedge B$ (or the set $A \times B$) is explained by establishing that the canonical element of $A \wedge B$ is a couple of proof-objects (a, b) where $a : A$ and $b : B$ – that is to say, where a is a proof-object of A and b is a proof-object of B:

$$\frac{a : A \quad b : B}{(a, b) : A \wedge B} \wedge\text{-introduction}$$

In order to formulate $\wedge$-elimination-ruel we will use certain operators, called *selectors*, by the means of which new functions could be defined that extract those elements that constitute a complex proof-object c (such as $c = (a, b)$). In the case of conjunction, the selectors are the projection functions ρ and q that yield, respectively, the left and the right side of the couple of

[9] Cf. Stephen Read (2008, 291, note 5).

proof-objects. Therefore, if c is a proof-object for a conjunction, then $p(c)$ yields the left component of c and $q(c)$ yields it's right component.

$$\frac{c : A \wedge B}{p(c) : A}\ \wedge\text{-}p\text{-elimination} \qquad \frac{c : A \wedge B}{q(c) : B}\ \wedge\text{-}q\text{-elimination}$$

If we know that $c = (a, b)$, then $p(c)$ will restore the right component of c (obtained by the introduction rule). In other words, $p(c) = p((a, b)) = a$, such that $a : A$. Analogously, $q(c)$ restores the right component:

$$\frac{a : A \quad b : B}{((a, b)) = a : A}\ \wedge\textit{-left equality} \qquad \frac{a : A \quad b : B}{q((a, b)) = b : B}\ \wedge\textit{– right-equality}$$

These are clear examples of how to use the notion of definitional equality, previously mentioned: projection functions p and q are explicitly defined by the inference rule in the way that, given proof-objects a and b, the projection q of (a, b) is definitionally equal to b, according to the equivalence relation which is defined within the type B. The projection p is introduced analogously. These equalities can be seen as displaying the execution of the computation rules mentioned above for the projections p and q in relation to the pair (x, y), such that $x : A, y : B$

On can also introduce a dual rule, called η, which could be defined in the following way:

$$\frac{c : A \wedge B}{(p(c), q(c)) = c : A \wedge B}\ \wedge\text{–}\eta\text{-equality}$$

In the present context these rules could be seen as ensuring that the rules are harmonic. That is to say, if the proof-object c for the conjunction is constituted by the pair (a, b), then both elimination rules provide an analysis of the proof-object c from which the components a and b result.

II.2.2 Definitional equality and propositional equality

II.2.2.1 The definitional equality

The definitional equality meets the basic conditions of equality:

Reflexivity	Symmetr	Transitivity
$\dfrac{a : A}{a = a : A}$	$\dfrac{a = b : A}{b = a : A}$	$\dfrac{a = b : A \qquad b = c : A}{a = c : A}$
$\dfrac{A : set}{A = A : set}$	$\dfrac{A = B : set}{B = A : set}$	$\dfrac{A = B : set \qquad B = C : set}{A = C : set}$

Extensionality

$$\frac{A = B : set \quad a : A}{a : B} \qquad \frac{A = B : set \quad a = b : A}{a = b : B}$$

In the context of hypothetical judgements the definitional equality between two proof-objects, for example *a* and *c*, in the given set, say *A*, produces the equality between functions *b*(*a*) and *b*(*c*), which is inferred from the substitution of the variable in the function *b*(*x*) from *A* to *B*(*x*) by *a* and *c* respectively:[10]

Substitution

$$\frac{a = c : A \quad b(x) : B(x)\ (x : A)}{b(a) = b(c) : B(a)} \qquad \frac{a = c : A \quad B(x) : set\ (x : A)}{B(a) = B(c) : set}$$

Hypothetical judgements involving hypothetical equalities yield the following rule:

Substitution involving definitional equalities between dependent objects

$$\frac{a : A \quad c(x) = b(x) : B(x)\ (x : A)}{b(a) = c(a) : B(a)}$$

Substitution rules for equalities in the context of hypothetical judgements require equally the general notion of substitution of variables expressed by the following rule:

General substitution rule for hypotheticals

$$\frac{a : A \quad b(x) : B(x)\ (x : A)}{b(a) : B(a)}$$ [11]

Let us now see how in the frame of CTT the notion of definitional equality and propositional equality relate.

II.2.2.2 Propositional equalities: Predicates of intensional and extensional equality

10 We present here the rules of substitution for equalities between the elements of a set, but it is equally possible to define analogously equalities between elements of the type *set*, that is equalities between sets. (see Nordström/Petersson/Smith (1990), p. 38).

11 Actually, this rule does not suffice, because it does not include the case of simultaneous substitutions, that are necessary, for example, in cases as *C*(*x*, *y*) : *set* (*x* : *A*, y : *B*(*x*)). There are different solutions to this respect, but the most practical is that from a procedural point of view (so dear to dialogical perspective), the one that allows partial substitutions in a context. Returning to our example, if it is the case that *a* : *A*, *we conduct a first application of the substitution to obtain C*(*a*, *y*) : *set* (y : *B*(*a*)), and then a second one to obtain *b* : *B*(*a*), *C*(*a*, *b*) : *set* (see Nordström/Petersson/Smith (1990), p. 39).

Now we need to introduce the equality predicate that allows both, expressing equality as a proposition and showing how definitional and propositional equality relate. Actually, it is possible to define two equality predicates, one intensional and one extensional. The proof-object of the latter is independent of the proof-objects of the premises.

- **The predicate of intensional equality**

The predicate of propositional intensional equality, defined over a set, say *A*, is usually expressed with the notation ***Id***(*A*, *a*, *b*). Sometimes the notation $a =_A b$ is also used. We will use the first notation, since it shows explicitly that ***Id*** is a relation.

Formation

$$\frac{A : set \quad a : A \quad b : A}{\textbf{\textit{Id}}(A, a, b) : set}$$

As pointed out by Thompson (1999, p. 110) this formation rule is quite different from the type-formation rules we have seen so far. The latter take the form

$$\frac{\ldots : type \qquad \ldots : type}{\ldots : type}$$

which means that with these rules alone, we can say what are the types of the system independently of which elements inhabit those types. The rule of ***Id***- formation above breaks this rule, since it requires that $a : A$ and $b : A$ for ***Id***(*A*, *a*, *b*) to be a type.. This means that the rules generating those types are inextricably bound to inference rules that involve specific proof-objects inhabiting the relevant types.

The basic introduction rule, yields reflexivity and stems from the fact that given the assertion $a : A$, we have then immediately $a = a : A$. From the inferential point of view, this amounts to introducing the reflexive case for ***Id***(*A*, *a*, *a*).

$$\frac{a : A}{r(a) : \textbf{\textit{Id}}(A, a, a)}$$

The proof-object for ***Id***(*A*, *a*, *a*) is *r*(*a*). The inner-structure of this proof-object is nothing more and nothing else than it's dependence upon *a*: what makes the prop ***Id***(*A*, *a*, *a*), true is an operator, namely *r*, such that applied to *a*, yields *a*.

By using Substitution in sets on $a = b : A$, ***Id***(*A*, *a*, *x*) : *prop* ($x : A$) and ***Id***-introduction we obtain ***Id***(*A*, *a*, *a*) = ***Id***(*A*, *a*, *b*) : *prop*.

$$\frac{\dfrac{A : set \quad a : A \qquad x : A^1}{\textbf{\textit{Id}}(A, a, x) : prop}\ \textbf{\textit{Id}}\text{-}\textbf{\textit{F}} \qquad a = b : A}{a : A}\ \text{subst}^1 \qquad \frac{a : A}{}\ \textit{Id}\text{-introduction}$$

$$\frac{\boldsymbol{Id}(A, a, a) = \boldsymbol{Id}(A, a, b) : prop \qquad r(a) : (A, a, a)}{r(a) : \boldsymbol{Id}\,(A, a, b)}\ \text{extensionality}$$

Hence, by set-equality and the introduction rule for reflexive ***Id***, we obtain also the introduction[13] of the non-reflexive case:

$$\frac{a : A \qquad a = b : A}{r(a) : \boldsymbol{Id}(A, a, b)}\ \boldsymbol{Id}\text{-introduction}$$

Elimination and Substitution

In relation to the elimination rule, it is related to a generalization of Leibniz's substitution rule, without making use of second-order quantification. The main concept behind Leibniz's substitution rule is that equals with respect to a given predicative identity can be substituted for equals with respect to that identity. Suppose that we have some proof *p* of a proposition *C* involving *a*, and also that we know that $c : \boldsymbol{Id}(A, a, b)$. We should be able to infer the proposition *C'* resulting from replacing (at least) *some* of the occurrences of *a* in *C* by *b*. To capture the idea of substituting *some* of the occurrences we think of *C* of having the form $C[a/x, a/y]$ in which *a* replaces two free variables *x* and *y*. In fact, more generally, we will also think of the form of *C* as including cases where also $r(a)$ occurs there. The point is that every proposition *P* on which a substitution w.r.t. *Id* is to be carried out presupposes that its formation includes $r(x)$ in such a way that *P* can be brought to the form $C[a/x, a/y, r(a/x)]$. This yields the following general rule

$$\frac{\begin{array}{l} a : A \\ b : A \\ c : \boldsymbol{Id}(A, a, b) \\ C[x, y, z] : set\ (x : A, y : A, z : \boldsymbol{Id}(A, a, b)) \\ d(x) : C[x, x, r(x)] \quad (x : A) \end{array}}{id(c, d) : C[a, b, c]}\ Id\text{-elimination}$$

What the operator in the expression $id(c, d)$ amounts to is that of applying the identity $c : \boldsymbol{Id}(A, a, b)$ to *d*, in our case $d(a)$, and this will return the proof-object *d(a)*. This already suggests the rules that determine when two $id(c, d)$ are equal:

$$\frac{\begin{array}{l} a : A \\ C[x, y, z] : set\ (x : A, y : A, z : \boldsymbol{Id}(A, a, b)) \\ d(x) : C[x, x, r(x)] \quad (x : A) \end{array}}{id(r(a), d) = d(a) : C[a, a, r(a)]}\ \text{equal elements}$$

Let us see now how the elimination rule works with some examples. We start showing that Leibniz's substitution rule can be inferred from the elimination rule. As already mentioned,

12 Cf. Ranta (1994), p. 53.

13 Cf. Nordström/Petersson/Smith (1990), p. 57.

the idea of the rule is that given $B(a)$ and $\boldsymbol{Id}(A, a, b)$), we should be able to conclude $B(b)$ by replacing a with b in $B(a)$.

Leibniz's substitution law

Assume that, $a, b : A$, and $c : \boldsymbol{Id}(A, a, b)$, and also $p : B(a)$.

The first task to accomplish it infer from $B(x)$ some expression of the form $C[x, x, r(x)]$. From $B(x)$ we can infer the suitable expression $B(x) \rightarrow B(x)$, with the identity function $\lambda x.x$ as proof-object. Thus, given $\lambda x.x : B(x) \rightarrow B(x)$ (here C does not mention $r(a)$) by $\boldsymbol{Id}$-elimination we obtain $id(c, d(x) : B(a) \rightarrow B(b)$, that is : $id(c, (x)\lambda x.x): B(a) \rightarrow B(b)$ and making use of the premise $p : B(a)$ and applying implication-elimination we come to $subst\ (c, p) : B(b)$. Where $subst(c, p)$ is an abbreviation for $ap(id(c, (x)\lambda x.x), p)$.

$$\dfrac{\dfrac{a : A \qquad b : A \quad c : \boldsymbol{Id}(A, a, b) \qquad \dfrac{x : B(x)}{\lambda x.x : B(x) \rightarrow B(x)\ (x : A)}\ \rightarrow I}{id(c, (x)\lambda x.x): B(a) \rightarrow B(b)}\ Id\text{-elimination}}{}$$

Making use of the premise $p : B(a)$ and applying implication-elimination we come to $subst(c, p) : B(b)$. Where $subst(c, p)$ is an abbreviation for $ap(id(c, (x)\lambda x.x), p)$.

$$\dfrac{\dfrac{\dfrac{a : A \qquad b : A \qquad c : \boldsymbol{Id}(A, a, b) \qquad \dfrac{x : B(x)}{\lambda x.x : B(x) \rightarrow B(x)\ (x : A)}\ \Pi I}{id(c, (x)\lambda x.x): B(a) \rightarrow B(b)} Id\text{-elimination} \qquad p : B(a)}{ap(id(c, (x)\lambda x.x), p) : B(b)}\ \Pi E$$

This leads to the following formulation of Leibniz's rule:

$$\dfrac{B(x) : set\ (x : A) \qquad a : A \quad b : A \quad c : \boldsymbol{Id}(A, a, b) \quad p : B(a)}{subst(c, p) : B(b)}$$

Id is also transmitted by reflexivity, transitivity and symmetry. Since the introduction rule yields reflexivity, we have to show that from the elimination rule we can obtain symmetry and transitivity

Symmetry: Assume that, $a, b : A$, and $d : \boldsymbol{Id}(A, a, b)$. Let us consider C to be e: $\boldsymbol{Id}(A, y, x)$. From a: A, we obtain by introduction $r(a) : \boldsymbol{Id}(A, a, a)$ and this yields by $\boldsymbol{Id}$-elimination $id(d, r(a)) : \boldsymbol{Id}(A, b, a)$. This allows writting down the following rule for symmetry:

$$\dfrac{d : \boldsymbol{Id}(A, a, b)}{symm(d) : \boldsymbol{Id}(A, b, a)}$$

where *symm*(*d*) stands for *id*(*d*, *r*(*a*)).

Transitivity: The main step in order to infer the transitivity of ***Id*** is to Consider *C* to be λ*y*.*y* : : ***Id***(*A*, *y*, *c*) →: ***Id***(*A*, *x*, *c*), once we assumed *d* : ***Id***(*A*, *a*, *b*) and *e* : ***Id***(*A*, *b*, *c*). The use of ***Id***-elimination yields *id*(*d*, (x)λ*y*.*y*) : ***Id***(*A*, *b*, *c*) →: ***Id***(*A*, *a*, *c*). By Π-Elimination – making use of *e* : ***Id***(*A*, *b*, *c*). – we obtain *trans*(*d*, *e*) : ***Id***(*A*, *a*, *c*), where *trans*(*d*, *e*) stands for *ap*(*id*(*d*, (x)λ*y*.*y*), *e*). This leads us to the following rule:

$$\frac{d : \mathbf{Id}(A, a, b) \quad e : \mathbf{Id}(A, b, c)}{trans(d, e) : \mathbf{Id}(A, a, c)}$$

- **The predicate of the extensional equality**

The formation rule of the predicate of extensional equality between *a* and *b,* defined in the set *A* (***Eq***(*A*, *a*, *b*)), is analogous to the predicate of intensional identity:

$$\frac{A : set \quad a : A \quad b : A}{\mathbf{Eq}(A, a, b) : set}\ \mathbf{Eq}\ \mathrm{F}$$

Introduction of *Eq*

The proof-object ***eq***, that verifies ***Eq***(*A*, *a*, *b*)), does not depend on either *a* or *b*. The veification of ***Eq***(*A*, *a*, *b*) only needs an identity of the arbitrary proof-object. Take *eq* to be, for example *s* = *s*.

$$\frac{a = b : A}{eq : \mathbf{Eq}(A, a, b)}\ \mathbf{Eq}\mathrm{I}$$

This rule represents the weakest way in which we could relate ontological equality with a propositional one. In fact, the proof-object *eq* neither contains, nor depends on any of the proof-objects upon which the assertion of the premise is based. This absence of links between the proof-objects of the premise and the conclusion is even more evident in the following elimination rules, defined non-inductively:

Elimination of *Eq*

$$\frac{c : \mathbf{Eq}(A, a, b)}{a = b : A}\ \mathbf{Eq}\mathrm{E}^{1}$$

In this elimination rule the proof-object *c* of the premise does not contain anything that could lead to the equality *a* = *b* of the conclusion. What is more, since equality expressed by ***Eq***(*A*, *a*, *b*) is numerical identity, any proof-object *c* of ***Eq***(*A*, *a*, *b*) is equal to *eq*:[14]

14 Véase Nordström /Petersson/Smith [1990], p. 61.

$$\frac{c : \boldsymbol{Eq}(A, a, b)}{c = eq : \boldsymbol{Eq}(A, a, b)}\ \boldsymbol{Eq}\mathrm{E}^2$$

It could be shown that with both elimination rules we obtain a substitution rule analogous to the one defined for intensional equality, and, therefore also obtain analogous rules of transmission.[15]

$$\frac{\begin{array}{l} a : A \\ b : A \\ c : \boldsymbol{Eq}(A, a, b) \\ C[x, y, z] : set\ (x : A, y : A, z : \boldsymbol{Eq}(A, a, b)) \\ d(x) : C[x, x, eq]\ \ (x : A) \end{array}}{d(a) : C[a, b, c]}\ \boldsymbol{Eq}\mathrm{E}^3$$

15 Véase Nordström /Petersson/Smith [1990], p. 61.