AUTOMATH and pure type systems

Eindhoven University of Technology
Department of Mathematics and Computing Science

AUTOMATH and Pure Type Systems

by

Twan Laan

96/11

ISSN 0926-4515

editors: prof.dr. R.C. Backhouse
prof.dr. J.C.M. Baeten

Reports are available at:
http://www.win.tue.nl/win/cs

Computing Science Report 96/11
Eindhoven, May 1996

AUTOMATH and Pure Type Systems

Twan Laan
Co-operation Centre Tilburg and Eindhoven Universities
Eindhoven University of Technology
P.O.Box 513, 5600 MB EINDHOVEN, THE NETHERLANDS
e-mail laan@win.tue.nl

April 26, 1996

Abstract

We study the position of AUTOMATH systems within the framework of the Pure Type Systems as discussed in [3].

Contents

One should not define
the word "mathematics"
by a list of traditional subjects,
but by the mathematical method

N.G. de Bruijn.

1 Introduction

The AUTOMATH project was started in 1967 at Eindhoven University of Technology, by N.G. de Bruijn. Though AUTOMATH heavily depends on logic and type theory, the reasons for its development are not to be found in these subjects, but in mathematics. Already for some years, de Bruijn had been wondering what a proof of a theorem in mathematics should be like, and how the correctness of a proof should be checked. The development of computers in the 60s made him wonder whether a machine could check the proof of a mathematical theorem, provided the proof was written in a very accurate way. De Bruijn developed the language AUTOMATH for this purpose. This language is not only (according to de Bruijn in [7]) *"a language which we claim to be suitable for expressing very large parts of mathematics, in such a way that the correctness of the mathematical contents is guaranteed as long as the rules of grammar are obeyed"* but also *"very close to the way mathematicians have always been writing"*.
The appearance of types in AUTOMATH finds its roots in de Bruijn's contacts with Heyting, who made de Bruijn familiar with the intuitionistic interpretation of the logical connectives (see [24], [18]). The interpretation of the proof of an implication $A \to B$ as an algorithm to transform any proof of A in a proof of B, so in fact a function from proofs of A to proofs of B, gave rise to interpret a proposition as a class (a type) of proofs. De Bruijn, who was not influenced by developments in λ-calculus or type theory when he started his work on AUTOMATH, discovered this notion of "proofs as objects", better known as "propositions as types", independently from Curry [12] and Howard [20].

As AUTOMATH was developed quite independently from other developments in the world of type theory and λ-calculus, there are many things to explain in the relation between the various AUTOMATH languages and other type theories.
Type theory was originally invented by Bertrand Russell to exclude the paradoxes that arose from Frege's "Begriffschrift" [14]. It was presented in 1910 in the famous "Principia Mathematica" [31], and simplified by Ramsey [28] and by Hilbert and Ackermann [19]. In 1940, Church combined his theory of functions, the λ-calculus ([9, 10]) with the simplified type theory, resulting in the so-called "Simple Theory of Types" [11]. This system has served as a basis for the many systems that have been developed since then. In 1989, Terlouw [30] presented, as an extension of Barendregt's work [3], a general framework for type systems, which is at the basis of the so-called Pure Type Systems (PTSs; see [16], [3], [15]). The theory of PTSs nowadays plays a central role in type theory and typed λ-calculus.

This paper will focus on the relation of AUTOMATH to PTSs.
Both [3] and [15] mention this relation in a few lines, but as far as we know a satisfactory explanation of the relation between AUTOMATH and PTSs is not available.
Moreover, both works consider AUTOMATH without one of its most important mechanisms: The definition system. Even the system PAL, which roughly consists of the definition system of AUTOMATH only, is able to express some simple mathematical reasoning (see for instance Section 5 of [7]). According to de Bruijn [8] this is *"due to the fact that mathematicians worked with abbreviations all the time already"*.
Also, recent developments on the use of definitions in Pure Type Systems by Bloo, Kamareddine and Nederpelt [6, 21] and Severi and Poll [29] justify renewed research on the relation between AUTOMATH and PTSs.

In Section 2 we give a short overview of Pure Type Systems. In Section 3 we give a description of AUT-68. In Section 4 we discuss how we can transform AUT-68 into a PTS. We must notice that AUT-68 has some properties that are not usual for PTSs:

- AUT-68 has η-reduction;
- AUT-68 has Π-application and Π-reduction;
- AUT-68 has a definition system.

In systems with Π-application, a term $\Pi x{:}A.B$ can be applied to a term N (of type A). This results in $(\Pi x{:}A.B)N$. The usual application rule of Pure Type Systems then changes to

$$\frac{\Gamma \vdash M : \Pi x{:}A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : (\Pi x{:}A.B)N}$$

In such systems, Π behaves like λ, and as a consequence, there is also a rule of Π-reduction

$$(\Pi x{:}A.B)N \to_\Pi B[x{:=}N].$$

In AUTOMATH, one even uses the same notation for the terms $\Pi x{:}A.B$ and $\lambda x{:}A.B$, namely: $[x{:}A]B$, and it is not always easy to see whether a term $[x{:}A]B$ represents $\lambda x{:}A.B$ or $\Pi x{:}A.B$. For more details on Π-application and Π-reduction, see [21], [22] and the literature on AUTOMATH [26].

For reasons of clarity, we only treat the system AUT-68 *without* η-reduction, Π-application and Π-reduction in this paper. In Section 5, we present a system $\lambda 68$ that is (almost) a PTS. We show that it has the usual properties of PTSs and we prove that $\lambda 68$ is to AUT-68 without η-reduction, Π-application and Π-reduction.
In Section 6 we compare the definition system of AUT-68 with several other, more modern, type systems with definitions.

2 Pure Type Systems

Pure Type Systems (PTSs) were introduced (in a somewhat different way than presented below) by Terlouw [30] in 1989 and were also implicitly present in the work of Berardi [5]. Many type systems can be described as a PTS and this makes PTSs a central notion in type theory. Below we repeat the definition of PTS as presented in [3], In [3], one can also find the basic properties of PTSs, and some examples. We assume that we have an infinite set $\mathbb{C}$ of constants, and an infinite set $\mathbb{V}$ of variables.

Definition 2.1 (Pure Type Systems) Let $\mathcal{S} \subseteq \mathbb{C}$ (the set of *sorts*), $\mathcal{A}$ a set of *axioms* of the form $c : s$, where $c \in \mathbb{C}$ and $s \in \mathcal{S}$, and $\mathcal{R}$ a set of *rules* of the form (s_1, s_2, s_3) with $s_1, s_2, s_3 \in \mathcal{S}$. The PTS determined by $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ is induced as follows:

- The set of *terms* $\mathbb{T}$ is defined by

$$\mathbb{T} ::= \mathbb{V} \mid \mathbb{C} \mid \mathbb{T}\mathbb{T} \mid \lambda \mathbb{V}{:}\mathbb{T}.\mathbb{T} \mid \Pi \mathbb{V}{:}\mathbb{T}.\mathbb{T}.$$

- On terms we have the well-known notions of β-reduction (indicated by $\to_\beta$) and β-conversion (indicated by $=_\beta$), defined by the contraction rule

$$(\lambda x{:}A.B)C \to_\beta B[x{:=}C].$$

- A *pseudocontext* is a finite (possibly empty) list of *declarations* $x_1{:}A_1, \ldots, x_n{:}A_n$ with $x_i \in \mathbb{V}$ and $A_i \in \mathbb{T}$ for all i.

- A *statement* has the form $\Gamma \vdash A : B$, where Γ is a pseudocontext and $A, B \in \mathbb{T}$. The following rules determine the valid statements of the PTS specified by $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ (x ranges

over variables; s ranges over sorts):

$$\text{(Axiom)} \quad \vdash c : s \qquad (c{:}s \in \mathcal{A})$$

$$\text{(Start)} \quad \frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A}$$

$$\text{(Weak)} \quad \frac{\Gamma \vdash M : N \qquad \Gamma \vdash A : s}{\Gamma, x{:}A \vdash M : N}$$

$$\text{(Π-form)} \quad \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_3} \qquad ((s_1, s_2, s_3) \in \mathcal{R})$$

$$(\lambda) \quad \frac{\Gamma, x{:}A \vdash F : B \qquad \Gamma \vdash \Pi x{:}A.B : s}{\Gamma \vdash (\lambda x{:}A.F) : (\Pi x{:}A.B)}$$

$$\text{(App)} \quad \frac{\Gamma \vdash M : \Pi x{:}A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x{:=}N]}$$

$$\text{(Conv)} \quad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s \qquad A =_\beta B}{\Gamma \vdash M : B}$$

It is assumed that the newly introduced variables in the Start and Weakening rules do not occur in Γ.
A pseudo-context Γ is called a *context* if there are A, B such that $\Gamma \vdash A : B$.

We further introduce some abbreviations:

Notation 2.2 We write

$$\textstyle\prod_{i=1}^{n} x_i{:}A_i.B$$

as shorthand for $\Pi x_1{:}A_1.\cdots\Pi x_n{:}A_n.B$ and

$$\boldsymbol{\lambda}_{i=1}^{n} x_i{:}A_i.b$$

as shorthand for $\lambda x_1{:}A_1.\cdots\lambda x_n{:}A_n.b$. Moreover, if $\Gamma \equiv x_1{:}A_1, \ldots, x_n{:}A_n$ we write $\prod\Gamma.B$ for $\prod_{i=1}^{n} x_i{:}A_i.B$ and $\boldsymbol{\lambda}\Gamma.b$ for $\boldsymbol{\lambda}_{i=1}^{n} x_i{:}A_i.b$.

3 AUT-68: the first AUTOMATH system

During the AUTOMATH-project, several AUTOMATH-languages have been developed. They all have two mechanisms for describing mathematics. One of them is a typed λ-calculus, with the important features of λ-abstraction, λ-application and β-reduction. The other mechanism is the use of definitions. The definition mechanism is the same for most AUTOMATH-systems, and the difference between the various systems is mainly caused by different λ-calculi that are included in them. In this section we will describe the system AUT-68 which not only is one of the first AUTOMATH-systems, but also a system with a relatively simple typed λ-calculus, which makes it easier to focus on the (less known) definition mechanism.
AUT-68 has also some other characteristics that are not present in many type systems: η-reduction, Π-application and Π-reduction. In order to keep the attention focussed on the definition system without being diverted by these other characteristics, we will look at AUT-68 *without* η-reduction, Π-application and Π-reduction.
A more extensive description of AUT-68, on which our description below is based, can be found in [4].

3a Books, lines and expressions

Definitions in AUTOMATH-systems are stored in so-called *books*. For writing books in AUT-68 we need

- The symbol **type**;
- A set $\mathcal{V}$ of variables (called *block openers* in [7]);

- A set $\mathcal{C}$ of constants;
- The symbols ( ) [ ] : — and ,.

We assume that $\mathcal{V}$ and $\mathcal{C}$ are infinite, or at least offer us as much different elements as needed. We also assume that $\mathcal{V} \cap \mathcal{C} = \varnothing$ and that $\mathtt{type} \notin \mathcal{V} \cup \mathcal{C}$.
The elements of $\mathcal{V} \cup \mathcal{C}$ are called *identifiers* in [7].

Definition 3.1 (Expressions) We define the set $\mathcal{E}_{68}$ of AUT-68 -expressions inductively:

- If $x \in \mathcal{V}$ then $x \in \mathcal{E}_{68}$;
- If $a \in \mathcal{C}$, $n \in \mathbb{N}$ ($n = 0$ is allowed) and $\Sigma_1, \ldots, \Sigma_n \in \mathcal{E}_{68}$ then $a(\Sigma_1, \ldots, \Sigma_n) \in \mathcal{E}_{68}$.
- If $x \in \mathcal{V}$, $\Sigma \in \mathcal{E}_{68} \cup \{\mathtt{type}\}$ and $\Omega \in \mathcal{E}_{68}$ then $[x{:}\Sigma]\Omega \in \mathcal{E}_{68}$;
- If $\Sigma_1, \Sigma_2 \in \mathcal{E}_{68}$ then $\langle \Sigma_2 \rangle \Sigma_1 \in \mathcal{E}_{68}$.

Sometimes we will consider the set $\mathcal{E}_{68}^{+} \stackrel{\text{def}}{=} \mathcal{E}_{68} \cup \{\mathtt{type}\}$.

$[x{:}\Sigma]\Omega$ is AUTOMATH-notation for abstraction terms. In PTS-notation one would write $\lambda x{:}\Sigma.\Omega$ or $\Pi x{:}\Sigma.\Omega$. In a relatively simple AUTOMATH-system like AUT-68, it is easy to determine whether $\lambda x{:}\Sigma.\Omega$ or $\Pi x{:}\Sigma.\Omega$ is the correct interpretation for $[x{:}\Sigma]\Omega$. This is harder in more complicated systems like AUT-QE.
$\langle \Sigma_2 \rangle \Sigma_1$ is AUTOMATH-notation for the application of the function Σ_1 to the argument Σ_2. In PTS-notation: $\Sigma_1 \Sigma_2$.

Definition 3.2 (Free variables)

- $\mathrm{FV}(x) \stackrel{\text{def}}{=} \{x\}$;
- $\mathrm{FV}(a(\Sigma_1, \ldots, \Sigma_n)) \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} \mathrm{FV}(\Sigma_i)$;
- $\mathrm{FV}([x{:}\Sigma]\Omega) \stackrel{\text{def}}{=} \mathrm{FV}(\Sigma) \cup (\mathrm{FV}(\Omega) \setminus \{x\})$;
- $\mathrm{FV}(\langle \Sigma_2 \rangle \Sigma_1) \stackrel{\text{def}}{=} \mathrm{FV}(\Sigma_1) \cup \mathrm{FV}(\Sigma_2)$.

Convention 3.3 We make the usual convention that names of bound variables in an expression differ from the free variables in that expression.
We use $\equiv$ to denote syntactical equivalence (up to renaming of bound variables) on terms.

Definition 3.4 If $\Omega, \Sigma_1, \ldots, \Sigma_n$ are expressions, and $x_1, \ldots, x_n$ are distinct variables, then

$$\Omega[x_1, \ldots, x_n{:=}\Sigma_1, \ldots, \Sigma_n]$$

denotes the expression Ω in which all free occurrences of $x_1, \ldots, x_n$ have simultaneously been replaced by $\Sigma_1, \ldots, \Sigma_n$. This, again, is an expression (this can be proved by induction on the structure of Ω).
$\mathtt{type}[x_1, \ldots, x_n{:=}\Sigma_1, \ldots, \Sigma_n]$ is defined as $\mathtt{type}$.

Definition 3.5 (Books and lines) An AUT-68-*book* (or *book* if no confusion arises) is a nonempty, finite list of (AUT-68)-lines.
An AUT-68-*line* (or *line* if no confusion arises) is a 4-tuple $(\Gamma; k; \Sigma_1; \Sigma_2)$. Here,

- Γ is a context, i.e. a finite (possibly empty) list $x_1{:}\alpha_1, \ldots, x_n{:}\alpha_n$, where the x_is are different elements of $\mathcal{V}$ and the α_is are elements of $\mathcal{E}_{68}^{+}$;
- k is an element of $\mathcal{V} \cup \mathcal{C}$;
- Σ_1 can be

- An element of $\mathcal{E}_{68}$ (if $k \in \mathcal{C}$);
- The symbol PN (if $k \in \mathcal{C}$);
- The symbol — (if $k \in \mathcal{V}$)

- Σ_2 is an element of $\mathcal{E}_{68}^+$.

Note that, for $k \in \mathcal{C}$, there are two possibilities for Σ_1:

- Σ_1 is an element of $\mathcal{E}_{68}$; then k is a *defined constant*;
- Σ_1 is the symbol PN; then k is a *primitive notion*[1].

If $k \in \mathcal{V}$ (and $\Sigma_1 =$—), then k is a newly introduced variable.

Intuitively, a book $\mathfrak{B}$ can be seen as a list of definitions. For $\Sigma_1 \in \mathcal{E}_{68}$, a line $(\Gamma; k; \Sigma_1; \Sigma_2)$ of $\mathfrak{B}$ is a definition. k must be interpreted as the definiendum; Σ_1 as the definiens, and Σ_2 as the type of k. Γ is the context in which this definition takes place. In two cases, the normal notion of "definition" is extended: In the case that $\Sigma_1 =$ PN, the line $(\Gamma; k; \Sigma_1; \Sigma_2)$ is a "primitive" definition, introducing a constant without definiens. In the case that $\Sigma_1 =$—, the line $(\Gamma; k; \Sigma_1; \Sigma_2)$ "defines" k to be a new variable.

Not all books are good books. If $(\Gamma; k; \Sigma_1; \Sigma_2)$ is a line of a book $\mathfrak{B}$, the expressions Σ_1 and Σ_2 (as long as Σ_1 isn't PN or —, and Σ_2 isn't **type**) must be well-defined, i.e. the symbols occurring in them must have been defined in previous parts of $\mathfrak{B}$. The same holds for the type assignments $x_i{:}\alpha_i$ that occur in Γ. Moreover, if Σ_1 isn't PN or —, then Σ_1 must be of the same type as k, hence Σ_1 must be of type Σ_2 (within the context Γ). Finally, there should be only one definition of an object, so k shouldn't occur in the preceding lines of the book.
Hence we need notions of correctness (with respect to a book and/or a context) and we need a definition of the notion "Σ_1 is of type Σ_2" (within a book and a context). They are defined below.

Definition 3.6 (Correct contexts) Let $\mathfrak{B}$ be a book.

- The empty context $\varnothing$ is correct (with respect to $\mathfrak{B}$);
- If Γ is a correct context and $\mathfrak{B}$ contains a line $(\Gamma; x; —; \alpha)$, then $\Gamma, x{:}\alpha$ is a correct context (with respect to $\mathfrak{B}$).

Definition 3.7 (Correct books)

- The empty book (consisting of 0 lines) is correct;
- If $\mathfrak{B}$ is a correct book and $\mathfrak{B}'$ is the book consisting of the lines of $\mathfrak{B}$, and finally a new line $(\Gamma; k; \Sigma_1; \Sigma_2)$, then $\mathfrak{B}'$ is correct if and only if
  - Γ is correct with respect to $\mathfrak{B}$;
  - k doesn't occur in $\mathfrak{B}$;
  - $\Sigma_1 \equiv$ PN, $\Sigma_1 \equiv$ —, or Σ_1 is a correct expression with respect to $\mathfrak{B}$ and Γ;
  - $\Sigma_2 \equiv$ **type**, or Σ_2 is a correct expression of type **type** with respect to $\mathfrak{B}$ and Γ;
  - If Σ_1 is an expression, then it has a type that is definitionally equal to Σ_2.

  The notions "correct expression" and "definitionally equal" are defined below.

[1] Examples of primitive notions are the axiomatically introduced number 0 in $\mathbb{N}$ and the "classical" axiom $p \vee \neg p$, for all propositions p.

Definition 3.8 (Correct expressions) Let $\mathfrak{B}$ be a book, Γ a context that is correct with respect to $\mathfrak{B}$. We define the notion Σ *is a correct expression of type* Ω *with respect to* $\mathfrak{B};\Gamma$, shorthand: $\mathfrak{B};\Gamma \vdash \Sigma : \Omega$, by induction on Σ:

$$\frac{x{:}\alpha \in \Gamma}{\mathfrak{B};\Gamma \vdash x{:}\alpha}$$

$$\frac{(x_1{:}\alpha_1,\ldots,x_n{:}\alpha_n;b;\Omega_1;\Omega_2) \in \mathfrak{B} \quad \mathfrak{B};\Gamma \vdash \Sigma_i{:}\alpha_i[x_1,\ldots,x_{i-1}{:=}\Sigma_1,\ldots,\Sigma_{i-1}] \quad (i=1,\ldots,n)}{\mathfrak{B};\Gamma \vdash b(\Sigma_1,\ldots,\Sigma_n) : \Omega_2[x_1,\ldots,x_n{:=}\Sigma_1,\ldots,\Sigma_n]}$$

$$\frac{\mathfrak{B};\Gamma \vdash \Sigma_1{:}\mathtt{type} \qquad \mathfrak{B};\Gamma,x{:}\Sigma_1 \vdash \Omega_1{:}\mathtt{type}}{\mathfrak{B};\Gamma \vdash [x{:}\Sigma_1]\Omega_1 : \mathtt{type}}$$

$$\frac{\mathfrak{B};\Gamma \vdash \Sigma_1{:}\mathtt{type} \qquad \mathfrak{B};\Gamma,x{:}\Sigma_1 \vdash \Omega_1{:}\mathtt{type} \qquad \mathfrak{B};\Gamma,x{:}\Sigma_1 \vdash \Sigma_2{:}\Omega_1}{[x{:}\Sigma_1]\Sigma_2 : [x{:}\Sigma_1]\Omega_1}$$

$$\frac{\mathfrak{B};\Gamma \vdash \Sigma_1 : [x{:}\Omega_1]\Omega_2 \qquad \mathfrak{B};\Gamma \vdash \Sigma_2{:}\Omega_1}{\mathfrak{B};\Gamma \vdash \langle\Sigma_2\rangle\Sigma_1 : \Omega_2[x{:=}\Sigma_2]}$$

$$\frac{\mathfrak{B};\Gamma \vdash \Sigma{:}\Omega_1 \qquad \mathfrak{B};\Gamma \vdash \Omega_2{:}\mathtt{type} \qquad \Omega_1 \text{ and } \Omega_2 \text{ are definitionally equal}}{\mathfrak{B};\Gamma \vdash \Sigma : \Omega_2}$$

As was explained before, a line $(\Gamma;k;\Sigma_1;\Sigma_2)$ of a book should be read as "in context Γ, k is defined as Σ_1 of type Σ_2". That is why we did not demand in Definition 3.7 that the type of Σ_1 must be (syntactically) equal to Σ_2, but only definitionally equal; this also explains the last rule of Definition 3.8.

3b Definitional equality

We still need to give a definition of "definitional equality". This definition is based on both the definition mechanism and the abstraction mechanism of AUT-68. The abstraction mechanism provides the well-known notions of β-equality and η-equality, originating from the rules of β-conversion and η-conversion:

$$\begin{array}{rcll} \langle\Sigma\rangle[x{:}\Omega_2]\Omega_1 & \to_\beta & \Omega_1[x{:=}\Sigma] & \\ [x{:}\Omega]\langle x\rangle\Sigma & \to_\eta & \Sigma & (x \notin \mathrm{FV}(\Sigma)) \end{array}$$

For the moment, we will regard AUT-68 without η-equality. We will use notations like $\twoheadrightarrow_\beta$, $=_\beta$, as usual.

We now describe the definition mechanism of AUT-68 via the notion of d-*equality*.

Definition 3.9 (d-equality) Let $\mathfrak{B}$ be a book, Γ a correct context with respect to $\mathfrak{B}$, and Σ a correct expression with respect to $\mathfrak{B};\Gamma$. We define the d-*normal form* $\mathrm{nf_d}(\Sigma)$ of Σ by induction on expressions and on the length of the book $\mathfrak{B}$.

- If Σ is a variable x, then $\mathrm{nf_d}(\Sigma) \stackrel{\mathrm{def}}{=} x$;
- Now assume $\Sigma \equiv b(\Omega_1,\ldots,\Omega_n)$, and assume that the normal forms of the Ω_is have already been defined.
  Determine a line $(\Delta;b;\Xi_1;\Xi_2)$ in the book $\mathfrak{B}$ (There is exactly one such line).
  Write $\Delta \equiv x_1{:}\alpha_1,\ldots,x_n{:}\alpha_n$. Distinguish:
  - $\Xi_1 \equiv$ —. This case doesn't occur, as $b \in C$;
  - $\Xi_1 \equiv$ PN. Then define $\mathrm{nf_d}(\Sigma) \stackrel{\mathrm{def}}{=} b(\mathrm{nf_d}(\Omega_1),\ldots,\mathrm{nf_d}(\Omega_n))$.
  - Ξ_1 is an expression. Then Ξ_1 is correct with respect to a book $\mathfrak{B}'$ that contains fewer lines than $\mathfrak{B}$ ($\mathfrak{B}'$ doesn't contain the line $(\Delta;b;\Xi_1;\Xi_2)$, and all lines of $\mathfrak{B}'$ are also lines of $\mathfrak{B}$), hence we can assume that $\mathrm{nf_d}(\Xi_1)$ has already been defined. Now define

$$\mathrm{nf_d}(\Sigma) \stackrel{\mathrm{def}}{=} \mathrm{nf_d}(\Xi_1)[x_1,\ldots,x_n{:=}\mathrm{nf_d}(\Omega_1),\ldots,\mathrm{nf_d}(\Omega_n)];$$

- If $\Sigma \equiv [x{:}\Omega_1]\Omega_2$ then $\mathrm{nf_d}(\Sigma) \stackrel{\mathrm{def}}{=} [x{:}\mathrm{nf_d}(\Omega_1)]\mathrm{nf_d}(\Omega_2)$;
- If $\Sigma \equiv (\Omega_2)\Omega_1$ then $\mathrm{nf_d}(\Sigma) \stackrel{\mathrm{def}}{=} (\mathrm{nf_d}(\Omega_2))\mathrm{nf_d}(\Omega_1)$.

We write $\Sigma_1 =_\mathrm{d} \Sigma_2$ when $\mathrm{nf_d}(\Sigma_1) \equiv \mathrm{nf_d}(\Sigma_2)$.

As we see, the d-normal form $\mathrm{nf_d}(\Sigma)$ of a correct expression Σ depends on the book $\mathfrak{B}$, and in order to be completely correct we should write $\mathrm{nf}_{\mathrm{d}\mathfrak{B}}(\Sigma)$ instead of only $\mathrm{nf_d}(\Sigma)$. We will, however, omit the subscript $\mathfrak{B}$ as long as no confusion arises.

Definition 3.10 (Definitional equality) Σ_1 and Σ_2 are called *definitionally equal* (with respect to a book $\mathfrak{B}$) if and only if $\Sigma_1 =_{\beta\mathrm{d}} \Sigma_2$.

With this definition, the description of AUT-68 is completed. Again, definitional equality of expressions Σ_1 and Σ_2 depends on the book $\mathfrak{B}$, so we should write $=_{\beta\mathrm{d}\mathfrak{B}}$ instead of $=_{\beta\mathrm{d}}$. Again we leave out the subscript $\mathfrak{B}$ as long as no confusion arises.

As an alternative to Definition 3.9, we describe the notion of d-equality via a reduction relation.

Definition 3.11 Let $\mathfrak{B}$ be a book, Γ a correct context with respect to $\mathfrak{B}$, and Σ a correct expression with respect to $\mathfrak{B}; \Gamma$. We define $\Sigma \rightarrow_\delta \Omega$ by the usual compatibility rules, and

(δ) If $\Sigma = b(\Sigma_1, \ldots, \Sigma_n)$, and $\mathfrak{B}$ contains a line $(x_1{:}\alpha_1, \ldots, x_n{:}\alpha_n; b; \Xi_1; \Xi_2)$ where $\Xi_1 \in \mathcal{E}^+_{68}$, then

$$\Sigma \rightarrow_\delta \Xi_1[x_1, \ldots, x_n{:=}\Sigma_1, \ldots, \Sigma_n]$$

We say that Σ is in δ-normal form if for no expression Ω, $\Sigma \rightarrow_\delta \Omega$, and use notations like $\twoheadrightarrow_\delta$ and $\twoheadrightarrow^+_\delta$ as usual. $\rightarrow_\delta$ depends on $\mathfrak{B}$, but as we did before with $\mathrm{nf_d}$ and $=_\mathrm{d}$ we only mention this explicitly if it is not clear in relation to which book $\mathfrak{B}$ $\rightarrow_\mathrm{d}$ is considered.

We have:

Lemma 3.12

1. *$\rightarrow_\delta$ has the Church-Rosser-property;*
2. *$\mathrm{nf_d}(\Sigma)$ is the (unique) δ-normal form of Σ;*
3. *$\Sigma =_\delta \Omega$ if and only if $\Sigma =_\mathrm{d} \Omega$.*
4. *$\rightarrow_\delta$ is strongly normalising.*

PROOF: AUT-68 with $\rightarrow_\delta$ can be seen as an orthogonal term rewrite system.

1. Such a term rewrite system has the Church-Rosser property (see [23]).
2. It is not hard to show that $\Sigma \twoheadrightarrow_\delta \mathrm{nf_d}(\Sigma)$. By induction on the definition of $\mathrm{nf_d}(\Sigma)$ one shows that $\mathrm{nf_d}(\Sigma)$ is in δ-normal form. The uniqueness of this normal form follows from the Church-Rosser property.
3. $\Sigma =_\delta \Omega$ if and only if both Σ and Ω reduce to $\mathrm{nf_d}(\Sigma)$, if and only if $\Sigma =_\mathrm{d} \Omega$.
4. We already know that $\rightarrow_\delta$ is weakly normalising (by 2). Moreover, the definition of $\mathrm{nf_d}(\Sigma)$ in 3.9 induces an innermost reduction strategy. By a theorem by O'Donnell (see [27], or pages 75–76 of [23]), $\rightarrow_\delta$ is strongly normalising.

□

4 From AUT-68 towards a PTS

We want to give a more modern description of AUT-68, preferably in the framework of the Pure Type Systems.
First, we must make a translation of the expressions in AUT-68 to typed λ-terms. This translation is very straightforward:

Definition 4.1 We define a mapping $\overline{[\ldots]}$ from the correct expressions in $\mathcal{E}_{68}$ (relative to a book $\mathfrak{B}$ and a context Γ) to $\mathbb{T}$, the set of terms for PTSs (see Definition 2.1). We assume that $\mathcal{C} \cup \mathcal{V} \subseteq \mathbb{V}$ ($\mathbb{V}$ is the set of variables for PTS-terms).

- $\overline{x} \stackrel{\text{def}}{=} x$ for $x \in \mathcal{V}$;
- $\overline{b(\Sigma_1, \ldots, \Sigma_n)} \stackrel{\text{def}}{=} b\overline{\Sigma_1} \cdots \overline{\Sigma_n}$;
- $\overline{[x{:}\Sigma]\Omega} \stackrel{\text{def}}{=} \Pi x{:}\overline{\Sigma}.\overline{\Omega}$ if $[x{:}\Sigma]\Omega$ has type **type**, otherwise $\overline{[x{:}\Sigma]\Omega} \stackrel{\text{def}}{=} \lambda x{:}\overline{\Sigma}.\overline{\Omega}$;
- $\overline{\langle\Omega\rangle\Sigma} \stackrel{\text{def}}{=} \overline{\Sigma\Omega}$.

Moreover, we define: $\overline{\mathtt{type}} \stackrel{\text{def}}{=} *$.

With this translation in mind, we want to find a type system $\lambda 68$ that "suits" AUT68, i.e. if Σ is a correct expression of type Ω with respect to a book $\mathfrak{B}$ and a context Γ, then we want $\mathfrak{B}', \Gamma' \vdash \overline{\Sigma} : \overline{\Omega}$ to be derivable in $\lambda 68$, and vice versa. Here, $\mathfrak{B}'$ and Γ' are some suitable translations of $\mathfrak{B}$ and Γ. The search for a suitable $\lambda 68$ will concentrate on three points, which we first discuss informally. In the next section we give a formal definition of $\lambda 68$, and prove that it has the property we described above.

4a The choice of the correct formation (Π) rules

The definition of correct expressions 3.8 gives, when we keep in mind that $\overline{\mathtt{type}} \equiv *$, a clear answer on the question of which Π-rules are implied by the abstraction mechanism of AUT-68. The rule

$$\textit{If } \mathfrak{B}, \Gamma \vdash \Sigma_1 : \mathtt{type} \textit{ and } \mathfrak{B}, \Gamma, x{:}\Sigma_1 \vdash \Omega_1 : \mathtt{type} \textit{ then } \mathfrak{B}, \Gamma \vdash [x{:}\Sigma_1]\Omega_1 : \mathtt{type}.$$

immediately translates into Π-rule $(*, *, *)$ for PTSs.

It is, however, not immediately clear which Π-rules are induced by the definition mechanism of AUT-68.
Let $\Sigma \equiv b(\Sigma_1, \ldots, \Sigma_n)$ be a correct expression of type Ω with respect to a book $\mathfrak{B}$ and a context Γ. There is a line

$$(x_1{:}\alpha_1, \ldots, x_n{:}\alpha_n; b; \Xi_1; \Xi_2)$$

in $\mathfrak{B}$ such that Σ_i is a correct expression with respect to $\mathfrak{B}$ and Γ, and has a type that is definitionally equal to $\alpha_i[x_1, \ldots, x_{i-1}{:=}\Sigma_1, \ldots, \Sigma_{i-1}]$. We also know that $\Omega =_{\beta\mathrm{d}} \Xi_2[x_1, \ldots, x_n{:=}\Sigma_1, \ldots \Sigma_n]$. Now $\overline{\Sigma} \equiv b\overline{\Sigma_1} \cdots \overline{\Sigma_n}$, and, assuming that we can derive in $\lambda 68$ that $\overline{\Sigma_i}$ has type

$$\overline{\alpha_i}[x_1, \ldots, x_{i-1}{:=}\overline{\Sigma_1}, \ldots, \overline{\Sigma_{i-1}}],$$

it isn't unreasonable to give b as type $\prod_{i=1}^{n} x_i{:}\overline{\alpha_i}.\overline{\Xi_2}$. Then we can derive (using the application rule that we will introduce for $\lambda 68$ n times) that $\overline{\Sigma}$ has type $\overline{\Omega}$ in $\lambda 68$.

It is important to notice that the type of b, $\prod_{i=1}^{n} x_i{:}\overline{\alpha_i}.\overline{\Xi_2}$, does not necessarily have an equivalent in AUT-68, as in AUT-68 abstractions over **type** are not allowed (only abstractions over expressions Σ that have **type** as type are possible). This is the reason to create a special sort $\triangle$, in which the possible types of AUT-68 constants and abbreviations are stored.
To construct $\Pi x_n{:}\overline{\alpha_n}.\overline{\Xi_2}$ from $\overline{\Xi_2}$, we need rules of the form $(*, *, s_1)$, $(*, \Box, s_2)$, $(\Box, *, s_3)$, $(\Box, \Box, s_4)$. A straightforward choice is $s_1 \equiv s_2 \equiv s_3 \equiv s_4 \equiv \triangle$.
To construct $\prod_{i=1}^{n} .x_i{:}\overline{\alpha_i}.\overline{\Xi_2}$ from $\Pi x_n{:}\overline{\alpha_n}.\overline{\Xi_2}$ we introduce rules $(*, \triangle, \triangle)$ and $(\Box, \triangle, \triangle)$ for similar

reasons.

In Example 5.2.4.8 of [3], there is no rule $(*,*,\Delta)$. In principle, this rule is superfluous, as types constructed with $(*,*,\Delta)$ can also be constructed using rule $(*,*,*)$. Nevertheless we want to maintain this rule:

- First of all, the presence of both $(*,*,*)$ and $(*,*,\Delta)$ in the system stresses the fact that AUT-68 has two type mechanisms: one provided by the definition system and one by the abstraction mechanism.
- Secondly, there are technical arguments to make a distinction between types formed by the abstraction mechanism and types that appear via the definition mechanism. In this paper, we will denote product types constructed by the abstraction mechanism in the usual way (so: $\Pi x{:}A.B$), whilst we will use the notation $\otimes x{:}A.B$ for a type constructed by the definition mechanism. Hence, we have for the constant b above that $b : \bigotimes_{i=1}^{n} x_i{:}\overline{\alpha_i}.\overline{\Xi_2}$.
- There is another reason to make a distinction between types formed by the abstraction mechanism and types that appear in the translation via the definition mechanism. For the moment, we consider AUT-68 *without* so-called Π-application. In AUT-68 with Π-application, however, the application rule of Definition 3.8

  $$\textit{If } \mathfrak{B};\Gamma \vdash \Sigma_1{:}[x{:}\Omega_1]\Omega_2 \textit{ and } \mathfrak{B};\Gamma \vdash \Sigma_2{:}\Omega_1 \textit{ then } \mathfrak{B};\Gamma \vdash \langle\Sigma_2\rangle\Sigma_1{:}\Omega_2[x{:=}\Sigma_2]$$

  is replaced by

  $$\textit{If } \mathfrak{B};\Gamma \vdash \Sigma_1{:}[x{:}\Omega_1]\Omega_2 \textit{ and } \mathfrak{B};\Gamma \vdash \Sigma_2{:}\Omega_1 \textit{ then } \mathfrak{B};\Gamma \vdash \langle\Sigma_2\rangle\Sigma_1{:}\langle\Sigma_2\rangle\Omega_2$$

  but the rule describing the type of $b(\Sigma_1,\ldots,\Sigma_n)$ is the same as the rule in Definition 3.8. This means that in the translation of AUT-68 *with* Π-application, the application rule for Π-terms has to be different from the application rule for ⊗-terms.

4b The different treatment of constants and variables

When we seek for a translation in $\lambda 68$ of the AUT-68 judgement $\mathfrak{B};\Gamma \vdash \Sigma : \Omega$, we must pay extra attention to the translation of $\mathfrak{B}$, as there is no equivalent of books in PTSs. Our solution is to store the type information on constants of $\mathfrak{B}$ in the context. Therefore, contexts of $\lambda 68$ will have the form $\Delta;\Gamma$. The left part Δ contains type information on constants, and can be seen as the translation of $\mathfrak{B}$. In the right part Γ we find the usual type information on variables.
It is natural to store type information on constants in the left part of a context. Let $\mathfrak{B}$ be a correct AUT-68 book, to which we add a line $(\Gamma; b; \textsc{pn}; \Xi_2)$. Then $\Gamma \equiv x_1{:}\alpha_1,\ldots,x_n{:}\alpha_n$ is a correct context with respect to $\mathfrak{B}$, and $\mathfrak{B};\Gamma \vdash \Xi_2{:}\mathtt{type}$ (or $\Xi_2 \equiv \mathtt{type}$). In $\lambda 68$ we can work as follows. Assume the information on constants in $\mathfrak{B}$ has been translated into the left part Δ of a $\lambda 68$ context. We have (assuming that $\lambda 68$ is a type system that behaves like AUT-68, and writing $\overline{\Gamma}$ for the translation $x_1{:}\overline{\alpha_1},\ldots,x_n{:}\overline{\alpha_n}$ of Γ):

$$\Delta;\overline{\Gamma} \vdash \overline{\Xi_2}{:}s$$

($s \equiv *$ if $\mathfrak{B};\Gamma \vdash \Xi_2{:}\mathtt{type}$; $s \equiv \Box$ if $\Xi_2 \equiv \mathtt{type}$). Applying the ⊗-formation rule n times, we obtain

$$\Delta;\vdash \bigotimes\overline{\Gamma}.\overline{\Xi_2} : \Delta^2$$

As $\bigotimes\overline{\Gamma}.\overline{\Xi_2}$ is exactly the type that we want to give to b (see the discussion in Subsection 4a), we use this statement as premise for the start rule that introduces b. As the right part $\overline{\Gamma}$ of the original context has disappeared when we applied the ⊗-formation rules, the declaration $b{:}\bigotimes\overline{\Gamma}.\overline{\Xi_2}$ is placed at the end of the left part Δ of that context: The conclusion of the start rule is

$$\Delta, b{:}\bigotimes\overline{\Gamma}.\overline{\Xi_2} \vdash b{:}\bigotimes\overline{\Gamma}.\overline{\Xi_2}$$

[2] If Γ is the empty context, then $\bigotimes\overline{\Gamma}.\overline{\Xi_2} \equiv \overline{\Xi_2}$, and $\overline{\Xi_2}$ has type $*$ or $\Box$ instead of Δ

Adding $b{:}\bigotimes\overline{\Gamma}.\overline{\Xi_2}$ at the end of Δ can be compared with adding the line $(\Gamma; b; \text{PN}; \Xi_2)$ at the end of $\mathfrak{B}$.
The process above can be caught in one rule:

$$\frac{\Delta;\overline{\Gamma} \vdash \overline{\Xi_2}{:}s_1 \qquad \Delta;\vdash \bigotimes\overline{\Gamma}.\overline{\Xi_2}{:}s_2}{\Delta, b{:}\bigotimes\overline{\Gamma}.\overline{\Xi_2};\vdash b{:}\bigotimes\overline{\Gamma}.\overline{\Xi_2}}$$

Here $s_1 \in \{*, \Box\}$ (compare: $\Xi_2{:}\mathbf{type}$ or $\Xi_2 \equiv \mathbf{type}$) and $s_2 \in \{*, \Box, \triangle\}$ (usually, $s_2 \equiv \triangle$. The cases $s_2 \equiv *, \Box$ only occur if Γ is empty; see footnote 2).

4c The definition system

A line $(x_1{:}\alpha_1, \ldots, x_n{:}\alpha_n; b; \Xi_1; \Xi_2)$, in which b is a constant and $\Xi_1 \in \mathcal{E}_{68}$, represents a definition. It should be read as: For all expressions $\Omega_1, \ldots, \Omega_n$ (obeying certain type conditions), $b(\Omega_1, \ldots, \Omega_n)$ is an abbreviation for $\Xi_1[x_1, \ldots, x_n{:=}\Omega_1, \ldots, \Omega_n]$, and has type $\Xi_2[x_1, \ldots, x_n{:=}\Omega_1, \ldots, \Omega_n]$. So in $\lambda 68$, the context should also mention that $bX_1 \cdots X_n$ "is equal to" $\Xi_1[x_1, \ldots, x_n{:=}X_1, \ldots, X_n]$, for all terms $X_1, \ldots, X_n$. The most straightforward way to do this, is to write

$$b{:=}\left(\boldsymbol{\lambda}_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_1}\right) : \left(\bigotimes_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_2}\right)$$

in the context instead of only $b{:}\bigotimes_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_2}$, and to add a δ-reduction rule that allows to unfold the definition of b:

$$\Delta \vdash b \to_\delta \boldsymbol{\lambda}_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_1}$$

whenever $b{:=}\left(\boldsymbol{\lambda}_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_1}\right) : \left(\bigotimes_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_2}\right) \in \Delta$.
Unfolding the definition of b in a term $b\overline{\Sigma_1}\cdots\overline{\Sigma_n}$ and applying β-reduction n times results in $\overline{\Xi_1}[x_1{:=}\Sigma_1]\cdots[x_n{:=}\Sigma_n]$. This procedure corresponds exactly to the δ-reduction of $b(\Sigma_1, \ldots, \Sigma_n)$ to $\Xi_1[x_1, \ldots, x_n{:=}\Sigma_1, \ldots, \Sigma_n]$ in AUT-68[3].

This method, however, has some disadvantages.

- Look again at a line $(x_1{:}\alpha_1, \ldots, x_n{:}\alpha_n; b; \Xi_1; \Xi_2)$ in some AUT-68 book, and at a term $B \equiv b\overline{\Sigma_1}\cdots\overline{\Sigma_m}$ in $\lambda 68$ for some $m < n$. B has no equivalent in AUT-68: Only after B has been applied to suitable terms $\overline{\Sigma_{m+1}}, \ldots, \overline{\Sigma_n}$ the term $B\overline{\Sigma_{m+1}}\cdots\overline{\Sigma_n}$ has $b(\Sigma_1, \ldots, \Sigma_n)$ as its equivalent in AUT-68. B must not be seen as a term of AUTOMATH, but only as an intermediate result that is necessary to construct the equivalent of the term $b(\Sigma_1, \ldots, \Sigma_n)$. B is recognizable as an intermediate result via its type $\bigotimes_{i=m+1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_2}$, which has sort $\triangle$ (instead of $*$ or $\Box$).
  The method above allows to unfold the definition of b already in B, but it is more in line with AUT-68 to make such unfolding not possible before *all* n arguments $\overline{\Sigma_1}, \ldots, \overline{\Sigma_n}$ have been applied to b, and the construction of the equivalent of $b(\Sigma_1, \ldots, \Sigma_n)$ has been completed.

- $\boldsymbol{\lambda}_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_1}$ has not necessarily an equivalent in AUT-68. Take for instance the constant b in the line $(\alpha{:}\mathbf{type}, b, [x{:}\alpha]x, [x{:}\alpha]\alpha)$. In this case, $\boldsymbol{\lambda}_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_1} \equiv \lambda\alpha{:}*.\lambda x{:}\alpha.x$. Its equivalent in AUT-68 would be $[\alpha{:}\mathbf{type}][x{:}\alpha]x$, but an abstraction $[\alpha{:}\mathbf{type}]$ cannot be made in AUT-68. It is undesirable to allow terms in $\lambda 68$ that do not have an equivalent in AUT-68.

Therefore we choose a different translation. The line $(x_1{:}\alpha_1, \ldots, x_n{:}\alpha_n; b; \Xi_1; \Xi_2)$, where $\Xi_1 \in \mathcal{E}_{68}$, will be translated by putting

$$b{:=}\overline{\Xi_1}{:}\bigotimes_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_2}$$

in the left part of the translated context Δ, and a reduction rule

$$bX_1 \cdots X_n \to_\delta \overline{\Xi_1}[x_1, \ldots, x_n{:=}X_1, \ldots, X_n]$$

is added for all pseudoterms $X_1, \ldots, X_n$. Note that we make an "abuse of language" in the pseudodefinition $b{:=}\overline{\Xi_1}$: b is *not* an abbreviation of $\overline{\Xi_1}$. However, $bx_1 \cdots x_n$ can be seen as an

[3] We can assume that the x_i do not occur in the Σ_j, so the simultaneous substitution $\Xi_1[x_1, \ldots, x_n{:=}\Sigma_1, \ldots, \Sigma_n]$ is equal to $\Xi_1[x_1{:=}\Sigma_1]\cdots[x_n{:=}\Sigma_n]$

abbreviation of $\overline{\Xi_1}$, since $bx_1 \cdots x_n \to_\delta \overline{\Xi_1}$. This is in line with the situation in AUTOMATH, where b in the line $(x_1{:}\alpha_1, \ldots, x_n{:}\alpha_n; b; \Xi_1; \Xi_2)$ must be read as $b(x_1, \ldots, x_n)$. On the other hand, the *type* of b is rendered correctly in $b{:=}\overline{\Xi_1}{:}\bigotimes_{i=1}^n x_i{:}\overline{\alpha_i}.\overline{\Xi_2}$, whereas the type of $bx_1 \cdots x_n$ is $\overline{\Xi_2}$.
Note also that $\overline{\Xi_2}$ is not of the form $\otimes y{:}M.N$, so the number n (necessary for the determination of the δ-reduction rule for b) can be determined from the context: it is equal to the number of $\otimes$-abstractions at the head of the type of b.

5 $\lambda 68$

5a Definition and elementary properties

We give the formal definition of $\lambda 68$, based on the motivation in Section 4.

Definition 5.1 ($\lambda 68$)

1. The pseudoterms of $\lambda 68$ form a set $\mathcal{T}$ defined by

$$\mathcal{T} ::= \mathcal{V} \mid \mathcal{C} \mid \mathcal{S} \mid \mathcal{T}\mathcal{T} \mid \lambda \mathcal{V}{:}\mathcal{T}.\mathcal{T} \mid \Pi \mathcal{V}{:}\mathcal{T}.\mathcal{T} \mid \otimes \mathcal{V}{:}\mathcal{T}.\mathcal{T}$$

   where $\mathcal{S}$ is the set of sorts $\{*, \Box, \triangle\}$.
   We also define the sets of free variables $\text{FV}(T)$ and ("free")[4] constants $\text{FC}(T)$ of a term T in the straightforward way.

2. We define the notion of pseudocontext inductively.

   - ; is a pseudocontext; $\text{DOM}(;) = \varnothing$.
   - If $\Delta; \Gamma$ is a pseudocontext, $x \in \mathcal{V}$, x doesn't occur in $\Delta; \Gamma$ and $A \in \mathcal{T}$ then $\Delta; \Gamma, x{:}A$ is a pseudocontext (x is a newly introduced variable); $\text{DOM}(\Delta; \Gamma, x{:}A) = \text{DOM}(\Delta; \Gamma) \cup \{x\}$.
   - If $\Delta; \Gamma$ is a pseudocontext, $b \in \mathcal{C}$, b doesn't occur in $\Delta; \Gamma$ and $A \in \mathcal{T}$ then $\Delta, b{:}A; \Gamma$ is a pseudocontext (In this case b is a primitive constant; cf. Section 3a); $\text{DOM}(\Delta, b{:}A; \Gamma) = \text{DOM}(\Delta; \Gamma) \cup \{b\}$.
   - If $\Delta; \Gamma$ is a pseudocontext, $b \in \mathcal{C}$, b doesn't occur in $\Delta; \Gamma$; $A \in \mathcal{T}$, and $T \in \mathcal{T}$ then $\Delta, b{:=}T{:}A; \Gamma$ is a pseudocontext (in this case b is a defined constant); $\text{DOM}(\Delta, b{:=}T{:}A; \Gamma) = \text{DOM}(\Delta; \Gamma) \cup \{b\}$.

   Observe that a semicolon is used as the separation mark between the two parts of the context, and that a comma is used to separate the different expressions within each of these parts.
   We define

$$\text{PRIMCONS}(\Delta; \Gamma) = \{b \in \text{DOM}(\Delta; \Gamma) | b \text{ is a primitive constant}\}$$

$$\text{DEFCONS}(\Delta; \Gamma) = \{b \in \text{DOM}(\Delta; \Gamma) | b \text{ is a defined constant}\}$$

3. We define the notion of δ-reduction on pseudoterms. Let Δ be the left part of a pseudocontext. If $(b{:=}T{:}\bigotimes_{i=1}^n x_i{:}A_i.B) \in \Delta$, where B is not of the form $\otimes y{:}B_1.B_2$, then

$$\Delta \vdash bX_1 \cdots X_n \to_\delta T[x_1, \ldots, x_n{:=}X_1, \ldots, X_n]$$

   for all $X_1, \ldots X_n \in \mathcal{T}$.
   We also have the usual compatibility rules on δ-reduction. We use notations like $\twoheadrightarrow_\delta$, $\to_\delta^+$, $=_\delta$ as usual. When there is no confusion about which Δ is considered, we simply write

$$bX_1 \cdots X_n \to_\delta T[x_1, \ldots, x_n{:=}X_1, \ldots, X_n].$$

4. We use the usual notion of β-reduction.

[4] Of course, to call a constant "free" is a bit peculiar, since there are no *bound* constants

5. Statements in $\lambda 68$ have the form $\Delta;\Gamma \vdash A : B$, where $\Delta;\Gamma$ is a pseudocontext and A and B are terms. In the case that a judgement $\Delta;\Gamma \vdash A : B$ is derivable according to the rules below, $\Delta;\Gamma$ is a *legal* context and A and B are *legal* terms.
We write $\Delta;\Gamma \vdash A : B : C$ if both $\Delta;\Gamma \vdash A : B$ and $\Delta;\Gamma \vdash B : C$ are derivable in $\lambda 68$.
Here are the rules:

(Axiom) $\quad ;\vdash * : \Box$

(Start: primcons) $\quad \dfrac{\Delta;\Gamma \vdash B : s_1 \qquad \Delta;\vdash \bigotimes\Gamma.B : s_2}{\Delta, b{:}\bigotimes\Gamma.B;\vdash b : \bigotimes\Gamma.B}(s_1 \equiv *, \Box)$

(Start: defcons) $\quad \dfrac{\Delta;\Gamma \vdash T : B : s_1 \qquad \Delta;\vdash \bigotimes\Gamma.B : s_2}{\Delta, b{:=}T{:}\bigotimes\Gamma.B;\vdash b : \bigotimes\Gamma.B}(s_1 \equiv *, \Box)$

(Start: var) $\quad \dfrac{\Delta;\Gamma \vdash A : s}{\Delta;\Gamma, x{:}A \vdash x : A}(s = *, \Box)$

(Weak: primcons) $\quad \dfrac{\Delta;\vdash M : N \qquad \Delta;\Gamma \vdash B : s_1 \qquad \Delta;\vdash \bigotimes\Gamma.B : s_2}{\Delta, b{:}\bigotimes\Gamma.B;\vdash M : N}(s_1 \equiv *, \Box)$

(Weak: defcons) $\quad \dfrac{\Delta;\vdash M : N \qquad \Delta;\Gamma \vdash T : B : s_1 \qquad \Delta;\vdash \bigotimes\Gamma.B : s_2}{\Delta, b{:=}T{:}\bigotimes\Gamma.B;\vdash M : N}(s_1 \equiv *, \Box)$

(Weak: var) $\quad \dfrac{\Delta;\Gamma \vdash M : N \qquad \Delta;\Gamma \vdash A : s}{\Delta;\Gamma, x{:}A \vdash M : N}(s \equiv *, \Box)$

(Π-form) $\quad \dfrac{\Delta;\Gamma \vdash A : * \qquad \Delta;\Gamma, x{:}A \vdash B : *}{\Delta;\Gamma \vdash \Pi x{:}A.B : *}$

($\otimes$-form) $\quad \dfrac{\Delta;\Gamma \vdash A : s_1 \qquad \Delta;\Gamma, x{:}A \vdash B : s_2}{\Delta;\Gamma \vdash \otimes x{:}A.B : \Delta}(s_1 \equiv *, \Box)$

(λ) $\quad \dfrac{\Delta;\Gamma \vdash \Pi x{:}A.B : * \qquad \Delta;\Gamma, x{:}A \vdash F : B}{\Delta;\Gamma \vdash (\lambda x{:}A.F) : (\Pi x{:}A.B)}$

(App_1) $\quad \dfrac{\Delta;\Gamma \vdash M : \Pi x{:}A.B \qquad \Delta;\Gamma \vdash N : A}{\Delta;\Gamma \vdash MN : B[x{:=}N]}$

(App_2) $\quad \dfrac{\Delta;\Gamma \vdash M : \otimes x{:}A.B \qquad \Delta;\Gamma \vdash N : A}{\Delta;\Gamma \vdash MN : B[x{:=}N]}$

(Conv) $\quad \dfrac{\Delta;\Gamma \vdash M : A \qquad \Delta;\Gamma \vdash B : s \qquad \Delta \vdash A =_{\beta\delta} B}{\Delta;\Gamma \vdash M : B}$

The newly introduced variables in the Start- and Weakening-rules are assumed to be fresh; moreover, when introducing a variable x with a Primcons- or Defcons-rule, we assume $x \in \mathcal{C}$; when introducing x via a Variable-rule, we assume $x \in \mathcal{V}$.

Many basic properties for Pure Type Systems also hold for $\lambda 68$. Due to the split of contexts and the different treatment of constants and variables, these properties are on some points a little bit differently formulated than in, for instance, [3] (where these properties are formulated for standard PTSs).

Lemma 5.2 (Free Variable Lemma) *Assume* $\Delta;\Gamma \vdash M : N$. *Write* $\Delta \equiv b_1{:}B_1, \ldots, b_m{:}B_m$; $\Gamma \equiv x_1{:}A_1, \ldots, x_n{:}A_n$ *(in* Δ, *also expressions* $b_i{:=}T_i{:}B_i$ *may occur, but for uniformity of notation we leave out the* $:=T_i$*-part).*

- $b_1, \ldots, b_m \in \mathcal{C}$ *are all distinct;* $x_1, \ldots, x_n \in \mathcal{V}$ *are all distinct.*

- $\text{FC}(M), \text{FC}(N) \subseteq \{b_1, \ldots, b_m\}$; $\text{FV}(M), \text{FV}(N) \subseteq \{x_1, \ldots, x_n\}$.
- $b_1{:}B_1, \ldots, b_{i-1}{:}B_{i-1} \vdash B_i{:}s_i$ *for a* $s_i \in \{*, \Box, \Delta\}$;
  $\Delta; x_1{:}A_1, \ldots, x_{j-1}{:}A_{j-1} \vdash A_j{:}t_j$ *for a* $t_j \in \{*, \Box\}$.

Lemma 5.3 (Start Lemma) *Let* $\Delta;\Gamma$ *be a legal context. Then* $\Delta;\Gamma \vdash * : \Box$, *and if* $c{:}A \in \Delta;\Gamma$, *or* $c{:=}T{:}A \in \Delta$, *then* $\Delta;\Gamma \vdash x : A$.

Lemma 5.4 (Definition Lemma) *Assume* $\Delta_1, b{:=}T{:}\bigotimes_{i=1}^{n} x_i{:}A_i.B, \Delta_2; \Gamma \vdash M : N$, *where* B *is not of the form* $\otimes y{:}B_1.B_2$. *Then* $\Delta_1; x_1{:}A_1, \ldots, x_n{:}A_n \vdash T : B : s$ *for an* $s \in \{*, \Box\}$.

Lemma 5.5 (Transitivity Lemma) *Let* $\Delta_1;\Gamma_1$ *and* $\Delta_2;\Gamma_2$ *be contexts, of which* $\Delta_1;\Gamma_1$ *is legal. Assume that for all* $c{:}A \in \Delta_2;\Gamma_2$ *and for all* $c{:=}T{:}A \in \Delta_2$, $\Delta_1;\Gamma_1 \vdash c{:}A$.
Then $\Delta_2;\Gamma_2 \vdash B : C \Rightarrow \Delta_1;\Gamma_1 \vdash B : C$.

Lemma 5.6 (Substitution Lemma) *Assume* $\Delta;\Gamma_1, x{:}A, \Gamma_2 \vdash B : C$ *and* $\Delta;\Gamma_1 \vdash D : A$. *Then*

$$\Delta;\Gamma_1, \Gamma_2[x{:=}D] \vdash B[x{:=}D] : C[x{:=}D].$$

Lemma 5.7 (Thinning Lemma) *Let* $\Delta_1;\Gamma_1$ *be a legal context, and let* $\Delta_2;\Gamma_2$ *be a legal context such that* $\Delta_1 \subseteq \Delta_2$ *and* $\Gamma_1 \subseteq \Gamma_2$. *Then* $\Delta_1;\Gamma_1 \vdash A : B \Rightarrow \Delta_2;\Gamma_2 \vdash A : B$.

Lemma 5.8 (Generation Lemma)

- *If* $x \in \mathcal{V}$, $\Delta;\Gamma \vdash x{:}C$ *then there is* $s \in \{*, \Box\}$ *and* $B =_{\beta\delta} C$ *such that* $\Delta;\Gamma \vdash B : s$ *and* $x{:}B \in \Gamma$.
- *If* $b \in \mathcal{C}$, $\Delta;\Gamma \vdash b{:}C$ *then there is* $s \in \mathcal{S}$ *and* $B =_{\beta\delta} C$ *such that* $\Delta;\Gamma \vdash B : s$, *and either* $b{:}B \in \Delta$ *or there is* T *such that* $b{:=}T{:}B \in \Delta$.
- *If* $s \in \mathcal{S}$, $\Delta;\Gamma \vdash s{:}C$ *then* $s \equiv *$ *and* $C =_\beta \Box$.
- *If* $\Delta;\Gamma \vdash MN : C$ *then there are* A, B *such that* $\Delta;\Gamma \vdash M : \Pi x{:}A.B$ *or* $\Delta;\Gamma \vdash M : \otimes x{:}A.B$, *and* $\Delta;\Gamma \vdash N{:}A$ *and* $C =_{\beta\delta} B[x{:=}N]$.
- *If* $\Delta;\Gamma \vdash \lambda x{:}A.b : C$ *then there is* B *such that* $\Delta;\Gamma \vdash \Pi x{:}A.B : *$, $\Delta;\Gamma, x{:}A \vdash b : B$ *and* $C =_{\beta\delta} \Pi x{:}A.B$.
- *If* $\Delta;\Gamma \vdash \Pi x{:}A.B : C$ *then* $C =_{\beta\delta} *$; $\Delta;\Gamma \vdash A{:}*$ *and* $\Delta;\Gamma, x{:}A \vdash B{:}*$.
- *If* $\Delta;\Gamma \vdash \otimes x{:}A.B : C$ *then* $C =_{\beta\delta} \Delta$; $\Delta;\Gamma \vdash A{:}s_1$ *for an* $s_1 \in \{*, \Box\}$ *and* $\Delta;\Gamma, x{:}A \vdash B{:}s_2$.

Lemma 5.9 (Unicity of Types) *If* $\Delta;\Gamma \vdash A : B_1$ *and* $\Delta;\Gamma \vdash A : B_2$ *then* $B_1 =_{\beta\delta} B_2$.

Lemma 5.10 *If* $\Delta;\Gamma \vdash A : B$ *then there is* $s \in \mathcal{S}$ *such that* $B \equiv s$ *or* $\Delta;\Gamma \vdash B : s$.

Lemma 5.11 *If* $\Delta;\Gamma \vdash A : \Pi x{:}B_1.B_2$ *then* $\Delta;\Gamma \vdash B_1 : *$ *and* $\Delta;\Gamma, x{:}B_1 \vdash B_2 : *$;

Lemma 5.12 *If* $\Delta;\Gamma \vdash A : \otimes x{:}B_1.B_2$ *then* $\Delta;\Gamma \vdash B_1 : s_1$ *for a* $s_1 \in \{*, \Box\}$ *and* $\Delta;\Gamma, x{:}B_1 \vdash B_2{:}s_2$ *for a sort* s_2.

5b Reduction and conversion

In this section we show some properties of the reduction relations $\rightarrow_\beta$, $\rightarrow_\delta$ and $\rightarrow_{\beta\delta}$. As δ-reduction also depends on books, we first have to give a translation of AUT-68 books and AUT-contexts to λ68-contexts:

Definition 5.13 Let Γ be a AUT-68-context $x_1{:}\alpha_1, \ldots, x_n{:}\alpha_n$. Then $\overline{\Gamma} \stackrel{\text{def}}{=} x_1{:}\overline{\alpha_1}, \ldots, x_n{:}\overline{\alpha_n}$.

Definition 5.14 Let $\mathfrak{B}$ be a book. We define the left part $\overline{\mathfrak{B}}$ of a pseudocontext in λ68:

- $\overline{\langle\rangle} \stackrel{\text{def}}{=} \langle\rangle$ ($\langle\rangle$ is the empty list);
- $\overline{\mathfrak{B}, (\Gamma; b; \text{PN}; \Omega)} \stackrel{\text{def}}{=} \overline{\mathfrak{B}}, b{:}\bigotimes\overline{\Gamma}.\overline{\Omega}$;
- $\overline{\mathfrak{B}, (\Gamma; x; —; \Omega)} \stackrel{\text{def}}{=} \overline{\mathfrak{B}}$;
- $\overline{\mathfrak{B}, (\Gamma; b; \Sigma; \Omega)} \stackrel{\text{def}}{=} \overline{\mathfrak{B}}, b{:=}\overline{\Sigma}{:}\bigotimes\overline{\Gamma}.\overline{\Omega}$.

Lemma 5.15 *Assume, Σ is a correct expression with respect to a book $\mathfrak{B}$.*

- *$\Sigma \to_\beta \Sigma'$ if and only if $\overline{\Sigma} \to_\beta \overline{\Sigma'}$;*
- *$\Sigma \to_\delta \Sigma'$ (with respect to $\mathfrak{B}$) if and only if $\overline{\Sigma} \to_\delta \overline{\Sigma'}$ (with respect to $\overline{\mathfrak{B}}$).*

PROOF: An easy induction on the structure of Σ. □

The Church-Rosser property of $\to_{\beta\delta}$ will be proved by the method of Parallel Reduction, invented by Martin-Löf and Tait (see Section 3.2 of [2]).

Definition 5.16 Let Δ be the left part of a pseudocontext.
We define a reduction relation $\Rightarrow_{\beta\delta}$ ("parallel reduction") on the set of pseudoterms $\mathcal{T}$:

- For $x \in \mathcal{V}$, $\Delta \vdash x \Rightarrow_{\beta\delta} x$;
- For $b \in \mathcal{C}$, $\Delta \vdash b \Rightarrow_{\beta\delta} b$;
- For $s \in \mathcal{S}$, $\Delta \vdash s \Rightarrow_{\beta\delta} s$;
- If $\Delta \vdash P \Rightarrow_{\beta\delta} P'$ and $\Delta \vdash Q \Rightarrow_{\beta\delta} Q'$ then
  - $\Delta \vdash \lambda x{:}P.Q \Rightarrow_{\beta\delta} \lambda x{:}P'.Q'$;
  - $\Delta \vdash \Pi x{:}P.Q \Rightarrow_{\beta\delta} \Pi x{:}P'.Q'$;
  - $\Delta \vdash \otimes x{:}P.Q \Rightarrow_{\beta\delta} \otimes x{:}P'.Q'$;
  - $\Delta \vdash PQ \Rightarrow_{\beta\delta} P'Q'$.
- If $\Delta \vdash Q \Rightarrow_{\beta\delta} Q'$ and $\Delta \vdash R \Rightarrow_{\beta\delta} R'$ then $\Delta \vdash (\lambda x{:}P.Q)R \Rightarrow_{\beta\delta} Q'[x{:=}R']$;
- If $b{:=}T{:}\bigotimes_{i=1}^{n} x_i{:}A_i.U \in \Delta$, U not of the form $\otimes y{:}U_1.U_2$, $\Delta \vdash T \Rightarrow_{\beta\delta} T'$ and $\Delta \vdash M_i \Rightarrow_{\beta\delta} M_i'$ for $i = 1, \ldots, n$ then $\Delta \vdash bM_1 \cdots M_n \Rightarrow_{\beta\delta} T'[x_1, \ldots, x_n{:=}M_1', \ldots, M_n']$.

Some elementary properties of $\Rightarrow_{\beta\delta}$ are:

Lemma 5.17 (Properties of $\Rightarrow_{\beta\delta}$) *Let Δ be the left part of a pseudocontext. For all pseudoterms M, N:*

1. *$\Delta \vdash M \Rightarrow_{\beta\delta} M$;*
2. *If $\Delta \vdash M \to_{\beta\delta} M'$ then $\Delta \vdash M \Rightarrow_{\beta\delta} M'$;*
3. *If $\Delta \vdash M \Rightarrow_{\beta\delta} M'$ then $\Delta \vdash M \twoheadrightarrow_{\beta\delta} M'$;*
4. *If $\Delta \vdash M \Rightarrow_{\beta\delta} M'$ and $\Delta \vdash N \Rightarrow_{\beta\delta} N'$ then $\Delta \vdash M[y{:=}N] \Rightarrow_{\beta\delta} M'[y{:=}N']$.*

PROOF: All proofs can be given by induction on the structure of M. □

We conclude that $\twoheadrightarrow_{\beta\delta}$ in the context Δ is the reflexive and transitive closure of $\Rightarrow_{\beta\delta}$ in Δ. Therefore, if we want to prove the Church-Rosser theorem for $\twoheadrightarrow_{\beta\delta}$, it suffices to prove the Diamond Property for $\Rightarrow_{\beta\delta}$. We first make some preliminary definitions and remarks:

Lemma 5.18 *Assume, Δ and Δ, Δ' are left parts of legal contexts, and $\text{FC}(M) \subseteq \text{DOM}(\Delta)$. Then $\Delta \vdash M \Rightarrow_{\beta\delta} N$ if and only if $\Delta, \Delta' \vdash M \Rightarrow_{\beta\delta} N$.*

PROOF: By induction on the length of Δ and by induction on the definition of $\Delta \vdash M \Rightarrow_{\beta\delta} N$.
All cases in the definition of $\Delta \vdash M \Rightarrow_{\beta\delta} N$ follow immediately from the induction hypothesis on $\Delta \vdash M \Rightarrow_{\beta\delta} N$, except for the case $bM_1 \cdots M_n \Rightarrow_{\beta\delta} T'[x_1, \ldots, x_n := M_1', \ldots, M_n']$.
As $\mathrm{FC}(M) \subseteq \mathrm{DOM}(\Delta)$, $b \in \mathrm{DOM}(\Delta)$. Write $\Delta \equiv \Delta_1, b{:=}T{:}\bigotimes_{i=1}^n x_i{:}A_i.U, \Delta_2$.

- Notice that T is typable in $\Delta_1; x_1{:}A_1, \ldots, x_n{:}A_n$ (Definition Lemma). By the Free Variable Lemma: $\mathrm{FC}(T) \subseteq \mathrm{DOM}(\Delta_1)$ By the induction hypothesis on the length of Δ we have $\Delta_1 \vdash T \Rightarrow_{\beta\delta} T'$ iff $\Delta \vdash T \Rightarrow_{\beta\delta} T'$, and $\Delta_1 \vdash T \Rightarrow_{\beta\delta} T'$ iff $\Delta, \Delta' \vdash T \Rightarrow_{\beta\delta} T'$.
- We conclude: $\Delta \vdash T \Rightarrow_{\beta\delta} T'$ iff $\Delta, \Delta' \vdash T \Rightarrow_{\beta\delta} T'$.
- By the induction hypothesis on the definition of $\Delta \vdash M \Rightarrow_{\beta\delta} N$, we have $\Delta \vdash M_i \Rightarrow_{\beta\delta} M_i'$ iff $\Delta, \Delta' \vdash M_i \Rightarrow_{\beta\delta} M_i'$.
- As $b{:=}T{:}\bigotimes_{i=1}^n x_i{:}A_i.U$ is element of both Δ and Δ, Δ', and $b \notin \mathrm{DOM}(\Delta')$ (because Δ, Δ' is left part of a legal context) we have $\Delta \vdash bM_1 \cdots M_n \Rightarrow_{\beta\delta} N$ if and only if $\Delta, \Delta' \vdash bM_1 \cdots M_n \Rightarrow_{\beta\delta} N$.

□

For left parts Δ of pseudocontexts and for $M \in \mathcal{T}$ with $\mathrm{FC}(M) \subseteq \mathrm{DOM}(\Delta)$, we define a term M^Δ. In M^Δ, all β-redexes that exist in M are contracted simultaneously (this is a usual step in a proof of Church-Rosser by Parallel Reduction), but also all δ-redexes are contracted. We will show that $N \Rightarrow_{\beta\delta} M^\Delta$, for any N with $M \Rightarrow_{\beta\delta} N$, so M^Δ helps us to show the Diamond Property for $\Rightarrow_{\beta\delta}$.

Definition 5.19 We define, for any left part Δ of a pseudocontext and any $M \in \mathcal{T}$ such that $\mathrm{FC}(M) \subseteq \mathrm{DOM}(\Delta)$, M^Δ. The definition of M^Δ is by induction on the length of Δ and on the structure of M:

- $x^\Delta \stackrel{\mathrm{def}}{=} x$ for any $x \in \mathcal{V}$;
- $b^\Delta \stackrel{\mathrm{def}}{=} b$ for any $b \in \mathcal{C} \setminus \mathrm{DEFCONS}(\Delta;)$;
- $s^\Delta \stackrel{\mathrm{def}}{=} s$ for any $s \in \mathcal{S}$;
- $(\lambda x{:}P.Q)^\Delta \stackrel{\mathrm{def}}{=} \lambda x{:}P^\Delta.Q^\Delta$;
  $(\Pi x{:}P.Q)^\Delta \stackrel{\mathrm{def}}{=} \Pi x{:}P^\Delta.Q^\Delta$;
  $(\otimes x{:}P.Q)^\Delta \stackrel{\mathrm{def}}{=} \otimes x{:}P^\Delta.Q^\Delta$;
- $(PQ)^\Delta \stackrel{\mathrm{def}}{=} P^\Delta Q^\Delta$ if PQ is not a $\beta\delta$-redex;
- $((\lambda x{:}P.Q)R)^\Delta \stackrel{\mathrm{def}}{=} Q^\Delta[x := R^\Delta]$;
- If $M \equiv bM_1 \cdots M_n$, and $\Delta \equiv \Delta_1, b{:=}T{:}\bigotimes_{i=1}^n x_i{:}A_i.U, \Delta_2$, where U is not of the form $\otimes y{:}U_1.U_2$, then $\Delta_1; x_1{:}A_1, \ldots, x_n{:}A_n \vdash T : U$ (due to the Definition Lemma), so we can assume that T^{Δ_1} has already been defined.
  Define $(bM_1 \cdots M_n)^\Delta \stackrel{\mathrm{def}}{=} T^{\Delta_1}[x_1, \ldots, x_n := M_1^\Delta, \ldots, M_n^\Delta]$.

Lemma 5.20 *Let Δ be the left part of a legal context. $\Delta \vdash M \Rightarrow_{\beta\delta} M^\Delta$ for all M with $\mathrm{FC}(M) \subseteq \mathrm{DOM}(\Delta)$.*

PROOF: By induction on the definition of M^Δ.
We only treat the case $bM_1 \cdots M_n \Rightarrow_{\beta\delta} (bM_1 \cdots M_n)^\Delta$.
As in the definition of $(bM_1 \cdots M_n)^\Delta$, write $\Delta \equiv \Delta_1, b{:=}T{:}\bigotimes_{i=1}^n x_i{:}A_i.U, \Delta_2$.
By induction, we may assume that $\Delta_1 \vdash T \Rightarrow_{\beta\delta} T^{\Delta_1}$ and $\Delta \vdash M_i \Rightarrow_{\beta\delta} M_i^\Delta$.
By the Definition Lemma, T is typable in $\Delta_1; x_1{:}A_1, \ldots, x_n{:}A_n$, so by the Free Variable Lemma, $\mathrm{FC}(T) \subseteq \mathrm{DOM}(\Delta_1)$. By Lemma 5.18, $\Delta \vdash T \Rightarrow_{\beta\delta} T^{\Delta_1}$.
So $\Delta \vdash bM_1 \cdots M_n \Rightarrow_{\beta\delta} T^{\Delta_1}[x_1, \ldots, x_n := M_1^\Delta, \ldots, M_n^\Delta]$. □

Theorem 5.21 *Let Δ be the left part of a legal context and assume* $\text{FC}(M) \subseteq \text{DOM}(\Delta)$. *Assume* $\Delta \vdash M \Rightarrow_{\beta\delta} N$. *Then* $\Delta \vdash N \Rightarrow_{\beta\delta} M^{\Delta}$.

PROOF: Induction on the the definition of M^{Δ}.

- $M \equiv x$. Then $N \equiv x$ and $M^{\Delta} \equiv x$.
- $M \equiv b$ and $b \in \mathcal{C} \setminus \text{DEFCONS}(\Delta;)$. Then $N \equiv b$ and $M^{\Delta} \equiv b$.
- $M \equiv s$. Then $N \equiv s$ and $M^{\Delta} \equiv s$.
- $M \equiv \lambda x{:}P.Q$. Then $N \equiv \lambda x{:}P'.Q'$ for some P', Q' with $\Delta \vdash P \Rightarrow_{\beta\delta} P'$ and $\Delta \vdash Q \Rightarrow_{\beta\delta} Q'$. By the induction hypothesis on P and Q we find $\Delta \vdash P' \Rightarrow_{\beta\delta} P^{\Delta}$ and $\Delta \vdash Q' \Rightarrow_{\beta\delta} Q^{\Delta}$. Therefore $\Delta \vdash \lambda x{:}P'.Q' \Rightarrow_{\beta\delta} \lambda x{:}P^{\Delta}.Q^{\Delta}$.
  The cases $M \equiv \Pi x{:}P.Q$, $M \equiv \otimes x{:}P.Q$, and $M \equiv PQ$ where PQ is not a β-redex, are proved similarly.
- $M \equiv (\lambda x{:}P.Q)R$. Distinguish:
  - $N \equiv (\lambda x{:}P'.Q')R'$ for P', Q', R' with $\Delta \vdash P \Rightarrow_{\beta\delta} P'$, $\Delta \vdash Q \Rightarrow_{\beta\delta} Q'$ and $\Delta \vdash R \Rightarrow_{\beta\delta} R'$. By induction, $\Delta \vdash Q' \Rightarrow_{\beta\delta} Q^{\Delta}$ and $\Delta \vdash R' \Rightarrow_{\beta\delta} R^{\Delta}$. Therefore $\Delta \vdash N \Rightarrow_{\beta\delta} Q^{\Delta}[x{:=}R^{\Delta}]$.
  - $N \equiv Q'[x{:=}R']$ for Q', R' with $\Delta \vdash Q \Rightarrow_{\beta\delta} Q'$ and $\Delta \vdash R \Rightarrow_{\beta\delta} R'$. By induction, $\Delta \vdash Q' \Rightarrow_{\beta\delta} Q^{\Delta}$ and $\Delta \vdash R' \Rightarrow_{\beta\delta} R^{\Delta}$. By Lemma 5.17.4, $\Delta \vdash Q'[x{:=}R'] \Rightarrow_{\beta\delta} Q^{\Delta}[x{:=}R^{\Delta}]$.
- $M \equiv bM_1 \cdots M_n$, $\Delta \equiv \Delta_1, b{:=}T{:}\bigotimes_{i=1}^{n} x_i{:}A_i.U, \Delta_2$. Distinguish:
  - $N \equiv bM_1' \cdots M_n'$ for M_i' with $\Delta \vdash M_i \Rightarrow_{\beta\delta} M_i'$. By induction, we have $\Delta \vdash M_i' \Rightarrow_{\beta\delta} M_i^{\Delta}$. By the Definition Lemma, T is typable in a context $\Delta_1;\Delta$, so by the Free Variable Lemma, $\text{FC}(T) \subseteq \text{DOM}(\Delta_1)$. By Lemma 5.20, $\Delta_1 \vdash T \Rightarrow_{\beta\delta} T^{\Delta_1}$. By Lemma 5.18, $\Delta \vdash T \Rightarrow_{\beta\delta} T^{\Delta_1}$. Hence $\Delta \vdash N \Rightarrow_{\beta\delta} T^{\Delta_1}[x_1, \ldots, x_n{:=}M_1^{\Delta}, \ldots, M_n^{\Delta}]$.
  - $N \equiv T'[x_1, \ldots, x_n{:=}M_1', \ldots, M_n']$ for a T' with $\Delta \vdash T \Rightarrow_{\beta\delta} T'$ and for M_i' with $\Delta \vdash M_i \Rightarrow_{\beta\delta} M_i'$. By the Definition Lemma, T is typable in $\Delta_1; x_1{:}A_1, \ldots, x_N{:}A_n$, so by the Free Variable Lemma, $\text{FC}(T) \subseteq \text{DOM}(\Delta_1)$. By Lemma 5.18, $\Delta_1 \vdash T \Rightarrow_{\beta\delta} T'$. By the induction hypothesis on T, $\Delta_1 \vdash T' \Rightarrow_{\beta\delta} T^{\Delta_1}$. As $\Delta_1 \vdash T \Rightarrow_{\beta\delta} T'$, $\text{FC}(T') \subseteq \text{DOM}(\Delta_1)$, so by Lemma 5.18, $\Delta \vdash T' \Rightarrow_{\beta\delta} T^{\Delta_1}$.
    By the induction hypothesis, also $\Delta \vdash M_i' \Rightarrow_{\beta\delta} M_i^{\Delta}$.
    By a repeated application of Lemma 5.17.4, we find that[5]

$$\Delta \vdash T'[x_1, \ldots, x_n{:=}M_1', \ldots M_n'] \Rightarrow_{\beta\delta} T^{\Delta_1}[x_1, \ldots, x_n{:=}M_1^{\Delta}, \ldots, M_n^{\Delta}].$$

□

Corollary 5.22 (Diamond Property for $\Rightarrow_{\beta\delta}$) *Let Δ be the left part of a context in which M is typable. Assume $\Delta \vdash M \Rightarrow_{\beta\delta} N_1$ and $\Delta \vdash M \Rightarrow_{\beta\delta} N_2$. Then there is P such that $\Delta \vdash N_1 \Rightarrow_{\beta\delta} P$ and $\Delta \vdash N_2 \Rightarrow_{\beta\delta} P$.*

PROOF: Immediately from the theorem above: Take $P \equiv M^{\Delta}$. □

Corollary 5.23 (Church-Rosser property for $\Rightarrow_{\beta\delta}$) *Let Δ be the left part of a context in which M is typable. If $\Delta \vdash M \twoheadrightarrow_{\beta\delta} N_1$ and $\Delta \vdash M \twoheadrightarrow_{\beta\delta} N_2$ then there is P such that $\Delta \vdash N_1 \twoheadrightarrow_{\beta\delta} P$ and $\Delta \vdash N_2 \twoheadrightarrow_{\beta\delta} P$.*

PROOF: Directly from Corollary 5.22. □

[5] We must remark that

$$T'[x_1, \ldots, x_n{:=}M_1', \ldots, M_n'] \equiv T'[x_1{:=}M_1'] \cdots [x_n{:=}M_n']$$

and

$$T^{\Delta_1}[x_1, \ldots, x_n{:=}M_1^{\Delta}, \ldots, M_n^{\Delta}] \equiv T^{\Delta_1}[x_1{:=}M_1^{\Delta}] \cdots [x_n{:=}M_n^{\Delta}].$$

This is correct as we can assume that the x_i do not occur in the M_i' and M_i^{Δ}.

5c Subject Reduction

Lemma 5.24 (Subject Reduction) *If* $\Delta;\Gamma \vdash A : B$ *and* $A \to_\beta A'$ *then* $\Delta;\Gamma \vdash A' : B$.

PROOF: The proof is as in [3]. □

Subject Reduction also holds for the reduction relation $\to_\delta$:

Lemma 5.25 (Subject Reduction for $\to_\delta$) *If* $\Delta;\Gamma \vdash A : B$ *and* $A \to_\delta A'$ *then* $\Delta;\Gamma \vdash A' : B$.

PROOF: Following the line of [3], we define $\Delta;\Gamma \to_\delta \Delta;\Gamma'$ if $\Gamma \equiv \Gamma_1, x{:}A, \Gamma_2$, and $\Gamma' \equiv \Gamma_1, x{:}A', \Gamma_2$, and $\Delta \vdash A \to_\delta A'$. We define $\Delta;\Gamma \to_\delta \Delta';\Gamma$ similarly, and we simultaneously prove

$$\begin{aligned}
\Delta;\Gamma \vdash A{:}B \text{ and } \Delta \vdash A \to_\delta A' &\Rightarrow \Delta;\Gamma \vdash A'{:}B\\
\Delta;\Gamma \vdash A{:}B \text{ and } \Delta;\Gamma \to_\delta \Delta';\Gamma &\Rightarrow \Delta';\Gamma \vdash A{:}B\\
\Delta;\Gamma \vdash A{:}B \text{ and } \Delta;\Gamma \to_\delta \Delta;\Gamma' &\Rightarrow \Delta;\Gamma' \vdash A{:}B
\end{aligned}$$

using induction on the derivation of $\Delta;\Gamma \vdash A{:}B$.
We only treat the case in which the last applied rule is the application rule, and only prove the first of the three statements for this case.
We write $A[x_i{:=}B_i]_{i=m}^n$ as a shorthand for $A[x_m{:=}B_m][x_{m+1}{:=}B_{m+1}]\cdots[x_n{:=}B_n]$.
We can assume that

$$\Delta \equiv \Delta_1, b{:=}T{:}\bigotimes_{i=1}^n x_i{:}A_i.B, \Delta_2 \tag{1}$$

with $B \not\equiv \otimes y{:}B_1.B_2$, and that the conclusion of the application rule is

$$\Delta;\Gamma \vdash bM_1\cdots M_n : K_n \tag{2}$$

and therefore

$$\Delta \vdash bM_1\cdots M_n \to_\delta T[x_i{:=}M_i]_{i=1}^n.$$

We repeatedly apply the Generation Lemma, starting with (1), thus obtaining $K_n, K_{n-1}, \ldots, K_1$, $K'_n, K'_{n-1}, \ldots, K_1$, $L_n, L_{n-1}, \ldots, L_1$ such that

$$\Delta;\Gamma \vdash bM_1\cdots M_{i-1} : (\otimes x_i{:}L_i.K'_i) \tag{3}$$

$$\Delta;\Gamma \vdash M_i : L_i \tag{4}$$

$$K_i =_{\beta\delta} K'_i[x_i{:=}M_i] \tag{5}$$

$$K_{i-1} =_{\beta\delta} \otimes x_i{:}L_i.K'_i \tag{6}$$

We end with $\Delta;\Gamma \vdash b : (\otimes x_1{:}L_i.K'_1)$, By the Generation Lemma: $\otimes x_1{:}L_1.K'_1 =_{\beta\delta} \bigotimes_{j=1}^n x_j{:}A_j.B$. By the Church-Rosser Theorem we have $L_1 =_{\beta\delta} A_1$ and $K'_1 = \bigotimes_{j=2}^n x_j{:}A_j.B$. Hence

$$\begin{aligned}
\bigotimes x_2{:}L_2.K'_2 &\overset{(6)}{=_{\beta\delta}} K_1\\
&\overset{(5)}{=_{\beta\delta}} \left(\bigotimes_{j=2}^n x_j{:}A_j.B\right)[x_1{:=}M_1]\\
&\equiv \bigotimes_{i=2}^n x_i{:}A_i[x_1{:=}M_1].B[x_1{:=}M_1]
\end{aligned}$$

so by the Church-Rosser Theorem $L_2 =_{\beta\delta} A_2[x_1{:=}M_1]$. Proceeding in this way, we obtain for $i = 1, \ldots, n$:

$$\begin{aligned}
L_i &=_{\beta\delta} A_i[x_j{:=}M_j]_{j=1}^{i-1}\\
K'_i &=_{\beta\delta} \bigotimes_{j=i+1}^n x_j{:}A_j[x_k{:=}M_k]_{k=1}^{i-1}.B[x_k{:=}M_k]_{k=1}^{i-1}\\
K_i &=_{\beta\delta} \bigotimes_{j=i+1}^n x_j{:}A_j[x_k{:=}M_k]_{k=1}^{i}.B[x_k{:=}M_k]_{k=1}^{i}
\end{aligned}$$

In particular, $K_n =_{\beta\delta} B[x_i{:=}M_i]_{i=1}^n$.
By the Definition Lemma on (1) we also have

$$\Delta_1; x_1{:}A_1, \ldots, x_n{:}A_n \vdash T : B, \tag{7}$$

so by the Start Lemma: $\Delta_1; x_1{:}A_1, \ldots, x_{i-1}{:}A_{i-1} \vdash A_i{:}s_i$. This yields:

$$\begin{array}{rl}
\Delta; \Gamma \vdash A_1 : s_1 & \text{(Thinning Lemma)} \\
\Delta; \Gamma, x_1{:}A_1 \text{ is legal} & \text{(Start Rule)} \\
\Delta; \Gamma, x_1{:}A_1 \vdash A_2 : s_2 & \text{(Thinning Lemma)} \\
\Delta; \Gamma, x_1{:}A_1, x_2{:}A_2 \text{ is legal} & \text{(Start Rule)} \\
\vdots & \\
\Delta; \Gamma, x_1{:}A_1, \ldots, x_n{:}A_n \text{ is legal} & \text{(Start Rule)}
\end{array}$$

By applying the Thinning Lemma to (7) we find:

$$\Delta; \Gamma, x_1{:}A_1, \ldots, x_n{:}A_n \vdash T : B.$$

As $\Delta; \Gamma \vdash M_1 : L_1$ and $\Delta; \Gamma \vdash A_1 : s_1$, we have $\Delta; \Gamma \vdash M_1 : A_1$ by the Conversion rule, so by the Substitution Lemma:

$$\begin{array}{rcl}
\Delta; \Gamma, x_2{:}A_2[x_1{:=}M_1], \ldots, x_n{:}A_n[x_1{:=}M_1] & \vdash & T[x_1{:=}M_1] : B[x_1{:=}M_1] \\
\Delta; \Gamma & \vdash & A_2[x_1{:=}M_1] : s_2
\end{array}$$

As $\Delta; \Gamma \vdash M_2 : L_2$ and $A_2[x_1{:=}M_1] =_{\beta\delta} L_2$ we have by conversion $\Delta; \Gamma \vdash M_2 : A_2[x_1{:=}M_1]$, and again by the Substitution Lemma:

$$\begin{array}{rcl}
\Delta; \Gamma, x_3{:}A_3[x_i{:=}M_i]_{i=1}^2, \ldots, x_n{:}A_n[x_i{:=}M_i]_{i=1}^2 & \vdash & T[x_i{:=}M_i]_{i=1}^2 : B[x_i{:=}M_i]_{i=1}^2 \\
\Delta; \Gamma & \vdash & A_3[x_1{:=}M_1][x_2{:=}M_2] : s_3
\end{array}$$

Proceeding in this way we finally find

$$\Delta; \Gamma \vdash T[x_i{:=}M_i]_{i=1}^n : B[x_i{:=}M_i]_{i=1}^n.$$

As $\Delta; \Gamma \vdash bM_1 \cdots M_n : K_n$ we have $\Delta; \Gamma \vdash K_n : s$ by Lemma 5.10. Now use the Conversion Rule and the fact that $K_n =_{\beta\delta} B[x_i{:=}M_i]_{i=1}^n$. □

The Subject Reduction Theorem for $\rightarrow_\delta$ is used to prove:

Lemma 5.26 *Assume $s \in \mathcal{S}$ and $\Delta; \Gamma \vdash M : N$. Then $M =_{\beta\delta} s \Rightarrow M \equiv s$ and $N =_{\beta\delta} s \Rightarrow N \equiv s$.*

PROOF: First assume $s \in \{\Box, \triangle\}$. If $M =_{\beta\delta} s$ then by Church-Rosser $M \twoheadrightarrow_{\beta\delta} s$, so by Subject Reduction $\Delta; \Gamma \vdash s : N$, contradicting the Generation Lemma. If $N =_{\beta\delta} s$ and $N \not\equiv s$ then we have by Lemma 5.10 that $\Delta; \Gamma \vdash N : P$ for some P, so again $\Delta; \Gamma \vdash s : P$, in contradiction with the Generation Lemma.
Now assume $s \equiv *$, and $M =_{\beta\delta} s$. Again by Church-Rosser, $M \twoheadrightarrow_{\beta\delta} *$, say $M \rightarrow_{\beta\delta} \ldots \rightarrow_{\beta\delta} M' \rightarrow_{\beta\delta} *$. By Subject Reduction, $\Delta; \Gamma \vdash M' : N$ and $\Delta; \Gamma \vdash * : N$. By the Generation Lemma $N =_{\beta\delta} \Box$, so $N \equiv \Box$. Distinguish:

- $M' \equiv (\lambda x{:}A.B)C$ and $* \equiv B[x{:=}C]$.
  By the Generation Lemma there is B' such that $B'[x{:=}C] =_{\beta\delta} \Box$ (hence $B'[x{:=}C] \equiv \Box$), $\Delta; \Gamma \vdash (\lambda x{:}A.B) : (\pi x{:}A.B')$ and $\Delta; \Gamma \vdash C : A$.
  $C \equiv \Box$ contradicts $\Delta; \Gamma \vdash C : A$, so $B' \equiv \Box$.
  By Lemma 5.11 $\Delta; \Gamma \vdash (\pi x{:}A.\Box) : *$, so by the Generation Lemma $\Delta; \Gamma, x{:}A \vdash \Box : *$, contradiction.
- $M' \equiv bM_1 \cdots M_n$ and $bM_1 \cdots M_n \rightarrow_\delta T[x_i{:=}M_i]_{i=1}^n$. The argument is similar as in the case $M' \equiv (\lambda x{:}A.B)C$.

If $s \equiv *$ and $N =_{\beta\delta} s$ then by Lemma 5.10 $N \equiv s$ (and we are done) or $\Delta; \Gamma \vdash N : s'$ (which is impossible by the above argument). □

5d Strong Normalization

We prove Strong Normalization for $\beta\delta$-reduction in $\lambda 68$ by mapping a typable term M (in a context $\Delta;\Gamma$) of $\lambda 68$ to a term $|M|_\Delta$ that is typable in a strongly normalizing PTS. The mapping is constructed in such a way that if $M \to_\beta N$, also $|M|_\Delta \to_\beta |N|_\Delta$, and that if $M \to_\delta N$, $|M|_\Delta \twoheadrightarrow_\beta^+ |N|_\Delta$. This last feature requires special attention for the definition of $|b|_\Delta$ when $\Delta \equiv \Delta_1, b{:=}T{:}\bigotimes_{i=1}^n x_i{:}A_i.U, \Delta_2$. Simply defining $|b|_\Delta = \boldsymbol{\lambda}_{i=1}^n x_i{:}|A_i|_{\Delta_1}.|T|_{\Delta_1}$ doesn't give the desired result if $n = 0$: In that case, the δ-reduction $b \to_\delta T$ results in 0 β-reductions in the translation: $|b|_\Delta \equiv |T|_{\Delta_1} \equiv |T|_\Delta$ (all definitions of T are mentioned in Δ_1, so $|T|_\Delta \equiv |T|_{\Delta_1}$).
To enforce at least one β-reduction in this case as well, we define $|b|_\Delta = \mathrm{Id}(\boldsymbol{\lambda}_{i=1}^n x_i{:}|A_i|_{\Delta_1}.|T|_{\Delta_1})$, where Id is the identity operator on the appropriate type.

Definition 5.27 Let Δ be the left part of a legal context and let $M \in \mathcal{T}$. We define $|M|_\Delta$ by induction on the length of Δ and the structure of M.

- $|x|_\Delta \stackrel{\text{def}}{=} x$ for $x \in \mathcal{V}$;
- $|b|_\Delta \stackrel{\text{def}}{=} b$ for all $b \in \mathcal{C} \setminus \textsc{defcons}(\Delta;)$;
- $|b|_\Delta \stackrel{\text{def}}{=} (\lambda b{:}\,(\prod_{i=1}^n x_i{:}|A_i|_{\Delta_1}.|U|_{\Delta_1})\,.b)\,(\boldsymbol{\lambda} i = 1^n x_i{:}\,|A_i|_{\Delta_1}.|T|_{\Delta_1})$
  if $\Delta \equiv \Delta_1, b{:=}T{:}\bigotimes_{i=1}^n x_i{:}A_i.U, \Delta_2$.
- $|s|_\Delta \stackrel{\text{def}}{=} s$ for $s \in \mathcal{S}$;
- $|\lambda x{:}P.Q|_\Delta \stackrel{\text{def}}{=} \lambda x{:}|P|_\Delta.|Q|_\Delta$;
- $|\Pi x{:}P.Q|_\Delta \stackrel{\text{def}}{=} \Pi x{:}|P|_\Delta.|Q|_\Delta$;
- $|\otimes x{:}P.Q|_\Delta \stackrel{\text{def}}{=} \Pi x{:}|P|_\Delta.|Q|_\Delta$;
- $|PQ|_\Delta \stackrel{\text{def}}{=} |P|_\Delta\,|Q|_\Delta$;

The following lemmas are usefull:

Lemma 5.28 *Let Δ be the left part of a legal context and $M \in \mathcal{T}$. Then* $\textsc{fv}(|M|_\Delta) = \textsc{fv}(M)$.

PROOF: The proof is by induction on the definition of $|M|_\Delta$ and is trivial for all cases except the case $M \equiv b$ and $\Delta \equiv \Delta_1, b{:=}T{:}\bigotimes\Gamma.U, \Delta_2$ ($U \not\equiv \otimes y{:}U_1.U_2$).
By the Definition Lemma, T is typable in $\Delta_1;\Gamma$; therefore $\textsc{fv}(T) \subseteq \textsc{dom}(\Gamma)$ (Free Variable Lemma). By the induction hypothesis, $\textsc{fv}(|T|_{\Delta_1}) \subseteq \textsc{dom}(\Gamma)$ and therefore $\textsc{fv}(|c|_\Delta) = \varnothing$. □

Lemma 5.29 *If Δ_1 and Δ_2 are left parts of legal contexts and $\Delta_2 \equiv \Delta_1, \Delta'$ then $|M|_{\Delta_2} \equiv |M|_{\Delta_1}$ for all $M \in \mathcal{T}$ with* $\textsc{fc}(M) \subseteq \textsc{dom}(\Delta_1)$.

PROOF: An easy induction on the definition of $|M|_{\Delta_1}$. □

Lemma 5.30 *Let Δ be left part of a legal context. For all M, N: $|M[x{:=}N]|_\Delta \equiv |M|_\Delta\,[x{:=}|N|_\Delta]$.*

PROOF: By induction on the definition of $|M|_\Delta$. In the case $M \equiv b$ and $b{:=}T{:}U \in \Delta$, use the fact that $\textsc{fv}(|M|_\Delta) = \textsc{fv}(M) = \varnothing$ (Lemma 5.28) and therefore $|M|_\Delta\,[x{:=}|N|_\Delta] \equiv |M|_\Delta \equiv |M[x{:=}N]|_\Delta$. □

The purpose of the definition of $|M|_\Delta$ (and especially the exception that was made for the case $b{:=}T{:}U \in \Delta$) is the following lemma:

Lemma 5.31 *If $\Delta \vdash M \to_{\beta\delta} N$ then $|M|_\Delta \twoheadrightarrow_\beta^+ |N|_\Delta$.*

PROOF: We use induction on the structure of M. We treat a few cases:

- $M \equiv (\lambda x{:}P.Q)R$ and $N \equiv Q[x{:=}R]$.

$$\begin{array}{rcl} |M|_\Delta & \equiv & (\lambda x{:}|P|_\Delta \,.\, |Q|_\Delta)\,|R|_\Delta \\ & \to_\beta & |Q|_\Delta\,[x{:=}|R|_\Delta] \\ & \overset{5.30}{\equiv} & |Q[x{:=}R]|_\Delta \end{array}$$

- $M \equiv bM_1 \cdots M_n$;
  $\Delta \equiv \Delta_1, b{:=}T{:}\bigotimes_{i=1}^n x_i{:}A_i.U, \Delta_2$;
  $N \equiv T[x_1, \ldots, x_n{:=}M_1, \ldots, M_n]$.

$$\begin{array}{rcl} |M|_\Delta & \equiv & \left((\lambda b{:}\left(\prod_{i=1}^n x_i{:}|A_i|_{\Delta_1} \,.\, |U|_{\Delta_1}\right).b)\,\left(\boldsymbol{\lambda} i = 1^n x_i{:}|A_i|_{\Delta_1} \,.\, |T|_{\Delta_1}\right)\right)|M_1|_\Delta \cdots |M_n|_\Delta \\ & \to_\beta & \left(\boldsymbol{\lambda} i = 1^n x_i{:}|A_i|_{\Delta_1} \,.\, |T|_{\Delta_1}\right)|M_1|_\Delta \cdots |M_n|_\Delta \\ & \twoheadrightarrow_\beta & |T|_{\Delta_1}\,[x_i{:=}|M_i|_\Delta]_{i=1}^n \\ & \overset{5.29}{\equiv} & |T|_\Delta\,[x_i{:=}|M_i|_\Delta]_{i=1}^n \\ & \overset{5.30}{\equiv} & |T[x_i{:=}M_i]_{i=1}^n|_\Delta \\ & \equiv & |T[x_1, \ldots, x_n{:=}M_1, \ldots, M_n]|_\Delta \end{array}$$

  At the last equivalence, we must make a remark similar to footnote 5.

□

Let λSN be the PTS over λ-terms with variables from $\mathcal{V} \cup \mathcal{C}$ and sorts from $\mathcal{S}$, and the following rules (we choose the name λSN because this system will help us in showing that $\lambda 68$ is SN):

$$\begin{array}{cc} & (*,*,*) \\ (*,*,\triangle) & (\square,*,\triangle) \\ (*,\square,\triangle) & (\square,\square,\triangle) \\ (*,\triangle,\triangle) & (\square,\triangle,\triangle) \\ (\square,\square,\square) & (\triangle,\triangle,\triangle) \end{array}$$

λSN is contained in the system **ECC** (see [25]). As **ECC** is β-strongly normalizing, also λSN is β-strongly normalizing.

We present a translation of $\lambda 68$-contexts to λSN-contexts:

Definition 5.32 Let $\Delta; \Gamma$ be a legal $\lambda 68$-context.

- We define $|\Delta|$ by induction on the length of Γ_1:
  - $|\varnothing| \overset{\text{def}}{=} \varnothing$;
  - $|\Delta, b{:}U| \overset{\text{def}}{=} |\Delta|\,, b{:}|U|_{\Gamma_1}$;
  - $|\Delta, b{:=}T{:}U| \overset{\text{def}}{=} |\Delta|$.
- If $\Gamma \equiv x_1{:}A_1, \ldots, x_n{:}A_n$ then $|\Delta; \Gamma| \overset{\text{def}}{=} |\Delta|\,, x_1{:}|A_1|_\Delta\,, \ldots, x_n{:}|A_n|_\Delta$.

We see that definitions $b{:=}T{:}U$ in Δ are not translated into $|\Delta|$. This corresponds to the fact that in $|M|_\Delta$, all these definitions are unfolded (replaced by their definiendum).

Now we are able to prove the most important lemma of this subsection:

Lemma 5.33 *If $\Delta; \Gamma \vdash M : N$ then $|\Delta; \Gamma| \vdash |M|_\Delta : |N|_\Delta$ is derivable in λSN.*

PROOF: The proof is by induction on the derivation of $\Delta; \Gamma \vdash M : N$. We treat a few cases:

(Start: Primitive Constants)

$$\frac{\Delta;\Gamma\vdash B:s_1 \qquad \Delta;\vdash \bigotimes\Gamma.B:s_2}{\Delta,b{:}\bigotimes\Gamma.B;\vdash b:\bigotimes\Gamma.B}(s_1=*,\Box)$$

By the induction hypothesis, $|\Delta|\vdash|\bigotimes\Gamma.B|_\Delta:s_2$, so by the Start rule:

$$|\Delta|,b{:}|\bigotimes\Gamma.B|_\Delta\vdash b{:}|\bigotimes\Gamma.B|_\Delta\,.$$

Observe that $|\Delta,b{:}\bigotimes\Gamma.B|\equiv|\Delta|,b{:}|\bigotimes\Gamma.B|_\Delta$, that $|b|_{\Delta,b:\bigotimes\Gamma.B}\equiv b$ and (by Lemma 5.29) $|\bigotimes\Gamma.B|_\Delta\equiv|\bigotimes\Gamma.B|_{\Delta,b:\bigotimes\Gamma.B}$.

(Start: Defined Constants)

$$\frac{\Delta;\Gamma\vdash T:B:s_1 \qquad \Delta;\vdash \bigotimes\Gamma.B:s_2}{\Delta,b{:=}T:\bigotimes\Gamma.B;\vdash b:\bigotimes\Gamma.B}(s_1=*,\Box)$$

This is the only case in which we really have to work. By induction we have

$$|\Delta;|\vdash|\bigotimes\Gamma.B|_\Delta:s_2 \tag{8}$$

so with the weakening rule:

$$|\Delta;|,b{:}|\bigotimes\Gamma.B|_\Delta\vdash|\bigotimes\Gamma.B|_\Delta:s_2 \tag{9}$$

and with rule (s_2,s_2,s_2) applied to (8) and (9):

$$|\Delta;|\vdash(\Pi b{:}|\bigotimes\Gamma.B|_\Delta\,.\,|\bigotimes\Gamma.B|_\Delta):s_2 \tag{10}$$

By (8) and the start rule:

$$|\Delta;|,b{:}|\bigotimes\Gamma.B|_\Delta\vdash b{:}|\bigotimes\Gamma.B|_\Delta \tag{11}$$

so with the λ-abstraction rule applied to (10) and (11):

$$|\Delta;|\vdash(\lambda b{:}|\bigotimes\Gamma.B|_\Delta\,.b):(\Pi b{:}|\bigotimes\Gamma.B|_\Delta\,.\,|\bigotimes\Gamma.B|_\Delta) \tag{12}$$

By induction, we also have $|\Delta;\Gamma|\vdash|T|_\Delta:|B|_\Delta$, so (write $\Gamma\equiv x_1{:}A_1,\ldots,x_n{:}A_n$):

$$|\Delta|,x_1{:}|A_1|_\Delta,\ldots,x_n{:}|A_n|_\Delta\vdash|T|_\Delta:|B|_\Delta \tag{13}$$

and by repeatedly applying the λ-rule on (13) and using the fact that, by the Induction Hypothesis, the types $\prod_{j=i}^n x_j{:}|A_j|_\Delta\,.\,|B|_\Delta$ are all typable, we find:

$$|\Delta;|\vdash\boldsymbol{\lambda}i=1^n x_i{:}|A_i|_\Delta\,.\,|T|_\Delta:\prod_{i=1}^n x_i{:}|A_i|_\Delta\,.\,|B|_\Delta \tag{14}$$

Notice that $\prod_{i=1}^n x_i{:}|A_i|_\Delta\,.\,|B|_\Delta\equiv|\bigotimes\Gamma.B|_\Delta$ and use application on (12) and (14):

$$|\Delta;|\vdash(\lambda b{:}|\bigotimes\Gamma.B|_\Delta\,.b)(\boldsymbol{\lambda}i=1^n x_i{:}|A_i|_\Delta\,.\,|T|_\Delta):|\bigotimes\Gamma.B|_\Delta$$

(as $\mathrm{FV}(|\bigotimes\Gamma.B|_\Delta)=\varnothing$, the usual substitution after the use of the application rule has no effect) and we are done.

(Application 1) (the Appication 2-case is similar)

$$\frac{\Delta;\Gamma\vdash\Pi x{:}A.B:* \qquad \Delta;\Gamma,x{:}A\vdash F:B}{\Delta;\Gamma\vdash(\lambda x{:}A.F):(\Pi x{:}A.B)}$$

By the induction hypothesis, we have $|\Delta;\Gamma|\vdash|M|_\Delta:\Pi x{:}|A|_\Delta\,.\,|B|_\Delta$ and $|\Delta;\Gamma|\vdash|N|_\Delta:|A|_\Delta$. The application rule gives

$$|\Delta;\Gamma|\vdash|M|_\Delta\,|N|_\Delta:|B|_\Delta\,[x{:=}|A|_\Delta]$$

Use the definition of $|MN|_\Delta$ and Lemma 5.30 to obtain

$$|\Delta;\Gamma|\vdash|MN|_\Delta:|B[x{:=}A]|_\Delta\,.$$

□

Corollary 5.34 (Strong Normalization) *$\lambda 68$ is $\beta\delta$-strongly normalizing.*

PROOF: Immediately from Lemma 5.31 and Lemma 5.33. □

5e The formal relation between AUT-68 and λ68

Theorem 5.35 *Let $\mathfrak{B}$ be a correct book and Γ a correct context with respect to $\mathfrak{B}$.*

- $\overline{\mathfrak{B}};\overline{\Gamma}$ *is legal;*
- *If* $\mathfrak{B},\Gamma \vdash \Sigma : \Omega$ *then* $\overline{\mathfrak{B}};\overline{\Gamma} \vdash \overline{\Sigma} : \overline{\Omega}$.

PROOF: We prove both statements simultaneously, using induction on the number of lines in $\mathfrak{B}$.

- $\mathfrak{B}$ is empty. All cases can be checked manually. This work is left to the reader.
- Assume, the lemma has been proved for all books with at most n lines, and assume $\mathfrak{B}$ has $n+1$ lines. Let $\mathfrak{B}'$ be the book consisting of the first n lines of $\mathfrak{B}$. Focus on the last line of $\mathfrak{B}$.
  - This line is of the form $(\Gamma', x, —, \Xi)$. Notice that $\overline{\mathfrak{B}'} \equiv \overline{\mathfrak{B}}$.
    If Γ is a correct context with respect to $\mathfrak{B}$, then either Γ is correct with respect to $\mathfrak{B}'$ (hence $\overline{\mathfrak{B}};\overline{\Gamma}$ is legal by the induction hypothesis) or $\Gamma \equiv \Gamma', x{:}\Xi$.
    In this last case: Notice that either $\Xi \equiv \mathtt{type}$ (then notice that $\overline{\mathfrak{B}};\overline{\Gamma'}$ is legal, hence $\overline{\mathfrak{B}};\overline{\Gamma'} \vdash \overline{\Xi}{:}\Box$), or Ξ is a correct expression of type `type` with respect to $\mathfrak{B}'$ and Γ' (and then by the induction hypothesis $\overline{\mathfrak{B}};\overline{\Gamma'} \vdash \overline{\Xi}{:}*$). By the start rule for variables we can deduce: $\overline{\mathfrak{B}};\overline{\Gamma'}, x{:}\overline{\Xi} \vdash x{:}\overline{\Xi}$, and we see that $\overline{\mathfrak{B}};\overline{\Gamma}$ is legal.
    Now assume $\mathfrak{B};\Gamma \vdash \Sigma : \Omega$. If $\Gamma \not\equiv \Gamma', x{:}\Xi$, then $\mathfrak{B}';\Gamma \vdash \Sigma : \Omega$ and we can use the induction hypothesis to obtain $\overline{\mathfrak{B}};\overline{\Gamma} \vdash \overline{\Sigma}{:}\overline{\Omega}$. If $\Gamma \equiv \Gamma', x{:}\Xi$, use a straightforward induction on the structure of Σ and the Start Lemma.
  - This line is of the form $(\Gamma', b, \mathrm{PN}, \Xi)$. Now $\overline{\mathfrak{B}} \equiv \overline{\mathfrak{B}'}, b{:}\bigotimes\overline{\Gamma'}.\overline{\Xi}$.Notice that $\Xi \equiv \mathtt{type}$ or Ξ is a correct expression of type `type` with respect to $\mathfrak{B}', \Gamma'$, hence: $\overline{\mathfrak{B}'};\overline{\Gamma'} \vdash \overline{\Xi}{:}s$ for $s = *$ or $s = \Box$.
    As all the types in $\overline{\Gamma'}$ have sort $*$ or $\Box$ (by the Generation Lemma), we can use the $\otimes$-formation rules to deduce $\overline{\mathfrak{B}'};\vdash \bigotimes\overline{\Gamma'}.\overline{\Xi}{:}\Delta$, and introduce the constant b: $\overline{\mathfrak{B}};\vdash b{:}\bigotimes\overline{\Gamma'}.\overline{\Xi}$. Using induction on the length of Γ and the Thinning Lemma, we can prove that $\overline{\mathfrak{B}};\overline{\Gamma}$ is legal.
    $\overline{\mathfrak{B}};\overline{\Gamma} \vdash \overline{\Sigma}{:}\overline{\Omega}$ is, as above, shown by induction on the structure of Σ.
  - This line is of the form $(\Gamma', b, \Xi_1, \Xi_2)$. The proof is similar as in the case $(\Gamma', b, \mathrm{PN}, \Xi)$.

□

It is possible to prove a conservativity theorem (in the style: If $\overline{\mathfrak{B}};\overline{\Gamma} \vdash \overline{\Sigma} : \overline{\Omega}$, then Σ is a correct expression of type Ω with respect to $\mathfrak{B}$ and Γ), but we want to prove that all the typable terms of λ68 have some interpretation in AUT-68, and not only the terms that have an equivalent in AUT-68. We have to distinguish 6 different cases, and the interpretation of these 6 cases is given after the proof of the theorem.

Theorem 5.36 *Assume $\Delta;\Gamma \vdash M : N$. Then there is a correct book $\mathfrak{B}$, and a context Γ' correct with respect to $\mathfrak{B}$ such that $\overline{\mathfrak{B}},\overline{\Gamma'} \equiv \Delta;\Gamma$. Moreover,*

1. *If $N \equiv \Box$ then $M \equiv *$;*
2. *If $\Delta;\Gamma \vdash N : \Box$ then $N \equiv *$ and there is $\Omega \in \mathcal{E}_{68}$ such that $\overline{\Omega} \equiv M$ and $\mathfrak{B};\Gamma' \vdash \Omega : \mathtt{type}$;*
3. *If $N \equiv \Delta$ then there is $\Gamma'' \equiv x_1{:}\Sigma_1, \ldots, x_n{:}\Sigma_n$ and $\Omega \in \mathcal{E}^+_{68}$ such that*
   - *Γ', Γ'' is correct with respect to $\mathfrak{B}$;*
   - *$M \equiv \prod\overline{\Gamma''}.\overline{\Omega}$;*
   - *$\Omega \equiv \mathtt{type}$ or $\mathfrak{B};\Gamma' \vdash \Omega : \mathtt{type}$.*

4. *If* $\Delta;\Gamma \vdash N : \Delta$ *then there are* $b \in C$ *and* $\Sigma_1,\ldots,\Sigma_n \in \mathcal{E}_{68}$ *such that* $M \equiv b\overline{\Sigma_1}\cdots\overline{\Sigma_n}$. *Moreover,* $\mathfrak{B}$ *contains a line*

$$(x_1{:}\Omega_1,\ldots,x_m{:}\Omega_m; b; \Xi_1; \Xi_2)$$

such that

- $m > n$;
- $\mathfrak{B};\Gamma' \vdash \Sigma_i{:}\Omega_i[x_1,\ldots,x_{i-1}{:=}\Sigma_1,\ldots,\Sigma_{i-1}]$ $(1 \leq i \leq n)$;
- $N \equiv \bigotimes_{i=n+1}^{m} x_i{:}\overline{\Omega_i}.\overline{\Xi_2}[x_1,\ldots,x_n{:=}\overline{\Sigma_1},\ldots,\overline{\Sigma_n}]$;

5. $N \equiv *$. *Then there is* $\Omega \in \mathcal{E}_{68}$ *such that* $\overline{\Omega} \equiv M$ *and* $\mathfrak{B};\Gamma' \vdash \Omega : \mathtt{type}$;

6. $\Delta;\Gamma \vdash N : *$. *Then there are* $\Sigma,\Omega \in \mathcal{E}_{68}$ *such that* $\overline{\Sigma} \equiv M$ *and* $\overline{\Omega} \equiv N$, *and* $\mathfrak{B};\Gamma' \vdash \Sigma : \Omega$, *and* $\mathfrak{B};\Gamma' \vdash \Omega : \mathtt{type}$.

PROOF: We use induction on the derivation of $\Delta;\Gamma \vdash M : N$. We only treat a few cases:

Start: Defined Constants

$$\frac{\Delta;\Gamma \vdash B : s_1 \qquad \Delta;\Gamma \vdash T : B \qquad \Delta;\vdash \bigotimes\Gamma.B : s_2}{\Delta; b{:=}T : \bigotimes\Gamma.B;\vdash b : \bigotimes\Gamma.B}\, s_1 \equiv *, \Box$$

Determine $\mathfrak{B}$ and Γ' such that $\overline{\mathfrak{B}};\overline{\Gamma'} \equiv \Delta;\Gamma$ (we can assume that the induction hypothesis on the three premises give the same book $\mathfrak{B}$). Assume $s_1 \equiv *$ (the case $s_1 \equiv \Box$ is similar). Determine $\Sigma,\Omega \in \mathcal{E}_{68}$ such that $\overline{\Sigma} \equiv T$ and $\overline{\Omega} \equiv B$, and $\mathfrak{B};\Gamma' \vdash \Sigma : \Omega$. Obtain a book $\mathfrak{B}'$ by adding a line

$$(\Gamma'; b; \Sigma; \Omega)$$

to $\mathfrak{B}$. Notice that

$$\begin{aligned}\overline{\mathfrak{B}'} &\equiv \overline{\mathfrak{B}}, b{:=}\overline{\Sigma} : \bigotimes\overline{\Gamma'}.\overline{\Omega}\\ &\equiv \Delta, b{:=}T : \bigotimes\Gamma.B\end{aligned}$$

If $\Gamma \equiv \varnothing$ then $\bigotimes\Gamma.B \equiv B$ and we are in case 6. Notice that $\mathfrak{B}';\varnothing \vdash b() : \Omega$.
If $\Gamma \not\equiv \varnothing$ then $\overline{\mathfrak{B}'};\vdash \bigotimes\Gamma.B : \Delta$ and we are in case 3. We can take $n = 0$ and $\Gamma'' \equiv \Gamma'$; we can take $\Xi_1 \equiv \Sigma$ and $\Xi_2 \equiv \Omega$.

Application 2

$$\frac{\Delta;\Gamma \vdash M : \otimes x{:}A.B \qquad \Delta;\Gamma \vdash N : A}{\Delta;\Gamma \vdash MN : B[x{:=}N]}$$

Determine $\mathfrak{B}$ and Γ' such that $\overline{\mathfrak{B}};\overline{\Gamma'} \equiv \Delta;\Gamma$ (again we can assume that the applications of the induction hypothesis on both premises result in the same book $\mathfrak{B}$).
Notice that $\Delta;\Gamma \vdash \otimes x{:}A.B : \Delta$. Determine, with the induction hypothesis, $\Sigma_1,\ldots,\Sigma_n \in \mathcal{E}_{68}$ and a line

$$(x_1{:}\Omega_1,\ldots,x_m{:}\Omega_m; b; \Xi_1; \Xi_2)$$

in $\mathfrak{B}$ such that

- $m > n$;
- $M \equiv b\overline{\Sigma_1}\cdots\overline{\Sigma_n}$;
- $\mathfrak{B};\Gamma' \vdash \Sigma_i : \Omega_i[x_j{:=}\Sigma_j]_{j=1}^{i-1}$;
- $\otimes x{:}A.B \equiv \bigotimes_{i=n+1}^{m} x_i{:}\overline{\Omega_i}.\overline{\Xi_2}[x_j{:=}\overline{\Sigma_j}]_{j=1}^{n}$.

Observe: $A \equiv \overline{\Omega_{n+1}}[x_j{:=}\overline{\Sigma_j}]_{j=1}^n$. As $\mathfrak{B}; \Gamma' \vdash \Omega_{n+1} : \mathtt{type}$ or $\Omega_{n+1} \equiv \mathtt{type}$, we have $\Delta; \Gamma \vdash \overline{\Omega_{n+1}} : s$ for an $s \in \{*, \Box\}$, and by Substitution Lemma and Transitivity Lemma we have $\Delta; \Gamma \vdash \overline{\Omega_{n+1}}[x_j{:=}\overline{\Sigma_j}]_{j=1}^n : s$, hence $\Delta; \Gamma \vdash A : s$.
With the induction hypothesis we determine $\Sigma \in \mathcal{E}_{68}$ such that $\mathfrak{B}; \Gamma' \vdash \Sigma : \Omega_{n+1}[x_j{:=}\Sigma_j]_{j=1}^n$. and $N \equiv \overline{\Sigma}$. We now treat the most important of the cases 1–6:

4. The only thing that doesn't directly follow from the results above is $m > n+1$. Assume, for the sake of the argument, $m = n+1$. Then $B[x{:=}N] \equiv \overline{\Xi_2}[x_j{:=}\overline{\Sigma_j}]_{j=1}^{n+1}$. As $\Delta; \Gamma \vdash B[x{:=}N] : \Delta$, $\overline{\Xi_2}[x_j{:=}\overline{\Sigma_j}]_{j=1}^{n+1}$ is of the form $\otimes x : P.Q$, which is impossible.
6. Notice: $B[x{:=}N] \equiv \left(\bigotimes_{j=n+2}^m x_i{:}\overline{\Omega_i}.\overline{\Xi_2}\right)[x_j{:=}\overline{\Sigma_j}]_{j=1}^{n+1}$.
As $\Delta; \Gamma \vdash B[x{:=}N] : *$, $B[x{:=}N]$ cannot be of the form $\otimes y{:}P.Q$, and therefore $m = n+1$. Therefore, $\mathfrak{B}; \Gamma' \vdash b(\Sigma_1, \ldots, \Sigma_{n+1}) : \Xi_2[x_1, \ldots, x_{n+1}{:=}\Sigma_1, \ldots, \Sigma_{n+1}]$, and this is what we wanted to prove.

□

Remark 5.37 We give some explanation to the different cases mentioned in the formulation of Theorem 5.36.

- The cases $N \equiv \Box$ and $\Delta; \Gamma \vdash N : \Box$ indicate that there are no other terms in $\lambda 68$ than $*$ itself at the same level as $*$. This corresponds to the fact that **type** is the only "top-expression" in AUT-68.
- The cases $N \equiv *$ and $\Delta; \Gamma \vdash N : *$ give a precise correspondence between expressions of AUT-68 and terms of $\lambda 68$: If $M : N$ in $\lambda 68$ then there are expressions Σ, Ω in AUT-68 such that $\Sigma : \Omega$ and $\overline{\Sigma} \equiv M$; $\overline{\Omega} \equiv N$.
- The cases $N \equiv \Delta$ and $\Delta; \Gamma \vdash N : \Delta$ cover terms that do not have an equivalent in AUT-68 but are necessary in $\lambda 68$ to form terms that have equivalents in AUT-68. More specific, this concerns terms of the form $\bigotimes_{i=1}^n x_i{:}A_i.B$ (which are needed to introduce constants) and terms of the form $bM_1 \cdots M_n$, where b is a constant of type $\bigotimes_{i=1}^m x_i{:}A_i.B$ for certain $m > n$ (which are needed to construct $\lambda 68$-equivalents of expressions of the form $b(\Sigma_1, \ldots, \Sigma_m)$).

We conclude that $\lambda 68$ and AUT-68 coincide as much as possible, and that the terms in $\lambda 68$ that do not have an equivalent in AUT-68 can be traced easily (these are the terms of type Δ and the terms of a type $M : \Delta$, and the sorts $\Box$ and Δ, which are needed to give a type to $*$ and to the Π-types).
Notice that the alternative definition of δ-reduction in $\lambda 68$, discussed at the end of Subsection 4c, would introduce more terms in $\lambda 68$ without an equivalent in AUT-68, namely terms of the form $\lambda x_1{:}A_1. \cdots \lambda x_n : A_n.B$.

6 Related Works

Recently, various type systems with definitions in PTS-style have been proposed by, among others, Bloo, Kamareddine and Nederpelt ([6, 21]) and by Severi and Poll ([29]). The presentation of AUT-68 in the PTS-like system $\lambda 68$ makes a good comparison between these systems and the definition system in AUT-68 possible.

6a Comparison with the DPTSs of Severi and Poll

In [29], Severi and Poll present an extension of PTSs with definitions, thus obtaining Pure Type Systems with Definitions (DPTSs). They extend the usual PTS-rules with the following D-rules:

$$\text{(D-start)} \qquad \frac{\Gamma \vdash a : A}{\Gamma, x{=}a{:}A \vdash x : A}$$

$$\text{(D-weak)} \quad \frac{\Gamma \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma, x{=}a{:}A \vdash b : B}$$

$$\text{(D-form)} \quad \frac{\Gamma, x{=}a{:}A \vdash B : s}{\Gamma \vdash (x{=}a{:}A \text{ in } B) : s}$$

$$\text{(D-intro)} \quad \frac{\Gamma, x{=}a{:}A \vdash b : B \qquad \Gamma \vdash (x{=}a{:}A \text{ in } B) : s}{\Gamma \vdash (x{=}a{:}A \text{ in } b) : (x{=}a{:}A \text{ in } B)}$$

$$\text{(D-conv)} \quad \frac{\Gamma \vdash b : B \qquad \Gamma \vdash B' : s \qquad \Gamma \vdash B =_{\mathrm{D}} B'}{\Gamma \vdash b : B'}$$

where D-reduction is defined by the following rules:

$$\Gamma_1, x{=}a{:}A, \Gamma_2 \vdash x \to_{\mathrm{D}} a$$

$$\Gamma \vdash (x{=}a{:}A \text{ in } b) \to_{\mathrm{D}} b \qquad (x \notin \mathrm{FV}(b))$$

$$\frac{\Gamma, x{=}a{:}A \vdash b \to_{\mathrm{D}} b'}{\Gamma \vdash (x{=}a{:}A \text{ in } b) \to_{\mathrm{D}} (x{=}a{:}A \text{ in } b')}$$

and the usual compatibility rules. As we see, there is an extra class of terms in DPTSs, namely those of the form $x{=}a{:}A$ in b.

When regarding both systems we find that

- In DPTSs, definitions do not only occur in a context, but may also occur in terms. Moreover, definitions may disappear from contexts when they are introduced in terms (e.g. the D-form and the D-intro rules, and the last of the three D-reduction rules), and definitions may disappear from terms when the definiendum does not occur in that term (the middle D-reduction rule).
  This gives definitions a more temporarily character: we can use them as long as needed, and when we do not need them any more, we can remove them from the context.
  Definitions can also play a more local role: A definition that is needed in only one term can be imported into that term while it is not necessary to carry it around in the (global) context, as well.
  This temporary and local behaviour of definitions is not present in AUTOMATH.

- Due to the fact that definitions can also play a local role, D-reduction can also unfold definitions which are not present in the (global) context, but which are given within the term. For example, we have $\alpha{:}* \vdash (id{=}\lambda x{:}\alpha.x \text{ in } id) \to_{\mathrm{D}} \lambda x{:}\alpha.x$, though there is no definition of id in the context $\alpha{:}*$.
  Again, this is not possible in AUTOMATH.

- The start rule for definitions in DPTSs,

  $$\frac{\Gamma \vdash T : B}{\Gamma, x{=}T{:}B \vdash x : B}$$

  does not require $\Gamma \vdash B : s$ for a sort s. In $\lambda 68$ we have the rule (St: def):

  $$\frac{\Delta; \Gamma \vdash T : B : s_1 \qquad \Delta; \vdash \bigotimes \Gamma.B : s_2}{\Delta, x{:=}T{:}\bigotimes \Gamma.B; \vdash x : \bigotimes \Gamma.B}(s_1 = *, \Box)$$

  where we see that both B and $\bigotimes \Gamma.B$ need to be of a certain sort (and B must be of sort $*$ or $\Box$).

- The start rules for definitions in DPTSs and in $\lambda 68$ differ in another point, too, namely the type of definiens and definiendum. In DPTSs they have the same type (in the notation of

the previous paragraph: B), while in $\lambda 68$ the definiens T has type B and the definiendum x has type $\bigotimes\Gamma_2.B$. This topic has already been discussed when we introduced the definition mechanism of $\lambda 68$ in Section 4c.

- D-reduction differs from δ-reduction, also when only global definitions are taken into account. For instance, δ-reduction is *substitutive*, i.e. if $\Delta \vdash A \to_\delta A'$ then $\Delta \vdash A[x{:=}b] \to_\delta A'[x{:=}b]$ (proof: induction on the structure of A). D-reduction is not substitutive: take $\Gamma \equiv \alpha{:}*, y{=}\alpha{:}*$. Then $\Gamma \vdash y \to_D \alpha$, but $\Gamma \not\vdash y[\alpha{:=}M] \to_D \alpha[\alpha{:=}M]$.
  In $\lambda 68$, this example would look as follows. Take $\Delta \equiv y{:=}\alpha{:}\otimes\alpha{:}*.*$. Then $\Delta \vdash y\alpha \to_\delta \alpha$ and $\Delta \vdash y\alpha[\alpha{:=}M] \to_\delta \alpha[\alpha{:=}M]$.
  Substitutivity for $\to_D$ is lost, because unfolding a definition by D-reduction may introduce new free variables in the term. In AUTOMATH, all free variables in the definiens must be added as parameters to the definiendum. In $\lambda 68$ this is visible in the Start and Weakening rules for defined constants: the right hand side Γ of the context $\Delta;\Gamma$ that is used to type the definiens T in these rules, serves as list of parameters in the definiendum. When an AUTOMATH-definition is unfolded, the free variables occurring in the definiens are replaced by the parameters.

- We see that the definition of y in $\lambda 68$ in the example above is more general than in the corresponding DPTS situation. In the DPTS-example, y D-reduces to one, fixed term x. In the $\lambda 68$ version, yM is defined for any (typable) term M. To do something similar in DPTSs, one needs to define y as $\lambda\alpha{:}*.\alpha$. In particular, one needs to type the term $\lambda\alpha{:}*.\alpha$, which involves the use of rule $(\Box,\Box)$, so the use of a higher type system. One could say that AUTOMATH and $\lambda 68$ use an implicit λ-abstraction where DPTSs need an explicit λ-abstraction. On this point, AUTOMATH and $\lambda 68$ are more flexible than DPTSs.

6b Comparison with systems of Bloo, Kamareddine and Nederpelt

In [21], Bloo, Kamareddine and Nederpelt extend the usual PTSs with both Π-conversion and definitions. Therefore it is useful to take Π-conversion into consideration when comparing AUTOMATH with $\lambda\beta\Pi$. Though our system $\lambda 68$ does not have Π-conversion, it is very easy to extend it to a system $\lambda\Pi 68$ by:

- Changing rule (App_1) into

$$\frac{\Delta;\Gamma \vdash M : \Pi x{:}A.B \qquad \Delta;\Gamma \vdash N : A}{\Delta;\Gamma \vdash MN : (\Pi x{:}A.B)N}$$

  (rule (App_2) remains unchanged — see also the discussion in Section 4a;

- Adding a new reduction rule $\to_\Pi$ by

$$(\Pi x{:}A.B)N \to_\Pi B[x{:=}N].$$

The system $\lambda\Pi 68$ is actually much closer to AUT-68 than $\lambda 68$ as AUT-68 has Π-conversion as well. In the rest of this paper we only did not focus on Π-conversion in order not to lose the view on what is going on in the definition system of AUTOMATH.
[21] starts with PTSs extended with Π-reduction, but *without* definitions (see [22]). This system (which we will call $\lambda\beta\Pi$ for the moment) does not have the Subject Reduction property. For instance, one can derive

$$\alpha{:}*, x{:}\alpha \vdash (\lambda y{:}\alpha.y)x : (\Pi y{:}\alpha.\alpha)x$$

but it is not possible to derive

$$\alpha{:}*, x{:}\alpha \vdash x : (\Pi y{:}\alpha.\alpha)x.$$

Adding a definition mechanism results in a system that we will call $\lambda\beta\Pi\mathrm{d}$ and is the main point of interest in [21]. As a sort of "side effect" of adding this definition mechanism, $\lambda\beta\Pi\mathrm{d}$ has Subject Reduction.

In $\lambda\Pi 68$ we do not have Subject Reduction: It is not hard to derive

$$;\alpha{:}*, x{:}\alpha \vdash (\lambda y{:}\alpha.y)x : (\Pi y{:}\alpha.\alpha)x$$

in $\lambda\Pi 68$. Nevertheless, we can not derive

$$;\alpha{:}*, x{:}\alpha \vdash x : (\Pi y{:}\alpha.\alpha)x$$

(in such a derivation, no definitions can occur: definitions, once they have been introduced, can not be removed from the left part of the context any more. When we are not allowed to use any definition rules, $\lambda\Pi 68$ has not more rules than the system $\lambda\beta\Pi$ of Bloo, Kamareddine and Nederpelt).
The "restauration" of Subject Reduction in $\lambda\beta\Pi$d is only due to the special way in which definitions are introduced and removed from the context. We do not go into details on this; the interested reader can consult [21].

Another main difference between $\lambda\Pi 68$ and $\lambda\beta\Pi$d has already appeared in Section 6a: In $\lambda\Pi 68$ there is a different correspondence between the types of definiendum and definiens as in $\lambda\beta\Pi$d.

7 Conclusions and Future Work

In this paper we described the most basic AUTOMATH-system, AUT-68, in a PTS style. Though such descriptions have been given before in, for example, [3] and [15], we feel that our description is more accurate than the two ones cited above. Moreover, our description pays attention to the definition system, which is a crucial item in AUTOMATH, and the descriptions above don't.

$\lambda 68$, the main topic of this paper, doesn't include Π-conversion (while AUTOMATH does). However, it is very easy to adapt $\lambda 68$ to include Π-conversion (this was done in Section 6b to compare our system to the system in [21]).

The adaption of $\lambda 68$ to a system λQE, representing the AUTOMATH-system AUT-QE isn't hard, either: It requires adaption of the Π-formation rule to include not only the rule $(*,*,*)$ but also $(*,\Box,\Box)$ and introduction of an additional reduction rule (so-called "type inclusion")

$$\Pi x{:}A.* \to_{\mathrm{QE}} *$$
$$\lambda x{:}A.* \to_{\mathrm{QE}} *$$

For more details on this rule, see [13]. Of course, the properties of $\lambda 68$ presented in Section 5 have to be reviewed for these new systems.

When comparing $\lambda 68$ to other type systems with definitions, we find an important difference. In $\lambda 68$, the correspondence between types of definiendum and definiens differs from the similar correspondence in the systems in [29] and [21].
The reason why $\lambda 68$ differs from other theories on this point has been discussed in Section 4c: the definition system in AUTOMATH allows *parameters* to occur in the definiens, and there is no parameter mechanism in PTSs. We are currently investigating the possibility of extending PTSs with such parametric definitions. This is not only interesting with respect to AUTOMATH, but also with respect to implementations of some type systems (like Coq and HOL), which also have a parameter mechanism.

Acknowledgements

I would like to thank Bert van Benthem-Jutting, Roel Bloo, Herman Geuvers, Kees Hemerik, Fairouz Kamareddine, Rob Nederpelt, Paula Severi de Santiago and Jan Zwanenburg for their help and suggestions.

References

[1] S. Abramsky, Dov M. Gabbay, and T.S.E. Maibaum, editors. *Handbook of Logic in Computer Science, Volume 2: Background: Computational Structures.* Oxford Science Publications, 1992.

[2] H.P. Barendregt. *The Lambda Calculus: its Syntax and Semantics.* North-Holland, Amsterdam, revised edition, 1984.

[3] H.P. Barendregt. Lambda calculi with types. In [1], pages 117–309. Oxford University Press, 1992.

[4] L.S. van Benthem Jutting. Description of AUT-68. Technical Report 12, Eindhoven University of Technology, 1981. also in [26].

[5] S. Berardi. Towards a mathematical analysis of the Coquand-Huet calculus of constructions and the other systems in Barendregt's cube. Technical report, Dept. of Computer Science, Carnegie-Mellon University and Dipartimento Matematica, Universita di Torino, 1988.

[6] R. Bloo, F. Kamareddine, and R. Nederpelt. The Barendregt Cube with definitions and generalised reduction. Technical Report 34, TUE Computing Science Reports, Eindhoven University of Technology, 1994. Also as Technical Report 8, University of Glasgow, Computing Science Department, 1994. To be published in Information and Computation.

[7] N.G. de Bruijn. The mathematical language AUTOMATH, its usage and some of its extensions. In M. Laudet, D. Lacombe, and M. Schuetzenberger, editors, *Symposium on Automatic Demonstration*, pages 29–61, IRIA, Versailles, 1968. Springer Verlag, Berlin, 1970. Lecture Notes in Mathematics **125**; also in [26].

[8] N.G. de Bruijn. Reflections on Automath. Eindhoven University of Technology, 1990. Also in [26].

[9] A. Church. A set of postulates for the foundation of logic (1). *Annals of Mathematics*, 33:346–366, 1932.

[10] A. Church. A set of postulates for the foundation of logic (2). *Annals of Mathematics*, 34:839–864, 1933.

[11] A. Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5:56–68, 1940.

[12] H.B. Curry and R. Feys. *Combinatory Logic*, volume I of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1958.

[13] D.T. van Daalen. *The Language Theory of Automath.* PhD thesis, Eindhoven University of Technology, 1980.

[14] G. Frege. *Begriffschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens.* Nebert, Halle, 1879. Also in [17].

[15] J.H. Geuvers. *Logics and Type Systems.* PhD thesis, Catholic University of Nijmegen, 1993.

[16] J.H. Geuvers and M.J. Nederhof. A modular proof of strong normalization for the Calculus of Constructions. *Journal of Functional Programming*, 1(2):155–189, 1991.

[17] J. van Heijenoort, editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931.* Harvard University Press, Cambridge, Massachusetts, 1967.

[18] A. Heyting. *Intuitionism, an introduction.* Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1956.

[19] D. Hilbert and W. Ackermann. *Grundzüge der Theoretischen Logik.* Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen, Band XXVII. Springer Verlag, Berlin, first edition, 1928.

[20] W.A. Howard. The formulas-as-types notion of construction. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490, New York, 1980. Academic Press.

[21] F. Kamareddine, R. Bloo, and R. Nederpelt. Definitions and Π-conversion in type theory. *Submitted*, 1995.

[22] F. Kamareddine and R.P. Nederpelt. Canonical typing and Π-conversion in the Barendregt Cube. *Journal of Functional Programming*, 1996. To appear.

[23] J.W. Klop. Term rewriting systems. In [1], pages 1–116. Oxford University Press, 1992.

[24] A.N. Kolmogorov. Zur Deutung der Intuitionistischen Logik. *Mathematisches Zeitschrift*, 35:58–65, 1932.

[25] Z. Luo. ECC and extended Calculus of Constructions. Department of Computer Science, University of Edinburgh.

[26] R.P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer, editors. *Selected Papers on Automath.* Studies in Logic and the Foundations of Mathematics **133**. North-Holland, Amsterdam, 1994.

[27] M.J. O'Donnell. *Computing in Systems Described by Equations*, volume 58 of *Lecture Notes in Computer Science.* Springer Verlag, 1977.

[28] F.P. Ramsey. The foundations of mathematics. *Proceedings of the London Mathematical Society*, pages 338–384, 1925.

[29] P. Severi and E. Poll. Pure type systems with definitions. In *LFCS'94* (LNCS **813**), pages 316–328. Springer Verlag, 1994.

[30] J. Terlouw. Een nadere bewijstheoretische analyse van GSTT's. Technical report, Department of Computer Science, University of Nijmegen, 1989.

[31] A.N. Whitehead and B. Russell. *Principia Mathematica.* Cambridge University Press, 1910^1, 1927^2.

Computing Science Reports

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.
93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness, p. 24
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.

93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.
93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.
93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.
93/46	T. Kloks D. Kratsch J. Spinrad	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Kornatzky	Browsing Semantics in the "Tower" Model, p. 19.
93/48	R. Gerth	Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.
94/01	P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart	The object-oriented paradigm, p. 28.
94/02	F. Kamareddine R.P. Nederpelt	Canonical typing and Π-conversion, p. 51.
94/03	L.B. Hartman K.M. van Hee	Application of Marcov Decision Processe to Search Problems, p. 21.
94/04	J.C.M. Baeten J.A. Bergstra	Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.
94/05	P. Zhou J. Hooman	Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22.
94/06	T. Basten T. Kunz J. Black M. Coffin D. Taylor	Time and the Order of Abstract Events in Distributed Computations, p. 29.
94/07	K.R. Apt R. Bol	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.

94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doornbos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.
94/18	F. Kamareddine R. Nederpelt	Refining Reduction in the Lambda Calculus, p. 15.
94/19	B.W. Watson	The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.
94/20	R. Bloo F. Kamareddine R. Nederpelt	Beyond β-Reduction in Church's $\lambda\rightarrow$, p. 22.
94/21	B.W. Watson	An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.
94/22	B.W. Watson	The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.
94/23	S. Mauw and M.A. Reniers	An algebraic semantics of Message Sequence Charts, p. 43.
94/24	D. Dams O. Grumberg R. Gerth	Abstract Interpretation of Reactive Systems: Abstractions Preserving $\forall$CTL*, $\exists$CTL* and CTL*, p. 28.
94/25	T. Kloks	$K_{1,3}$-free and W_4-free graphs, p. 10.
94/26	R.R. Hoogerwoord	On the foundations of functional programming: a programmer's point of view, p. 54.
94/27	S. Mauw and H. Mulder	Regularity of BPA-Systems is Decidable, p. 14.
94/28	C.W.A.M. van Overveld M. Verhoeven	Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20.
94/29	J. Hooman	Correctness of Real Time Systems by Construction, p. 22.
94/30	J.C.M. Baeten J.A. Bergstra Gh. Ştefanescu	Process Algebra with Feedback, p. 22.
94/31	B.W. Watson R.E. Watson	A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.
94/32	J.J. Vereijken	Fischer's Protocol in Timed Process Algebra, p. 38.
94/33	T. Laan	A formalization of the Ramified Type Theory, p.40.
94/34	R. Bloo F. Kamareddine R. Nederpelt	The Barendregt Cube with Definitions and Generalised Reduction, p. 37.
94/35	J.C.M. Baeten S. Mauw	Delayed choice: an operator for joining Message Sequence Charts, p. 15.
94/36	F. Kamareddine R. Nederpelt	Canonical typing and Π-conversion in the Barendregt Cube, p. 19.
94/37	T. Basten R. Bol M. Voorhoeve	Simulating and Analyzing Railway Interlockings in ExSpect, p. 30.
94/38	A. Bijlsma C.S. Scholten	Point-free substitution, p. 10.

94/39	A. Blokhuis T. Kloks		On the equivalence covering number of splitgraphs, p. 4.
94/40	D. Alstein		Distributed Consensus and Hard Real-Time Systems, p. 34.
94/41	T. Kloks D. Kratsch		Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.
94/42	J. Engelfriet J.J. Vereijken		Concatenation of Graphs, p. 7.
94/43	R.C. Backhouse M. Bijsterveld		Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.
94/44	E. Brinksma R. Gerth W. Janssen S. Katz M. Poel C. Rump	J. Davies S. Graf B. Jonsson G. Lowe A. Pnueli J. Zwiers	Verifying Sequentially Consistent Memory, p. 160
94/45	G.J. Houben		Tutorial voor de ExSpect-bibliotheek voor "Administratieve Logistiek", p. 43.
94/46	R. Bloo F. Kamareddine R. Nederpelt		The λ-cube with classes of terms modulo conversion, p. 16.
94/47	R. Bloo F. Kamareddine R. Nederpelt		On Π-conversion in Type Theory, p. 12.
94/48	Mathematics of Program Construction Group		Fixed-Point Calculus, p. 11.
94/49	J.C.M. Baeten J.A. Bergstra		Process Algebra with Propositional Signals, p. 25.
94/50	H. Geuvers		A short and flexible proof of Strong Normalazation for the Calculus of Constructions, p. 27.
94/51	T. Kloks D. Kratsch H. Müller		Listing simplicial vertices and recognizing diamond-free graphs, p. 4.
94/52	W. Penczek R. Kuiper		Traces and Logic, p. 81
94/53	R. Gerth R. Kuiper D. Peled W. Penczek		A Partial Order Approach to Branching Time Logic Model Checking, p. 20.
95/01	J.J. Lukkien		The Construction of a small CommunicationLibrary, p.16.
95/02	M. Bezem R. Bol J.F. Groote		Formalizing Process Algebraic Verifications in the Calculus of Constructions, p.49.
95/03	J.C.M. Baeten C. Verhoef		Concrete process algebra, p. 134.
95/04	J. Hidders		An Isotopic Invariant for Planar Drawings of Connected Planar Graphs, p. 9.
95/05	P. Severi		A Type Inference Algorithm for Pure Type Systems, p.20.
95/06	T.W.M. Vossen M.G.A. Verhoeven H.M.M. ten Eikelder E.H.L. Aarts		A Quantitative Analysis of Iterated Local Search, p.23.
95/07	G.A.M. de Bruyn O.S. van Roosmalen		Drawing Execution Graphs by Parsing, p. 10.
95/08	R. Bloo		Preservation of Strong Normalisation for Explicit Substitution, p. 12.
95/09	J.C.M. Baeten J.A. Bergstra		Discrete Time Process Algebra, p. 20
95/10	R.C. Backhouse R. Verhoeven O. Weber		Mathʃpad: A System for On-Line Prepararation of Mathematical Documents, p. 15

95/11	R. Seljée	Deductive Database Systems and integrity constraint checking, p. 36.
95/12	S. Mauw and M. Reniers	Empty Interworkings and Refinement Semantics of Interworkings Revised, p. 19.
95/13	B.W. Watson and G. Zwaan	A taxonomy of sublinear multiple keyword pattern matching algorithms, p. 26.
95/14	A. Ponse, C. Verhoef, S.F.M. Vlijmen (eds.)	De proceedings: ACP'95, p.
95/15	P. Niebert and W. Penczek	On the Connection of Partial Order Logics and Partial Order Reduction Methods, p. 12.
95/16	D. Dams, O. Grumberg, R. Gerth	Abstract Interpretation of Reactive Systems: Preservation of CTL*, p. 27.
95/17	S. Mauw and E.A. van der Meulen	Specification of tools for Message Sequence Charts, p. 36.
95/18	F. Kamareddine and T. Laan	A Reflection on Russell's Ramified Types and Kripke's Hierarchy of Truths, p. 14.
95/19	J.C.M. Baeten and J.A. Bergstra	Discrete Time Process Algebra with Abstraction, p. 15.
95/20	F. van Raamsdonk and P. Severi	On Normalisation, p. 33.
95/21	A. van Deursen	Axiomatizing Early and Late Input by Variable Elimination, p. 44.
95/22	B. Arnold, A. v. Deursen, M. Res	An Algebraic Specification of a Language for Describing Financial Products, p. 11.
95/23	W.M.P. van der Aalst	Petri net based scheduling, p. 20.
95/24	F.P.M. Dignum, W.P.M. Nuijten, L.M.A. Janssen	Solving a Time Tabling Problem by Constraint Satisfaction, p. 14.
95/25	L. Feijs	Synchronous Sequence Charts In Action, p. 36.
95/26	W.M.P. van der Aalst	A Class of Petri nets for modeling and analyzing business processes, p. 24.
95/27	P.D.V. van der Stok, J. van der Wal	Proceedings of the Real-Time Database Workshop, p. 106.
95/28	W. Fokkink, C. Verhoef	A Conservative Look at term Deduction Systems with Variable Binding, p. 29.
95/29	H. Jurjus	On Nesting of a Nonmonotonic Conditional, p. 14
95/30	J. Hidders, C. Hoskens, J. Paredaens	The Formal Model of a Pattern Browsing Technique, p.24.
95/31	P. Kelb, D. Dams and R. Gerth	Practical Symbolic Model Checking of the full μ-calculus using Compositional Abstractions, p. 17.
95/32	W.M.P. van der Aalst	Handboek simulatie, p. 51.
95/33	J. Engelfriet and JJ. Vereijken	Context-Free Graph Grammars and Concatenation of Graphs, p. 35.
95/34	J. Zwanenburg	Record concatenation with intersection types, p. 46.
95/35	T. Basten and M. Voorhoeve	An algebraic semantics for hierarchical P/T Nets, p. 32.
96/01	M. Voorhoeve and T. Basten	Process Algebra with Autonomous Actions, p. 12.
96/02	P. de Bra and A. Aerts	Multi-User Publishing in the Web: DreSS, A Document Repository Service Station, p. 12
96/03	W.M.P. van der Aalst	Parallel Computation of Reachable Dead States in a Free-choice Petri Net, p. 26.
96/04	S. Mauw	Example specifications in phi-SDL.
96/05	T. Basten and W.M.P. v.d. Aalst	A Process-Algebraic Approach to Life-Cycle Inheritance Inheritance = Encapsulation + Abstraction, p. 15.
96/06	W.M.P. van der Aalst and T. Basten	Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.
96/07	M. Voorhoeve	Structural Petri Net Equivalence, p. 16.
96/08	A.T.M. Aerts, P.M.E. De Bra, J.T. de Munk	OODB Support for WWW Applications: Disclosing the internal structure of Hyperdocuments, p. 14.
96/09	F. Dignum, H. Weigand, E. Verharen	A Formal Specification of Deadlines using Dynamic Deontic Logic, p. 18.
96/10	R. Bloo, H. Geuvers	Explicit Substitution: on the Edge of Strong Normalisation, p. 13.