On Two Forms of Bureaucracy in Derivations

Kai Brünnler[1] and Stéphane Lengrand[2]

[1] Institut für angewandte Mathematik und Informatik
Neubrückstr. 10, CH – 3012 Bern, Switzerland
[2] PPS, Université Paris 7, France

Abstract. We call irrelevant information in derivations *bureaucracy.* An example of such irrelevant information is the order between two consecutive inference rules that trivially permute. Building on ideas by Guglielmi, we identify two forms of bureaucracy that occur in the calculus of structures (and, in fact, in every non-trivial term rewriting derivation). We develop term calculi that provide derivations that do not contain this bureaucracy. We also give a normalisation procedure that removes bureaucracy from derivations and find that in a certain sense the normalisation process is a process of cut elimination.

1 Introduction

Consider the following two proofs in a sequent system for classical logic:

$$\dfrac{\dfrac{\vdash A, B, \bar{A}, C}{\vdash A, B, \bar{A} \vee C}}{\vdash A \vee B, \bar{A} \vee C} \qquad \text{and} \qquad \dfrac{\dfrac{\vdash A, B, \bar{A}, C}{\vdash A \vee B, \bar{A}, C}}{\vdash A \vee B, \bar{A} \vee C} \quad .$$

Clearly, these two proofs are essentially the same, and we prefer not to distinguish them. More to the point, the sequent calculus forces us to choose an order of two rule applications that we do not want to choose because it is not relevant. Let us call *bureaucracy* this fact that a proof-theoretic formalism forces us to distinguish morally identical proofs. Proof nets, introduced by Girard [4] for linear logic, are a less bureaucratic formalism than the sequent calculus. They have also been developed for classical logic, for example by Robinson [13] and by Lamarche and Straßburger [11]. Proof nets have the merit that they do not distinguish between proofs such as the above. However, establishing the correctness of a proof net generally requires checking a global criterion which is more algorithmic than deductive. The notion of *deduction step* is lost when moving from the sequent calculus to proof nets. Let us informally call a formalism *deductive* if it has a notion of inference step. That is of course a vague notion that can be made more precise in several ways, for example in asking for a locally checkable correctness criterion. The question then is whether there is a formalism that is both: bureaucracy-free and deductive. The quest for such *deductive proof nets* was initiated by Guglielmi. Starting from the calculus of structures [5,6] he designs two formalisms that reduce bureaucracy without losing deductiveness: they are called *Formalism A* [7] and *Formalism B* [8]. These formalisms are just steps towards deductive proof nets and not the final result. These formalisms still allow to form inessentially different proofs. However, they providing a third proof which is a canonical representative of the two. So, compared to the sequent calculus or the calculus of structures, the set of proofs grows larger. The ultimate aim in this quest is then a deductive formalism that does not allow the formation of non-canonical proofs in the first place.

Formalisms A and B address two kinds of bureaucracy that occur in every non-trivial term rewriting derivation. Formalism A addresses bureaucracy type A, where one has to choose an order for two consecutive applications of rewrite rules that apply to disjoint, i.e. non-overlapping, subterms of a term. Formalism B addresses bureaucracy type A and also bureaucracy type B, where one has to choose an order between two consecutive applications of rewrite rules where one rule applies to a term which is unchanged by the other rule because it is inside a variable. Clearly, equivalence in Formalism B then corresponds to standard notion of "equivalence modulo trivial permutation".

The starting point of the work presented in this paper was our goal to provide a normalisation procedure for Formalisms A and B which, given a bureaucratic derivation, yields its bureaucracy-free representant. In [7,8] Guglielmi does not provide such a normalisation procedure, though he defines a set of proofs that seems rich enough to accommodate bureaucracy-free representatives. To achieve our goal, we found it natural and also necessary to depart a bit from the definitions in [7,8] in two ways. First, we use a term calculus to define derivations. This is just a notational difference, comparable to the difference between the λ-calculus and natural deduction in minimal logic. Second, we found it hard to work modulo the equational theory that is used not only in [7,8] but also generally in systems in the calculus of structures. It typically contains equations like associativity and commutativity of conjunction and disjunction, for example. We drop the equational theory and add those equations that are needed for completeness as rules.

The plan of the paper is as follows: we first present a linear term rewriting system that is a deductive system for classical propositional logic. We go on to define proof terms for derivations in Formalism A and give a rewriting system for these proof terms that removes bureaucracy, which, as we will see, turns out to be a process of cut elimination. The following section, where we do the same for Formalism B, is intended to be the heart of the paper, but is still a bit of a construction site. The central notion though is already there, it is that of a *tube* which is a placeholder which winds through a derivation and contains another derivation. Tubes put an end to bureaucracy type B. Some discussion ends the paper.

2 Propositional Logic as Term Rewriting

A deductive system in the calculus of structures [6] is just a term rewriting system modulo an equational theory, cf. [10]. We should point out that the questions one asks about these systems are rather different. Typical properties of interest of a term rewriting system are termination and confluence, systems in the calculus of structures typically are neither. Typical questions to ask of systems in the calculus of structures are about the admissibility of rules and about the existence of certain normal forms for derivations. Nevertheless, term rewriting systems is what they are. In this section we present a linear term rewriting system on formulas in propositional logic such that a formula A rewrites to a formula B iff A implies B. This system is essentially obtained from system SKS from [2,1] by removing all equations and adding some of them as rewrite rules. The system is rather idiosyncratic and is not really central to the ideas developed here. We present it just in order to show that linear term rewriting indeed can serve as a proof-theoretic formalism and also to have some rules as running examples. Formalisms A and B as we present them easily generalise to any linear term rewriting system.

Formulas. There are propositional variables, denoted by v. Propositional variables v and their negations $\bar{v}$ are *atoms*, they are denoted by a, b, and so on. The letters A, B, C denote formulas, which are defined as follows:

$$A ::= \mathsf{f} \mid \mathsf{t} \mid a \mid (A \vee A) \mid (A \wedge A)$$

where f and t are the units *false* and *true*. We define $\bar{A}$, the *negation* of the formula A, in the usual way:

$$\begin{array}{ll} \bar{\mathsf{f}} = \mathsf{t} & \overline{A \vee B} = \bar{A} \wedge \bar{B} \\ \bar{\mathsf{t}} = \mathsf{f} & \overline{A \wedge B} = \bar{A} \vee \bar{B} \end{array} \qquad \bar{\bar{v}} = v \quad .$$

Term rewriting rules. A system of rewrite rules for classical propositional logic is given in Figure 1. The subsystem on the left is called KSf where K means classical, S means calculus of structures and f is for (equation-)free. The entire system is called SKSf, where the first S is for symmetric. On top of the arrow is the *label* of the rewrite rule. The labels of the rules on the left are short for duplication, unit, commutativity, identity, switch, weakening and contraction. Their dual rules on the right have the same name but with the prefix "co-". When viewing formulas as terms, we view atoms as constants. A rewrite rule containing an atom, like $\mathsf{f} \xrightarrow{\mathsf{aw}\downarrow} a$ is shorthand for the set of rewrite rules one obtains by replacing a by any atom.

We have soundness and completeness for classical propositional logic.

$\mathsf{t} \xrightarrow{\mathsf{du}\downarrow} \mathsf{t} \wedge \mathsf{t} \qquad A \xrightarrow{\mathsf{un}\downarrow} A \vee \mathsf{f}$	$A \wedge \mathsf{t} \xrightarrow{\mathsf{un}\uparrow} A \qquad \mathsf{f} \vee \mathsf{f} \xrightarrow{\mathsf{du}\uparrow} \mathsf{f}$
$A \vee B \xrightarrow{\mathsf{co}\downarrow} B \vee A$	$A \wedge B \xrightarrow{\mathsf{co}\uparrow} B \wedge A$
$\mathsf{t} \xrightarrow{\mathsf{i}\downarrow} \bar{A} \vee A$	$A \wedge \bar{A} \xrightarrow{\mathsf{i}\uparrow} \mathsf{f}$
$(A \vee B) \wedge (C \vee D) \xrightarrow{\mathsf{s}\downarrow} (A \vee C) \vee (B \wedge D)$	$(A \wedge C) \wedge (B \vee D) \xrightarrow{\mathsf{s}\uparrow} (A \wedge B) \vee (C \wedge D)$
$\mathsf{f} \xrightarrow{\mathsf{w}\downarrow} A \qquad A \vee A \xrightarrow{\mathsf{c}\downarrow} A$	$A \xrightarrow{\mathsf{c}\uparrow} A \vee A \qquad A \xrightarrow{\mathsf{w}\uparrow} \mathsf{t}$

Fig. 1. Rewrite rules for propositional logic

Theorem 1 *1.* $\mathsf{t} \rightarrow^*_{\mathsf{KSf}} A$ *iff* A *is valid.*
2. $A \rightarrow^*_{\mathsf{SKSf}} B$ *iff* A *implies* B.

Proof. Soundness in both cases follows from a simple induction on the length of the derivation and the observation that implication is closed under conjunction and disjunction. System KSf is complete: a formula can be derived from its conjunctive normal form via the rules $\mathsf{c}\downarrow, \mathsf{co}\downarrow, \mathsf{s}\downarrow$. If the formula is valid, then each of the nested disjunctions in the conjunctive normal form contains two dual atoms. By $\mathsf{w}\downarrow, \mathsf{un}\downarrow$ this formula can be derived from a formula where all atoms except for the two dual atoms are removed. By $\mathsf{i}\downarrow$ we derive this from a conjunction of lots of occurrences of t, which is derived from t by $\mathsf{du}\downarrow$. The completeness direction of 2) is then a matter of constructing a derivation from A to B in SKSf for each derivation from t to $\bar{A} \vee B$ in KSf, see [1] for details.

Linear rewrite rules. From SKSf we obtain a linear rewriting system SKSfl, where l is for linear, which is shown in Figure 2. Derivability in this system is the same as in the nonlinear system, for details see [1].

Theorem 2 *1.* $A \rightarrow^*_{\mathsf{SKSf}} B$ *iff* $A \rightarrow^*_{\mathsf{SKSfl}} B$
2. $A \rightarrow^*_{\mathsf{KSf}} B$ *iff* $A \rightarrow^*_{\mathsf{KSfl}} B$

3 Formalism A

Consider the following two rewrite paths (or derivations in the calculus of structures):

$$\mathsf{ac}\downarrow \dfrac{(a \vee a) \wedge (b \vee b)}{\mathsf{ac}\downarrow \dfrac{a \wedge (b \vee b)}{a \wedge b}} \qquad \text{and} \qquad \mathsf{ac}\downarrow \dfrac{(a \vee a) \wedge (b \vee b)}{\mathsf{ac}\downarrow \dfrac{(a \vee a) \wedge b}{a \wedge b}} \qquad .$$

$\mathsf{t} \xrightarrow{\mathsf{du}\downarrow} \mathsf{t} \wedge \mathsf{t} \qquad A \xrightarrow{\mathsf{un}\downarrow} A \vee \mathsf{f}$	$A \wedge \mathsf{t} \xrightarrow{\mathsf{un}\uparrow} A \qquad \mathsf{f} \vee \mathsf{f} \xrightarrow{\mathsf{du}\uparrow} \mathsf{f}$
$A \vee B \xrightarrow{\mathsf{co}\downarrow} B \vee A$	$A \wedge B \xrightarrow{\mathsf{co}\uparrow} B \wedge A$
$\mathsf{t} \xrightarrow{\mathsf{ai}\downarrow} \bar{a} \vee a$	$a \wedge \bar{a} \xrightarrow{\mathsf{ai}\uparrow} \mathsf{f}$
$(A \vee B) \wedge (C \vee D) \xrightarrow{\mathsf{s}\downarrow} (A \vee C) \vee (B \wedge D)$	$(A \wedge C) \wedge (B \vee D) \xrightarrow{\mathsf{s}\uparrow} (A \wedge B) \vee (C \wedge D)$
$(A \wedge B) \vee (C \wedge D) \xrightarrow{\mathsf{m}} (A \vee C) \wedge (B \vee D)$	$(A \wedge B) \vee (C \wedge D) \xrightarrow{\mathsf{m}} (A \vee C) \wedge (B \vee D)$
$(A \vee B) \vee (C \vee D) \xrightarrow{\mathsf{m_0}\downarrow} (A \vee C) \vee (B \vee D)$	$(A \wedge B) \wedge (C \wedge D) \xrightarrow{\mathsf{m_0}\uparrow} (A \wedge C) \wedge (B \wedge D)$
$\mathsf{f} \xrightarrow{\mathsf{w_0}\downarrow} \mathsf{f} \wedge \mathsf{f} \qquad \mathsf{f} \xrightarrow{\mathsf{aw}\downarrow} a \qquad a \vee a \xrightarrow{\mathsf{ac}\downarrow} a$	$a \xrightarrow{\mathsf{ac}\uparrow} a \vee a \qquad a \xrightarrow{\mathsf{aw}\uparrow} \mathsf{t} \qquad \mathsf{t} \wedge \mathsf{t} \xrightarrow{\mathsf{w_0}\uparrow} \mathsf{t}$

Fig. 2. Linear rewrite rules for propositional logic

They inessentially differ in the order in which the two rules are applied. No matter which of the two derivations we choose, it contains irrelevant information. We now define Formalism A which provides a third derivation which stores no information about the order between the two applications of $\mathsf{ac}\downarrow$. The solution is of course very simple: we introduce a parallel composition of derivations.

Proof terms. Proof terms (or just terms) of Formalism A, denoted by R, S, T, U are defined as follows:

$$R ::= \mathsf{id} \mid \rho \mid (R \mid R) \mid (R \,.\, R)$$

where id is *identity*, ρ is the label of a rewrite rule from Figure 2, $(R_1 \mid R_2)$ is *parallel composition* and $(R_1 \,.\, R_2)$ is *sequential composition.*

Typing rules. A judgement $A \xrightarrow{R} B$ which can be derived by the typing rules given in Figure 3 from the rewrite rules (aka typing axioms) in Figure 2 says that the proof term R allows to derive A implies B. Not each term is typeable, for example $\mathsf{s}\downarrow \,.\, \mathsf{s}\downarrow$ is not. In general, terms are typeable in different ways. For example we have $a \xrightarrow{\mathsf{id}} a$, just like $b \xrightarrow{\mathsf{id}} b$. We have soundness and completeness for classical propositional logic since we have the following theorem:

Theorem 3 *There is a proof term R of Formalism A with $A \xrightarrow{R} B$ iff $A \to^*_{\mathsf{SKSfl}} B$.*

Proof. The direction from left to right is an easy induction on the typing derivation. The converse is easy to see since a rewrite rule can be applied at an arbitray depth with a proof term build from the label of the rewrite rule, identity and parallel composition, and consecutive rule applications are represented using sequential composition.

$$A \xrightarrow{\mathsf{id}} A \qquad \frac{A \xrightarrow{R} B \quad B \xrightarrow{S} C}{A \xrightarrow{R.S} C}$$

$$\frac{A \xrightarrow{R} C \quad B \xrightarrow{S} D}{A \wedge B \xrightarrow{R|S} C \wedge D} \qquad \frac{A \xrightarrow{R} C \quad B \xrightarrow{S} D}{A \vee B \xrightarrow{R|S} C \vee D}$$

Fig. 3. Typing rules for Formalism A

Reduction rules. The reduction relation $\rightarrow_{\mathsf{A}}$ is given by the following rewrite rules:

$$\begin{aligned}
&R \,.\, \mathsf{id} \rightarrow R\\
&\mathsf{id} \,.\, R \rightarrow R\\
&\mathsf{id} \mid \mathsf{id} \rightarrow \mathsf{id}\\
&(R \mid S) \,.\, (T \mid U) \rightarrow (R \,.\, T) \mid (S \,.\, U)
\end{aligned}$$

Theorem 4 *The reduction relation $\rightarrow_{\mathsf{A}}$ is convergent.*

Proof. Each rule decreases the sum of the number of occurrences of id and the number of occurrences of parallel composition. Local confluence is easily checked.

We call normal forms of $\rightarrow_{\mathsf{A}}$ *canonical*. The reduction rules preserve types. If we call the typing rule for sequential composition "cut", then they actually correspond to cut elimination steps in the typing derivation, as we will see in the proof of the following theorem. Note however that the cut rule for a typing derivation has nothing to with a cut rule that may or may not be part of the logical system we represent using rewrite rules. In our case, all the rules with an up-arrow are in some sense cuts. Their admissibility follows from the previous section and is unrelated to the following theorem.

Theorem 5 (Subject reduction) *If $A \xrightarrow{R} B$ and $R \rightarrow_{\mathsf{A}}^{*} S$ then $A \xrightarrow{S} B$.*

Proof.

$$\frac{A \xrightarrow{\mathsf{id}} A \quad A \xrightarrow{R} B}{A \xrightarrow{\mathsf{id}.R} B} \quad \rightsquigarrow \quad A \xrightarrow{R} B$$

$$\frac{A \xrightarrow{\mathsf{id}} A \quad B \xrightarrow{\mathsf{id}} B}{A \wedge B \xrightarrow{\mathsf{id}|\mathsf{id}} A \wedge B} \quad \rightsquigarrow \quad A \wedge B \xrightarrow{\mathsf{id}} A \wedge B$$

$$\dfrac{\dfrac{A \xrightarrow{R} E \quad B \xrightarrow{S} F}{A \wedge B \xrightarrow{R|S} E \wedge F} \quad \dfrac{E \xrightarrow{T} C \quad F \xrightarrow{U} D}{E \wedge F \xrightarrow{T|U} C \wedge D}}{A \wedge B \xrightarrow{(R|S).(T|U)} C \wedge D}$$

$$\rightsquigarrow \quad \dfrac{\dfrac{A \xrightarrow{R} E \quad E \xrightarrow{T} C}{A \xrightarrow{R.T} C} \quad \dfrac{B \xrightarrow{S} F \quad F \xrightarrow{U} D}{B \xrightarrow{S.U} D}}{A \wedge B \xrightarrow{(R.T)|(S.U)} C \wedge D}$$

Example. The two derivations from the beginning of the section are the following terms: $(\mathsf{ac}{\downarrow} \mid \mathsf{id}) . (\mathsf{id} \mid \mathsf{ac}{\downarrow})$ and $(\mathsf{id} \mid \mathsf{ac}{\downarrow}) . (\mathsf{ac}{\downarrow} \mid \mathsf{id})$. Both normalise to $(\mathsf{ac}{\downarrow} \mid \mathsf{ac}{\downarrow})$.

However, there still is bureaucracy remaining in the canonical derivations of Formalism A. Consider the following two derivations:

$$\mathsf{co}{\downarrow}\dfrac{\mathsf{ac}{\downarrow}\dfrac{(b \vee b) \wedge a}{b \wedge a}}{a \wedge b} \quad \text{and} \quad \mathsf{ac}{\downarrow}\dfrac{\mathsf{co}{\downarrow}\dfrac{(b \vee b) \wedge a}{a \wedge (b \vee b)}}{a \wedge b} \quad ,$$

which have the following proof terms: $(\mathsf{ac}{\downarrow} \mid \mathsf{id}) . \mathsf{co}{\downarrow}$ and $\mathsf{co}{\downarrow} . (\mathsf{id} \mid \mathsf{ac}{\downarrow})$. There is no proof term in Formalism A that composes the two rules in such a way that no order between them is fixed. The next section will provide such a bureaucracy-free proof term.

4 Formalism B

Given an occurrence of an inference rule in a term, in general this rule can be permuted a certain distance to the left and a certain distance to the right (possibly both zero) until it hits another occurrence of an inference rule such that the two collide (do not permute). The actual position of the inference rule within these two points is irrelevant. To capture this free space between the two collision points we introduce *tubes*. Tubes have names, they have a start and an end, and they can be filled with derivations.

Types and proof terms. Starting from Formalism A, we extend the definition of formulas, which we now call types, and that of terms as follows:

$$A ::= \mathsf{f} \mid \mathsf{t} \mid a \mid (A \vee A) \mid (A \wedge A) \mid x_A^A$$

and

$$R ::= \mathsf{id} \mid \rho \mid (R \mid R) \mid (R \,.\, R) \mid x\triangleright \mid \triangleleft x \quad .$$

where x is a name for a tube, in x_B^A the types A and B respectively are premise and conclusion of the tube, $x\triangleright$ marks the start of the tube x and $\triangleleft x$ marks the end of the tube x. For each term we require that each tube name occurs at most once as a tube start and at most once as a tube end. Now our typing rules need to keep track of an environment ε, which is a finite partial mapping from tube names to terms. We write R, ε to denote a pair of a term and and environment. Given an environment ε which is undefined for x we write $\varepsilon, x : R$ to denote the environment which only differs from ε by mapping x to R.

Typing rules. The typing rules for Formalism B are shown in Figure 4.

$$A \xrightarrow{\mathsf{id},\varepsilon} A \qquad \frac{A \xrightarrow{R,\varepsilon} B \quad B \xrightarrow{S,\varepsilon} C}{A \xrightarrow{R.S,\varepsilon} C}$$

$$\frac{A \xrightarrow{R,\varepsilon} C \quad B \xrightarrow{S,\varepsilon} D}{A \wedge B \xrightarrow{R|S,\varepsilon} C \wedge D} \qquad \frac{A \xrightarrow{R,\varepsilon} C \quad B \xrightarrow{S,\varepsilon} D}{A \vee B \xrightarrow{R|S,\varepsilon} C \vee D}$$

$$\frac{A \xrightarrow{R,\varepsilon} B}{A \xrightarrow{x\triangleright,\varepsilon,x:R} x^A_B} \qquad \frac{A \xrightarrow{R,\varepsilon} B}{x^A_B \xrightarrow{\triangleleft x,\varepsilon,x:R} B}$$

Fig. 4. Typing rules for Formalism B

Just like in Formalism A, we have soundness and completeness for classical propositional logic since we have the following theorem:

Theorem 6 *For all formulas A, B there is a proof term R of Formalism A with $A \xrightarrow{R} B$ iff there is a proof term T of Formalism B with $A \xrightarrow{T} B$.*

Proof. The direction from left to right is obvious, just take an empty environment. For the converse, we first inductively define the premise $p(A)$ of a type A by pulling it over the propositional connectives and letting $p(x^A_B) = p(A)$. We define the conclusion $c(A)$ likewise, letting $c(x^A_B) = c(B)$. Now, by an easy induction on the typing derivation we establish that for all types A, B if we have $A \xrightarrow{T} B$ in Formalism B then there is a term R in Formalism A such that $p(A) \xrightarrow{R} c(B)$.

Normalisation process. To obtain a bureaucracy-free representant of a proof, we start from a proof term in Formalism A. The normalisation process has three steps.

The first step is an initialisation, in which for every rule we add its inner tubes, e.g. $\mathsf{co}{\downarrow}$ is replaced by $(x\triangleright \mid y\triangleright)\ .\ \mathsf{co}{\downarrow}\ .\ (\triangleleft y \mid \triangleleft x)$. We define the corresponding environment to map all occurring tubes to id.

The second step extends tubes as much as possible. It is a normalisation using the rewrite rules of Formalism A and the following rewrite rule which work both on the term and on the environment. The term R is either a parallel composition or an inference rule. The expression $S\{R\}\ldots\{T\}$ denotes a term with (fixed occurrences of) subterms $R\ldots T$. We refer to the first two rules as *tube extension* and to the third rule as *tube fusion*.

$$\begin{array}{l}\lhd x\;.\;R\\ \varepsilon, x:T\end{array} \quad\longrightarrow\quad \begin{array}{l}\lhd x\\ \varepsilon, x:T.R\end{array}$$

$$\begin{array}{l}R\;.\;x\rhd\\ \varepsilon, x:T\end{array} \quad\longrightarrow\quad \begin{array}{l}x\rhd\\ \varepsilon, x:R.T\end{array}$$

$$\begin{array}{l}S\{x\rhd\}\{\lhd x\;.\;y\rhd\}\{\lhd y\}\\ \varepsilon, x:T, y:U\end{array} \quad\longrightarrow\quad \begin{array}{l}S\{x\rhd\}\{\mathsf{id}\}\{\lhd x\}\\ \varepsilon, x:T.U, y:\mathsf{id}\end{array}$$

The third step is a cleanup phase, when all empty tubes are discarded:

$$\begin{array}{l}S\{x\rhd\}\{\lhd x\}\\ \varepsilon, x:\mathsf{id}\end{array} \quad\longrightarrow\quad \begin{array}{l}S\{\mathsf{id}\}\{\mathsf{id}\}\\ \varepsilon\end{array}$$

Examples. The minimal example are the terms $(\mathsf{id} \mid \mathsf{ac}{\downarrow})\;.\;\mathsf{co}{\downarrow}$ and $\mathsf{co}{\downarrow}\;.\;(\mathsf{ac}{\downarrow} \mid \mathsf{id})$ that both rewrite to:

$$(\mathsf{id} \mid x\rhd)\;.\;\mathsf{co}{\downarrow}\;.\;(\lhd x \mid \mathsf{id}) \quad , \quad x : \mathsf{ac}{\downarrow} \qquad .$$

More than one rule can be inside a tube. $(\mathsf{id} \mid \mathsf{ac}{\downarrow})\;.\;\mathsf{co}{\downarrow}\;.\;(\mathsf{ac}{\uparrow} \mid \mathsf{id})$ rewrites to:

$$(\mathsf{id} \mid x\rhd)\;.\;\mathsf{co}{\downarrow}\;.\;(\lhd x \mid \mathsf{id}) \quad , \quad x : \mathsf{ac}{\downarrow}\;.\;\mathsf{ac}{\uparrow} \qquad .$$

Tubes can be nested. $((\mathsf{ac}{\downarrow} \mid \mathsf{id}) \mid \mathsf{id})\;.\;\mathsf{co}{\downarrow}\;.\;(\mathsf{id} \mid \mathsf{co}{\downarrow})$ rewrites to:

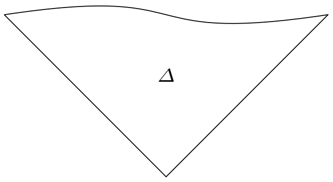
$$(x\rhd \mid \mathsf{id})\;.\;\mathsf{co}{\downarrow}\;.\;(\mathsf{id} \mid \lhd x) \quad , \quad \begin{array}{l}x : (y\rhd \mid \mathsf{id})\;.\;\mathsf{co}{\downarrow}\;.\;(\mathsf{id} \mid \lhd y)\\ y : \mathsf{ac}{\downarrow}\end{array} \qquad .$$

The reduction relation preserves types:

Theorem 7 (Subject reduction) *If $A \xrightarrow{R} B$ and $R \to_{\mathsf{B}}^{*} S$ then $A \xrightarrow{S} B$.*

Proof. It is easy to check that the first and third step preserve typing, we give the necessary transformation of the typing derivation for tube extension in the second step. Tube fusion works similarly. The derivation

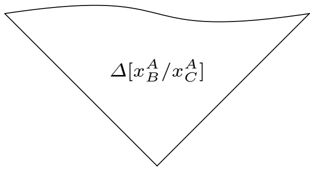
$$\dfrac{A \xrightarrow{T,\varepsilon} B}{A \xrightarrow{x\triangleright,\varepsilon,x:T} x^A_B} \qquad \dfrac{\dfrac{A \xrightarrow{T,\varepsilon} B}{x^A_B \xrightarrow{x\triangleright,\varepsilon,x:T} B} \quad B \xrightarrow{R,\varepsilon,x:T} C}{x^A_B \xrightarrow{\triangleleft x.R,\varepsilon,x:T} C}$$



$$D \xrightarrow{S\{x\triangleright\}\{\triangleleft x.R\},\varepsilon,x:T} E$$

transforms into

$$\dfrac{\dfrac{A \xrightarrow{T,\varepsilon} B \quad B \xrightarrow{R,\varepsilon} C}{A \xrightarrow{T.R,\varepsilon} C}}{A \xrightarrow{x\triangleright,\varepsilon,x:T.R} x^A_C} \qquad \dfrac{\dfrac{A \xrightarrow{T,\varepsilon} B \quad B \xrightarrow{R,\varepsilon} C}{A \xrightarrow{T.R,\varepsilon} C}}{x^A_C \xrightarrow{\triangleleft x,\varepsilon,x:T.R} C}$$



$$D \xrightarrow{S\{x\triangleright\}\{\triangleleft x\},\varepsilon,x:T.R} E \qquad ,$$

where a typing derivation for $B \xrightarrow{R,\varepsilon} C$ can be obtained from the one for $B \xrightarrow{R,\varepsilon,x:T} C$ since neither end of tube x can occur in R.

Conjecture 8 *The normalisation process is convergent modulo naming of tubes.*

5 Discussion

Is all bureaucracy gone now? Unfortunately, no. This work is only at the beginning and even the basic notions of types and terms in Formalism A are not stable yet. There still is bureaucracy due to associativity of sequential and parallel composition, such as $(R.T).U$ versus $R.(T.U)$. For sequential composition this is easy to get rid of by using multiary function symbols and by writing $(R.T.U)$. We have to suitably generalise the typing rule as follows:

$$\dfrac{A_1 \xrightarrow{R_1} A_2 \quad A_2 \xrightarrow{R_2} A_3 \ \ldots\ A_{n-1} \xrightarrow{R_{n-1}} A_n}{A_1 \xrightarrow{R_1.R_2.\ldots.R_{n-1}} A_n}$$

For parallel composition, however, the case is more complicated. In general, we cannot be sure that our canonical terms are bureaucracy-free until we have shown them to be in one-to-one correspondence with equivalence classes of rewriting derivations modulo trivial permutation. We are not there yet.

3-categories. Obtaining associativity of parallel composition will also be necessary in order to achieve our goal of making terms in Formalism B form a *3-category* à la Albert Burroni [3]. It seems that arrows in a 3-category capture exactly the bureaucracy we have in mind. See also Yves Guiraud's work [9] on the relationship between deep inference and 3-categories.

Type checking redundancy. The system of typing rules given for formalism B has the disadvantage that a derivation in a tube has to be type checked twice: once for the start of the tube and once for the end of the tube. It would be interesting to develop a type system where it has to be type checked only once, maybe by defining proof terms and types as

$$A ::= \mathsf{f} \mid \mathsf{t} \mid a \mid (A \vee A) \mid (A \wedge A) \mid x$$

and

$$R ::= \mathsf{id} \mid \rho \mid (R \mid R) \mid (R \,.\, R) \mid x\rhd \mid \lhd x \mid (x, A)\rhd \mid \lhd(x, A) \quad .$$

and replacing the rules for the tubes by the following ones:

$$A \xrightarrow{(x,A)\rhd,\emptyset} x \qquad\qquad x \xrightarrow{\lhd(x,A),\emptyset} A$$

$$\frac{A \xrightarrow{R\{(x,C)\rhd\}\{\lhd(x,D)\},\varepsilon} B \quad C \xrightarrow{S,\varepsilon} D}{A \xrightarrow{R\{x\rhd\}\{\lhd x\},\varepsilon,x:S} B} \quad .$$

Church vs. Curry We chose Curry-style typing for brevity, but it could be done in Church style. Then we need two parallel constructors, one for conjunction and one for disjunction. All inference rules are then parametrised by their types and instead of $A \xrightarrow{\mathsf{id}} A$ for every A, we have for each A an $\mathsf{id}(A)$ such that $A \xrightarrow{\mathsf{id}(A)} A$. Church style could be more convenient for enforcing associativity of parallel composition or for type checking.

Rewriting Logic. There is a close connection with rewriting logic that needs to be made explicit. In the language of rewriting logic [12], our goal with Formalism B is to give canonical representants for arrows in the initial model of a rewrite theory if the rewrite theory is linear and without equations. Speaking of rewriting logic: the deductive system for rewriting logic as given in [12] already provides proof terms that are free of bureaucracy type A. Its congruence rule corresponds to our rule for parallel composition. However, this deductive system does not provide derivations that are free of bureaucracy of type B. To see that one needs to consider an example with two rules that do not permute and a third rule that permutes through both.

Bureaucracy in the formalism vs. bureaucracy in the logic. The axioms (or rewrite rules) of SKSf just served as an example here, we really are addressing bureaucracy in the formalism, which is independent of the particular logic that we are formalising. For the attack on logic-independent bureaucracy two obvious directions for further work are the extension of our approach 1) to term rewriting systems in general, not only those with linear rules, and 2) to term rewriting systems modulo equations. But there is also logic-dependent bureaucracy that needs to be taken care of such as in classical logic a weakening followed by a contraction, to name a simple example.

References

1. Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. PhD thesis, Technische Universität Dresden, September 2003.
2. Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR 2001*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 347–361. Springer-Verlag, 2001.
3. Albert Burroni. Higher-dimensional word problems with applications to equational logic. *Theoretical Computer Science*, 115(1):43–62, 1993.
4. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
5. Alessio Guglielmi. The calculus of structures website. Available from http://www.ki.inf.tu-dresden.de/~guglielm/Research/.
6. Alessio Guglielmi. A system of interaction and structure. Technical Report WV-02-10, Technische Universität Dresden, 2002. To appear in ACM Transactions on Computational Logic.
7. Alessio Guglielmi. Formalism A. Manuscript. http://iccl.tu-dresden.de/~guglielm/p/AG11.pdf, 2004.
8. Alessio Guglielmi. Formalism B. Manuscript. http://iccl.tu-dresden.de/~guglielm/p/AG13.pdf, 2004.
9. Yves Guiraud. The three dimensions of proofs. Manuscript. http://iml.univ-mrs.fr/~guiraud/recherche/cos.pdf, 2005.
10. Ozan Kahramanoğulları. Implementing system BV of the calculus of structures in Maude. In Laura Alonso i Alemany and Paul Égré, editors, *Proceedings of the ESSLLI-2004 Student Session*, pages 117–127, Université Henri Poincaré, Nancy, France, 2004.
11. François Lamarche and Lutz Straßburger. Naming proofs in classical propositional logic. In Paweł Urzyczyn, editor, *Typed Lambda Calculi and Applications, TLCA 2005*, volume 3461 of *Lecture Notes in Computer Science*, pages 246–261. Springer-Verlag, 2005.
12. Narciso Martí-Oliet and José Meseguer. Rewriting logic: roadmap and bibliography. *Theoretical Computer Science*, 285(2):121–154, 2002.
13. Edmund P. Robinson. Proof nets for classical logic. *Journal of Logic and Computation*, 13(5):777–797, 2003.