On two forms of bureaucracy in derivations

Kai Brünnler and Stéphane Lengrand

16th July 2005

Structures and Deductions workshop, Lisbon, 2005

- Rewriting and Proof-terms for CoS
- Eliminating Type A bureaucracy
- Eliminating Type B bureaucracy

Rewriting and Proof-terms for CoS

Classical logic

Formulae: $A, B, C, \ldots ::= \mathsf{f} \mid \mathsf{t} \mid a \mid (A \vee B) \mid (A \wedge B)$
+ involutive negation: $A \mapsto \overline{A}$

In CoS, Down fragment (KS):

$$\mathsf{t} \xRightarrow{\mathsf{i}\downarrow} \overline{A} \vee A$$

$$A \wedge (C \vee B) \xRightarrow{\mathsf{s}} (A \wedge C) \vee B$$

$$\mathsf{f} \xRightarrow{\mathsf{w}\downarrow} A$$

$$A \vee A \xRightarrow{\mathsf{c}\downarrow} A$$

Proofs are sequences of such implications.
…looks like a Hilbert-style system (starting from a formula A, we apply modus ponens with the above implications)
except: implications apply at any depth
…looks like rewrite system, then.

...except formulae are considered up to the equivalence:

$$\begin{array}{rcl} \mathsf{t} \Longleftrightarrow \mathsf{t} \wedge \mathsf{t} & \qquad & \mathsf{f} \vee \mathsf{f} \Longleftrightarrow \mathsf{f} \\ A \Longleftrightarrow A \vee \mathsf{f} & & A \Longleftrightarrow \mathsf{t} \wedge A \\ A \vee B \Longleftrightarrow B \vee A & & A \wedge B \Longleftrightarrow B \wedge A \\ A \vee (B \vee C) \Longleftrightarrow (A \vee B) \vee C & & A \wedge (B \wedge C) \Longleftrightarrow (A \wedge B) \wedge C \end{array}$$

$\Longrightarrow$ rewriting modulo, or equivalently: 16 rewrite rules.

or equivalently, add to the the down fragment:

$$\mathsf{t} \stackrel{\mathsf{du}\downarrow}{\Longrightarrow} \mathsf{t} \wedge \mathsf{t}$$

$$A \stackrel{\mathsf{un}\downarrow}{\Longrightarrow} A \vee \mathsf{f}$$

$$A \vee B \stackrel{\mathsf{co}\downarrow}{\Longrightarrow} B \vee A$$

and turn switch into: $(A \vee B) \wedge (A' \vee B') \stackrel{\mathsf{s}}{\Longrightarrow} (A \vee A') \vee (B \wedge B')$

Completeness

Theorem (Brünnler)

$\mathsf{t} \Longrightarrow B$ iff B is valid (in classical logic)

When adding the up-fragment (the contrapositive of every rule):

$A \Longrightarrow B$ iff A implies B

Proof-terms

CoS inference rules represented by typed constants:

$\mathsf{du}\downarrow, \mathsf{un}\downarrow, \mathsf{co}\downarrow, \mathsf{i}\downarrow, \mathsf{s}, \mathsf{w}\downarrow, \mathsf{c}\downarrow, \ldots$

Composition (associative)

$$\frac{A \stackrel{R}{\Longrightarrow} B \qquad B \stackrel{S}{\Longrightarrow} C}{A \stackrel{R.S}{\Longrightarrow} C}$$

Denoting positions where rewrite rules apply

$$\frac{A \stackrel{R}{\Longrightarrow} A'}{(A \wedge B) \stackrel{R \,|\, \mathsf{id}}{\Longrightarrow} (A' \wedge B)} \qquad \frac{B \stackrel{S}{\Longrightarrow} B'}{(A \wedge B) \stackrel{\mathsf{id} \,|\, S}{\Longrightarrow} (A \wedge B')}$$

$$\frac{A \stackrel{R}{\Longrightarrow} A'}{(A \vee B) \stackrel{R \,|\, \mathsf{id}}{\Longrightarrow} (A' \vee B)} \qquad \frac{B \stackrel{S}{\Longrightarrow} B'}{(A \vee B) \stackrel{\mathsf{id} \,|\, S}{\Longrightarrow} (A \vee B')}$$

Example

$$\cfrac{\cfrac{\cfrac{(A \vee B) \wedge C}{(A \vee B) \wedge (C \vee \mathsf{f})}}{(A \vee B) \wedge (C \vee D)}}{(A \vee C) \vee (B \wedge D)}$$

is represented by

$$(\mathsf{id} \mid \mathsf{un}\downarrow).(\mathsf{id} \mid (\mathsf{id} \mid \mathsf{w}\downarrow)).\mathsf{s}$$

Eliminating type A bureaucracy

Type A bureaucracy

Example:

$$\dfrac{\dfrac{(a \vee a) \wedge (b \vee b)}{a \wedge (b \vee b)}\,\mathsf{ac}\!\downarrow}{a \wedge b}\,\mathsf{ac}\!\downarrow \qquad \dfrac{\dfrac{(a \vee a) \wedge (b \vee b)}{(a \vee a) \wedge b}\,\mathsf{ac}\!\downarrow}{a \wedge b}\,\mathsf{ac}\!\downarrow$$

are the proofs morally different?

Inference steps seen as rewriting: the two redexes are independent/parallel/disjoint

As proof-terms:

$$(\mathsf{ac}\!\downarrow \mid \mathsf{id}).(\mathsf{id} \mid \mathsf{ac}\!\downarrow) \qquad (\mathsf{id} \mid \mathsf{ac}\!\downarrow).(\mathsf{ac}\!\downarrow \mid \mathsf{id})$$

We just want to write

$$(\mathsf{ac}\!\downarrow \mid \mathsf{ac}\!\downarrow)$$

Formalism A

We just modify the rules:

$$\frac{}{A \xRightarrow{\mathsf{id}} A}$$

$$\frac{A \xRightarrow{R} A' \quad B \xRightarrow{S} B'}{(A \wedge B) \xRightarrow{R \mid S} (A' \wedge B')}$$

$$\frac{A \xRightarrow{R} A' \quad B \xRightarrow{S} B'}{(A \vee B) \xRightarrow{R \mid S} (A' \vee B')}$$

Producing canonical proofs

Proofs are normalised with the following rewrite rules:

$$
\begin{array}{lcl}
R.\mathsf{id} & \longrightarrow & R \\
\mathsf{id}.R & \longrightarrow & R \\
\mathsf{id} \mid \mathsf{id} & \longrightarrow & \mathsf{id} \\
(R \mid S).(T \mid U) & \longrightarrow & (R.T \mid S.U)
\end{array}
$$

Trivially terminating

Confluent

Normal forms are Type A-bureaucracy-free

Subject Reduction

If $A \overset{R}{\Longrightarrow} B$ and $R \longrightarrow S$ then $A \overset{S}{\Longrightarrow} B$

Need for explicit management of commutativity

otherwise $(R \mid S).(T \mid U)$ could rewrite
either to $(R.T \mid S.U)$ or $(R.U \mid S.T)$
$\Longrightarrow$ no sense!

here for the second possibility we need:
$(R \mid S).\mathsf{co}\downarrow.(T \mid U) \equiv (R.U \mid S.T).\mathsf{co}\downarrow \equiv \mathsf{co}\downarrow.(S.T \mid R.U)$

Also needed for type synthesis:
algorithm which, given a term, decides whether it is typable (=is a proof)

(idem for all equivalences turned into inference rules)

But for better readability, we can put the corresponding constant in green, or just hide them.

Eliminating Type B bureaucracy

Type B bureaucracy

Example:

$$\dfrac{\dfrac{(b \vee b) \wedge a}{b \wedge a}\,\mathsf{ac}\!\downarrow}{a \wedge b}\,\mathsf{co}\!\downarrow \qquad \dfrac{\dfrac{(b \vee b) \wedge a}{a \wedge (b \vee b)}\,\mathsf{co}\!\downarrow}{a \wedge b}\,\mathsf{ac}\!\downarrow$$

are the proofs morally different?

Inference steps seen as rewriting: one redex is inside another

As proof-terms:

$$(\mathsf{ac}\!\downarrow \mid \mathsf{id}).\mathsf{co}\!\downarrow \qquad \mathsf{co}\!\downarrow.(\mathsf{id} \mid \mathsf{ac}\!\downarrow)$$

What do we want to write?

From wired deduction

We use standard symbols for and and or: $\wedge$ and $\vee$.

We drop the notion of wire.

$\Longrightarrow$ Atomic contraction is symmetric: $\dfrac{a \vee a}{a}$

Pipes become tubes (written $x, y, z, \ldots$ rather than $A, B, \ldots$)

Syntax of formulae:

$$A, B, C, \ldots ::= \mathsf{f} \mid \mathsf{t} \mid a \mid (A.B) \mid (A \vee B) \mid (A \wedge B) \mid x_B^A$$

x_B^A = any formula along the tube x
where x corresponds to some derivation from A to B

Where is this derivation recorded?...

Inference rules

$\ldots$ kept in a environment = set of pairs $x : R$

R: derivation taking place on tube x

Using the tube in the rest of the proof

Starting the tube:

$$\frac{\Gamma \vdash A \stackrel{R}{\Longrightarrow} B}{\{\Gamma, x : R\} \vdash A \stackrel{x\triangleright}{\Longrightarrow} x}$$

Ending the tube:

$$\frac{\Gamma \vdash A \stackrel{R}{\Longrightarrow} B}{\{\Gamma, x : R\} \vdash x \stackrel{\triangleleft x}{\Longrightarrow} B}$$

Syntax of proofs:

$$R, S, T, \ldots ::= \mathsf{id} \mid \rho \mid (R \,|\, S) \mid (R.S) \mid x \triangleright \mid \triangleleft\, x$$

Linearity

use of tubes is linear

Type B-bureaucracy in presence of tubes with branching?

Example:

$$\dfrac{B \vee B}{\dfrac{B}{C}\,\Vert\, R}\,\mathsf{c}\!\downarrow \qquad \equiv^{?} \qquad \dfrac{\dfrac{B}{C}\,\Vert\, R \quad \vee \quad \dfrac{B}{C}\,\Vert\, R}{C}\,\mathsf{c}\!\downarrow$$

or equivalently: $\mathsf{c}\!\downarrow\!.R \equiv^{?} (R \,|\, R).\mathsf{c}\!\downarrow$
what with $(R \,|\, R').\mathsf{c}\!\downarrow$ as in:

$$\dfrac{\dfrac{(A \wedge B) \vee (B \wedge A)}{(A \wedge B) \vee (A \wedge B)}\,\mathsf{co}\!\downarrow}{A \wedge B}\,\mathsf{c}\!\downarrow$$

i.e. $(\mathsf{id} \,|\, \mathsf{co}\!\downarrow).\mathsf{c}\!\downarrow$

similarly:

$$\dfrac{\dfrac{\mathsf{t}}{(A \wedge B) \vee (\overline{A} \vee \overline{B})}\,\mathsf{i}\!\downarrow}{(B \wedge A) \vee (\overline{A} \vee \overline{B})}\,\mathsf{co}\!\downarrow \quad \equiv^{?} \quad \dfrac{\dfrac{\mathsf{t}}{(B \wedge A) \vee (\overline{B} \vee \overline{A})}\,\mathsf{i}\!\downarrow}{(B \wedge A) \vee (\overline{A} \vee \overline{B})}\,\mathsf{co}\!\uparrow$$

i.e.: $\mathsf{i}\!\downarrow\!.(\mathsf{co}\!\downarrow \,|\, \mathsf{id}) \equiv^{?} \mathsf{i}\!\downarrow\!.(\mathsf{id} \,|\, \mathsf{co}\!\uparrow)$

or more generally $\mathsf{i}\!\downarrow\!.(R \,|\, S) \equiv^{?} \mathsf{i}\!\downarrow\!.(\mathsf{id} \,|\, \overline{R}.S)$

similarly:

$$\begin{array}{c} \dfrac{\mathsf{f}}{A}\,\mathsf{w}\!\downarrow \\ \overline{\overline{B}}\; R \end{array} \qquad \equiv^{?} \qquad \dfrac{\mathsf{f}}{B}\,\mathsf{w}\!\downarrow$$

i.e. $\mathsf{w}\!\downarrow\!.R \equiv^{?} \mathsf{w}\!\downarrow$

$\Longrightarrow$ safer to consider linear tubes only

A linear system for classical logic

replace $\mathsf{c}\downarrow, \mathsf{i}\downarrow, \mathsf{w}\downarrow$ by

$$\Gamma \vdash a \vee a \xRightarrow{\mathsf{ac}\downarrow} a$$

$$\Gamma \vdash \mathsf{t} \xRightarrow{\mathsf{i}\downarrow} a \vee \overline{a}$$

$$\Gamma \vdash \mathsf{f} \xRightarrow{\mathsf{aw}\downarrow} a$$

and add:

$$\Gamma \vdash (A \wedge B) \vee (C \wedge D) \xRightarrow{\mathsf{m}} (A \vee C) \wedge (B \vee D)$$

$$\Gamma \vdash (A \vee B) \vee (C \vee D) \xRightarrow{\mathsf{m}_0\downarrow} (A \vee C) \vee (B \vee D)$$

$$\Gamma \vdash \mathsf{f} \xRightarrow{\mathsf{w}_0\downarrow} \mathsf{f} \wedge \mathsf{f}$$

idem with the up-fragment for contrapositive types

still complete (Brünnler)

Producing canonical proofs

First phase: we create inner tubes for each constant

Example:

$$\mathsf{co}\!\downarrow \longrightarrow \ (x \triangleright \mid y \triangleright).\mathsf{co}\!\downarrow\!.(\triangleleft y \mid \triangleleft\, x)$$

Second phase: normalisation

acting on term and environment: rules of Formalism A +

loading-right	$\begin{array}{c} \triangleleft x.R \\ \Gamma, x : T \end{array}$	$\longrightarrow$	$\begin{array}{c} \triangleleft x \\ \Gamma, x : T.R \end{array}$	if $R \neq y\triangleright$
loading-left	$\begin{array}{c} R.x\triangleright \\ \Gamma, x : T \end{array}$	$\longrightarrow$	$\begin{array}{c} x\triangleright \\ \Gamma, x : R.T \end{array}$	if $R \neq \triangleleft y$
connexion	$\begin{array}{c} S\{\triangleleft x.y\triangleright\} \\ \Gamma, x : T, y : U \end{array}$	$\longrightarrow$	$\begin{array}{c} S\{\mathsf{id}\}\{y\backslash x\} \\ \Gamma\{y\backslash x\}, x : T.U \end{array}$	
discharge	$\begin{array}{c} S\{x \triangleright . \triangleleft x\} \\ \Gamma, x : T \end{array}$	$\longrightarrow$	$\begin{array}{c} S\{T\} \\ \Gamma \end{array}$	

Third phase: clean up empty tubes

rules of Formalism A +

$$\textsf{clean up} \qquad \begin{array}{c} R \\ \Gamma, x : \textsf{id} \end{array} \longrightarrow \begin{array}{c} R\{x \triangleright \backslash\textsf{id}\}\{\triangleleft x\backslash\textsf{id}\} \\ \Gamma\{x \triangleright \backslash\textsf{id}\}\{\triangleleft x\backslash\textsf{id}\} \end{array}$$

Properties

Subject Reduction (for the 3 phases)

If $\Gamma \vdash A \stackrel{R}{\Longrightarrow} B$ and $\begin{array}{c} R \\ \\ \Gamma \end{array} \longrightarrow \begin{array}{c} R' \\ \\ \Gamma' \end{array}$ then $\Gamma' \vdash A \stackrel{R'}{\Longrightarrow} B$

Termination

Phases 2 and 3 (they include Form A normalisation) terminate

Conjecture

They are confluent (modulo naming of tubes)

Example (1)

$$
\begin{array}{llll}
(\mathsf{ac}\downarrow \mid \mathsf{id}).\mathsf{co}\downarrow & \longrightarrow_{Phase1} & (\mathsf{ac}\downarrow \mid \mathsf{id}).(x \rhd \mid y\rhd).\mathsf{co}\downarrow.(\lhd y \mid \lhd x) & \text{in } x : \mathsf{id}, y : \mathsf{id} \\
 & \longrightarrow_{FormA} & (\mathsf{ac}\downarrow.x \rhd \mid \mathsf{id}.y\rhd).\mathsf{co}\downarrow.(\lhd y \mid \lhd x) & \text{in } x : \mathsf{id}, y : \mathsf{id} \\
 & \longrightarrow_{FormA} & (\mathsf{ac}\downarrow.x \rhd \mid y\rhd).\mathsf{co}\downarrow.(\lhd y \mid \lhd x) & \text{in } x : \mathsf{id}, y : \mathsf{id} \\
 & \longrightarrow_{\mathsf{loading\text{-}left}} & (x \rhd \mid y\rhd).\mathsf{co}\downarrow.(\lhd y \mid \lhd x) & \text{in } x : \mathsf{ac}\downarrow.\mathsf{id}, y : \mathsf{id} \\
 & \longrightarrow_{FormA} & (x \rhd \mid y\rhd).\mathsf{co}\downarrow.(\lhd y \mid \lhd x) & \text{in } x : \mathsf{ac}\downarrow, y : \mathsf{id} \\
 & \longrightarrow_{\mathsf{clean\ up}} & (x \rhd \mid \mathsf{id}).\mathsf{co}\downarrow.(\mathsf{id} \mid \lhd x) & \text{in } x : \mathsf{ac}\downarrow
\end{array}
$$

Example (2)

$$
\begin{array}{llll}
\mathsf{co}\!\downarrow\!.(\mathsf{id} \mid \mathsf{ac}\!\downarrow) & \longrightarrow_{Phase1} & (x \triangleright \mid y\triangleright).\mathsf{co}\!\downarrow\!.(\triangleleft y \mid \triangleleft x).(\mathsf{id} \mid \mathsf{ac}\!\downarrow) & \text{in } x : \mathsf{id}, y : \mathsf{id} \\
& \longrightarrow_{FormA} & (x \triangleright \mid y\triangleright).\mathsf{co}\!\downarrow\!.(\triangleleft y.\mathsf{id} \mid \triangleleft x.\mathsf{ac}\!\downarrow) & \text{in } x : \mathsf{id}, y : \mathsf{id} \\
& \longrightarrow_{FormA} & (x \triangleright \mid y\triangleright).\mathsf{co}\!\downarrow\!.(\triangleleft y \mid \triangleleft x.\mathsf{ac}\!\downarrow) & \text{in } x : \mathsf{id}, y : \mathsf{id} \\
& \longrightarrow_{\mathsf{loading\text{-}right}} & (x \triangleright \mid y\triangleright).\mathsf{co}\!\downarrow\!.(\triangleleft y \mid \triangleleft x) & \text{in } x : \mathsf{id}.\mathsf{ac}\!\downarrow, y : \mathsf{id} \\
& \longrightarrow_{FormA} & (x \triangleright \mid y\triangleright).\mathsf{co}\!\downarrow\!.(\triangleleft y \mid \triangleleft x) & \text{in } x : \mathsf{ac}\!\downarrow, y : \mathsf{id} \\
& \longrightarrow_{\mathsf{clean\ up}} & (x \triangleright \mid \mathsf{id}).\mathsf{co}\!\downarrow\!.(\mathsf{id} \mid \triangleleft x) & \text{in } x : \mathsf{ac}\!\downarrow
\end{array}
$$

Future Work

- Prove confluence
- Characterise normal forms, so far, “it works”
- Write a type checker/synthesiser (try to avoid redundancy)
- Formulate theory for rewrite systems, independently from logic
- ... generalise to non-linear rewrite systems
- ... generalise to rewriting modulo (and deal with equations)
- Make parallel composition associative, and formulate theory with 3-categories
- Look at the connexion with Rewriting Logic, Melliès’ work,...