Getting Formalisms A and B by proof-terms and typing systems

Stéphane Lengrand and Kai Bruennler

11th February 2005

1 Formalism A

Structures, or formulae, are defined as follows:

$$A, B ::= a|\ \overline{a}|\ (A, B)|\ [A, B]$$

We extend the notion of duality to al formulae as follows:

$$\begin{array}{ll} \overline{\overline{a}} & = a \\ \overline{(A, B)} & = [\overline{A}, \overline{B}] \\ \overline{[A, B)} & = (\overline{A}, \overline{B}) \end{array}$$

Now we consider inference rules with one premise and one conclusion, as those of the Calculus of Structures, which can be expessed as pairs of formulae possibly using meta-variables $A, B \dots$ or $\overline{A}, \overline{B} \dots$

ρ, ρ' denote names of inference rules of the shape

$$\rho \frac{S\{A_i\}}{S'\{A_i\}}$$

The A_i represent the structure meta-variables that occur both in the premise and in the conclusion of the rule. Their number is called the arity of ρ.

We now turn each name into a first-order function symbol with its corresponding arity.

For instance in SK, the arity of identity is 0, the arity of switch is 3, the arity of contraction is 1 (but the arity of atomic contraction is 0), ...

In the logical systems that include a down-fragment and an up-fragment, there is a corresponding duality between these function symbols:

If $\rho \frac{S\{A_i\}}{S'\{A_i\}}$ then $\overline{\rho} \frac{\overline{S'}\{A_i\}}{\overline{S}\{A_i\}}$

Obviously ρ and $\overline{\rho}$ have the same arity.

Now we use those function symbols in a first-order syntax that denote derivations:

$M, N ::= 1|\ \rho|\ M \cdot N|\ (M||N)$

We type this syntax using structures as follows:

$$\frac{}{A \xrightarrow{1} A}$$

$$\frac{A \xrightarrow{M} B \qquad B \xrightarrow{N} C}{A \xrightarrow{M \cdot N} C}$$

$$\frac{A \xrightarrow{M} B \qquad C \xrightarrow{N} D}{(A, C) \xrightarrow{M||N} (B, D)}$$

$$\frac{A \xrightarrow{M} B \qquad C \xrightarrow{N} D}{[A, C] \xrightarrow{M||N} [B, D]}$$

$$\frac{}{S\{A_i\} \xrightarrow{\rho} S'\{A_i\}}$$

CoS is obtained by restricting the parallel composition to the form $1||M$ or $M||1$.

The sequential composition is now taken up to associativity, i.e.:

$$(R \cdot S) \cdot T \equiv R \cdot (S \cdot T) \equiv R \cdot S \cdot T$$

Now we consider the following proof normalisation rules:

$$\begin{array}{lll}
(M||N) \cdot (M'||N') & \longrightarrow_{\mathsf{Form\ A}} & (M \cdot M')||(N \cdot N') \\
M \cdot 1 & \longrightarrow_{\mathsf{Form\ A}} & M \\
1 \cdot M & \longrightarrow_{\mathsf{Form\ A}} & M \\
1||1 & \longrightarrow_{\mathsf{Form\ A}} & 1
\end{array}$$

Theorem 1 $\longrightarrow_{\mathsf{Form\ A}}$ *is convergent.*

Example:
In CoS, we would like to have a canonical proof for the bureaucratic equation:

$$(M||1) \cdot (1||N) \equiv (1||N) \cdot (M||1)$$

They are joined in formalism A, since they both reduce to $(M \cdot 1, 1 \cdot N) \longrightarrow (M, N)$

2 A rigorous way of dealing with equations

Strictly speaking, the reduction rules written above are not completely sound: Obviously, they are *meant* to satisfy the property of Subject Reduction (preservation of typing by the reduction), but if structures are considered up to some congruence (like associativity/commutativity of [ ] and ( )), some odd things can happen like:

$$\dfrac{\dfrac{A \xrightarrow{M} A'' \quad B \xrightarrow{N} B''}{(A,B) \xrightarrow{M||N} (A'',B'')} \qquad \dfrac{B'' \xrightarrow{N'} B' \quad A'' \xrightarrow{M'} A'}{(B'',A'') \xrightarrow{N'||M'} (B',A')}}{(A,B) \xrightarrow{(M||N)\cdot(N'||M')} (B',A')}$$

Obviously, the reduced form that makes sense is $(M \cdot M')||(N \cdot N')$, not $(M \cdot N')||(N \cdot M')$

A way to deal with the congruence on structures is to consider that every equation rule (for instance, commutativity of ( )), is itself an inference rule ($com\frac{(A,B)}{(B,A)}$), and is treated as such in the calculus.

This adds lots of function symbols in the calculus (like commutativity, of arity 2), but one can easily hide them from a user and only show in the proof-terms the symbols corresponding to proper inference rules.

Those corresponding to equations are just there to make sure that we're not doing unsound things and to be rigorous in proving theorems.

3 Formalism B

If we consider inference rules as rewrite rules on structures, normal forms of formalism A are free from the bureaucracy that is due to independent (parallel) redexes.

There is still some bureaucracy with the redexes that are inside one another, as in the following example:

$$\mathsf{s} \cdot [M,(1,1)] \equiv (1,[M,1]) \cdot \mathsf{s}$$

Here, M represents a derivation that acts inside the structure that instantiates a meta-variable of the switch. In a sense, the derivation M is *deeper* than the switch.

What the switch does is independent from what the derivation M does, which could occur before or after the switch. We could lift this ambiguity by saying that we prefer deeper derivations to occur earlier, or the contrary, but this is an arbitrary choice that would break the symmetry.

We prefer to keep the ambiguity, so that the first switch manipulates neither the premise nor the conclusion of M but some kind of *thread* that is propagated to the other switch and along which can be performed, at any random moment, the reasoning steps of M.

The thread can be imagined as the invisible (graphical?) connexion between the arguments of the two switches.

Hence we extend the syntax of proofs and formulae (which we then call *types*) by the notion of *thread*. A thread contains a starting type, a concluding type, and carries the derivation that enables to go from the former to the latter. A thread can be opened and closed by two new derivation constructors.

The syntaxes of derivations becomes:

$$M, N ::= 1|\ \rho|\ M \cdot N|\ (M||N)|\ \langle M|\ M\rangle|\ M \triangleright |\ \triangleleft M$$

Sequential composition is still taken up to associativity.

The syntax of types extends that of formulae as follows:

$$A, B := X|\ X \xrightarrow{M} Y$$

$$X, Y ::= a|\ \overline{a}|\ (A, B)|\ [A, B]$$

A, B are types, X, Y are types that are not threads (but could contain one under a conjunctive or disjunctive construction), and formulae are those types that contain no threads at all. We could also recursively add the thread constructor to the syntax of formulae, but then we would need to forbid the starting and ending types to be threads themselves. Indeed, it is bureaucratic to seal a derivation in two consecutive envelops.

The typing rules become

$$\frac{}{A \xrightarrow{1} A} \qquad \frac{A \xrightarrow{M} B \qquad B \xrightarrow{N} C}{A \xrightarrow{M \cdot N} C}$$

$$\frac{A \xrightarrow{M} B \qquad C \xrightarrow{N} D}{(A, C) \xrightarrow{M||N} (B, D)} \qquad \frac{A \xrightarrow{M} B \qquad C \xrightarrow{N} D}{[A, C] \xrightarrow{M||N} [B, D]}$$

$$\frac{}{S\{X_i \xrightarrow{R_i} Y_i\} \xrightarrow{\rho} S'\{X_i \xrightarrow{R_i} Y_i\}}$$

$$\frac{A \xrightarrow{M} B \qquad M \equiv M'}{A \xrightarrow{\langle M} (A \xrightarrow{M'} B)} \qquad \frac{A \xrightarrow{M} B \qquad M \equiv M'}{(A \xrightarrow{M'} B) \xrightarrow{M\rangle} B}$$

Formalism A is obtained by restricting the derivation constructors $\langle R$ and $R\rangle$ to $R = 1$.

The relation $\equiv$ is the transitive reflexive symmetric closure of the following notion of proof transformations:

$$
\begin{array}{lll}
(M||N)\cdot(M'||N') & \longrightarrow_{\textsf{Form B}} & (M\cdot M')||(N\cdot N') \\
M\cdot 1 & \longrightarrow_{\textsf{Form B}} & M \\
1\cdot M & \longrightarrow_{\textsf{Form B}} & M \\
1||1 & \longrightarrow_{\textsf{Form B}} & 1 \\
\\
(T||S)\cdot\langle R & \longrightarrow_{\textsf{Form B}} & \langle((T||S)\cdot R)\cdot(T||S)\triangleright \\
\rho\cdot\langle R & \longrightarrow_{\textsf{Form B}} & \langle(\rho\cdot R)\cdot\rho\triangleright \\
R\rangle\cdot(T||S) & \longrightarrow_{\textsf{Form B}} & \triangleleft(T||S)\cdot(R\cdot(T||S))\rangle \\
R\rangle\cdot\rho & \longrightarrow_{\textsf{Form B}} & \triangleleft\rho\cdot(R\cdot\rho)\rangle \\
R\rangle\cdot\langle S & \longrightarrow_{\textsf{Form B}} & \triangleleft S\cdot R\triangleright \\
\langle R\cdot S\rangle & \longrightarrow_{\textsf{Form B}} & R \\
\langle R\cdot S\rangle & \longrightarrow_{\textsf{Form B}} & S \\
\\
R\triangleright\cdot S\rangle & \longrightarrow_{\textsf{Form B}} & (R\cdot S)\rangle \\
\langle S\cdot\triangleleft R & \longrightarrow_{\textsf{Form B}} & \langle(S\cdot R) \\
R\triangleright\cdot\triangleleft S & \longrightarrow_{\textsf{Form B}} & \triangleleft S\cdot R\triangleright
\end{array}
$$

In addition to these rules, there are the rules that allows the propagation of $\triangleleft$ and $\triangleright$ through constants, like:

$$\mathsf{s}\cdot((\triangleleft R\cdot S)||T) \longrightarrow_{\textsf{Form B}} ((\triangleleft R||1)||1)\cdot\mathsf{s}\cdot(S||T)$$

Finally, I would expect a notion of proof-net to be the graphical version of that, a bit like interaction nets are a graphical version of lambda-calculus...