SOME NEWS ON SUBATOMIC LOGIC

Alessio Guglielmi (TU Dresden and University of Bath)
28.7.2005

After our experience with the calculus of structures (CoS) [WS], we observed that the inference rule schemes we need are of very limited variety. A striking example is Straßburger's local system for linear logic ([LS], p. 12). The largely prevailing scheme there is that of switch and medial, in versions for binary, unary and nullary logical relations.

We also observed that the behaviour of seemingly different inference rules in technical arguments like splitting is very similar, for example the behaviour of an atomic contraction doesn't differ from that of a switch.

All of this suggests that it should be possible to find a more abstract treatment of the various phenomena, which possibly reveals some sort of unity behind the CoS scenes. What I call subatomic logic is, in my opinion, the way to reveal this unity.

The main idea of subatomic logic is to consider atoms as non-primitive objects: they actually are logical relations like disjunction and conjunction, and the ultimate components of the language are units. By doing this, we see that there is *only one* inference rule, called *hyper-switch*, and this rule is sufficient for creating completely local deductive systems in CoS (and in all other formalisms more abstract than CoS, like A, B, wired deduction, etc.).

Let me stress the fact that all of this happens at the *logical* level, it's not the business of formalisms: we extend the logics such that the deductive systems defining them can be designed and analysed with the benefits of just dealing with one inference rule. We see here only classical propositional logic, but other logics I defined this way are very similar, so the case is representative.

I would also add that the point here is not to collect all inference rules together and artificially make a \`big´ inference rule scheme. On the contrary, we get a big simplification.

In the end, I aim at a very general (but concrete), *simple* explanation of cut elimination for most logics. Cut elimination is a purely syntactic, combinatorial phenomenon, determined by the shape of inference rules. Since with subatomic logic we get *one* inference rule for *all* logics, it should be possible to get cut elimination for all of them at once.

Last, but not least, this gives us a completely general design criterion for inference systems, not just in CoS but for all deep inference formalisms.

The idea shown below does not explain everything. It does explain why the hyper-switch rule works, but it doesn't explain why it suffices. However, this is a big improvement for me, since so far I was working with this stuff feeling on completely slippery grounds, and this is not the case any more.

The idea

Let's consider a derivation, for example

```
      b
------------
 ([a -a] b)
------------ .
 [(a b) -a]
```

We could consider it as the *superposition* of the four derivations we obtain from assigning the atoms a and b with the logical units **f** and **t**:

```
     f                t                f                   t
------------     ------------     ------------        ------------
 ([f -f] f)       ([f -f] t)       ([t -t] f)          ([t -t] t)
------------ ,   ------------ ,   ------------  and   ------------ .
 [(f f) -f]       [(f t) -f]       [(t f) -t]          [(t t) -t]
```

We could write the same in a more compact way as in the following derivation:

```
       bft
-----------------
 ([aft atf] bft)
----------------- ;
 [(aft bft) atf]
```

here, we suppose that a and b are *binary*, *self-dual*, *non-commutative* logical relations, written in Polish notation (i.e., the relation symbol precedes the arguments). An *assignment* for a derivation is the choice, for every atom and consistently across the derivation, of either \`left´ or \`right´. An inference rule is *sound* if it is sound for every assignment, for the usual boolean semantics in the case of classical logic.

What is this good for? Let's take an identity axiom, or interaction, and let's write it the \`subatomic way´:

```
 a [ft] [tf]
------------- .
  [aft atf]
```

This is indeed the identity, because the premiss is equivalent to **t** no matter the assignment. The advantage is that the rule above is no different from a medial, which, I recall, for the case of conjunction and disjunction is

$$\frac{[(RU)\ (TV)]}{([RT]\ [UV])}\ .$$

Another example: contraction. This is another `medial´:

$$\frac{[a\mathbf{ft}\ a\mathbf{ft}]}{a\ [\mathbf{ff}]\ [\mathbf{tt}]} \quad \text{or} \quad \frac{[a\mathbf{tf}\ a\mathbf{tf}]}{a\ [\mathbf{tt}]\ [\mathbf{ff}]}\ .$$

Eventually, all inference rules will be derivable for *hyper-switch*, a general form of switch, which in turn is more general than medial. The specificity of identity and cut completely disappears, the cut is just the up fragment of systems, that is, it's the dual of hyper-switch.

Some General Definitions and Observations

Notation: R, T, U and V are variables for structures. [] denotes disjunction, () denotes conjunction and a, b, ... denote atoms. We write atoms in Polish notation and disjunction and conjunction as in CoS. We denote units by **f** and **t**. The language is freely built over variables and units by disjunction, conjunction and atoms. For example,

$$(a\ R\ [TU]\ [V\mathbf{t}])$$

denotes

$$(a(R,(T \vee U)) \wedge (V \vee \mathbf{t}))\ .$$

<u>Rule</u>

There is only one rule in the down fragment of any CoS deductive system for any logic amenable to `subatomic treatment´, we call it the *hyper-switch*:

$$\mathsf{hs}{\downarrow}\ \frac{\beta\ \alpha RU\ \alpha TV}{\alpha\ \beta RT\ \gamma UV}\ , \qquad \text{for } \alpha\text{-}\beta\text{-}\gamma \text{ in C},$$

where C is a given set of triples α-β-γ. This set is chosen such that the rules generated are sound and the system is complete.

For example, the triple ()-[]-[] gives a medial:

$$\mathsf{hs}{\downarrow}\ \frac{[]\ ()RU\ ()TV}{()\ []RT\ []UV}\ , \quad \text{or, better,} \quad \mathsf{hs}{\downarrow}\ \frac{[(RU)\ (TV)]}{([RT]\ [UV])}\ ,$$

where we don't use the Polish notation for [] and ().

Clearly, there exists also an hs↑ rule, and cut elimination consists in showing that hs↑ is admissible.

<u>Wild and Tame Formulae</u>

Weird things can happen with the hs↓ rule: for example, the triple a-b-[], which is sound for classical logic, generates the rule

```
       b aRU aTV
hs↓ ------------- .
       a bRT [UV]
```

Of course, no formula in classical logic corresponds to the conclusion of such a rule, because there is no concept of `atom inside another atom´. However, this shouldn't worry us, because we can prove a conservativity result which says that formulae which are pathological for classical logic never interfere with provability of the normal ones.

In fact, let us say that a formula (or structure) is *wild* if one of its atoms occurs in the scope of another atom, and *tame* otherwise. Clearly, tame formulae are all and only the formulae of classical logic, when we normalise the boolean expressions that might appear in the scope of atoms. For example,

a [**tf**] (**ft**)

is tame and corresponds to -a. The following is just a simple observation:

Proposition If the conclusion of an hs↓ rule is tame, then so is the premiss.

This means that, even if the deductive systems we generate from the hs↓ rule prove much more than the normal ones in non-subatomic logic, they correspond to the normal ones on tame formulae.

Classical Propositional Logic

To specify a logic, we need three ingredients: 1) the behaviour of its units, 2) the behaviour of its logical relations wrt associativity and commutativity, 3) the set of sound α-β-γ triples for hs↓. In the case of classical logic, these are:

<u>1) Units</u>

1 [**f**R] ≡ R ,
2 (**t**R) ≡ R ;

3 (**ff**) ≡ **f** ,
4 [**tt**] ≡ **t** .

We will always use these equations explicitly in the rules 1/, 1\, etc. (see below).

<u>2) Associativity and Commutativity</u>

[R [TU]] ≡ [[RT] U] ,
(R (TU)) ≡ ((RT) U) ,
a R aTU ≡ a aRT U , for every atom a;

[RT] ≡ [TR] ,
(RT) ≡ (TR) .

We will always use these equations implicitly.

<u>3) Set of Triples</u>

This is a minimal, possible choice:

```
C = { []-a-a    ,
      []-()-[]  ,
      a-[]-[]   ,
      ()-[]-[]  ,
      ()-()-a  } .
```

<u>Soundness and Completeness</u>

It is straightforward to verify that C only generates sound rules. Completeness follows from observing that the rules of system KS can be so derived:

```
                                   t
                    2\;2\;2\ ------------
                              (t  t  t  t)
                         hs↓ ------------
                              (t  t att)
                       2/;2/ ----------
                                  att
                       1\;1\ --------------
      t                          a [ft] [tf]
  ---------     →         hs↓ -------------- ;
   [a -a]                       [aft atf]

                              [aft aft]
                   hs↓ --------------
                              a [ff] [tt]
                      4/ --------------
    [a a]                      a [ff] t
   --------     →         1/ -----------   (similar for atf);
      a                          aft
```

```
                                         f
                  2\;2\;2\ ------------
                               (t f t t)
                       hs↓ ------------
      f                        (t t aft)
     ---    →      2/;2/ ----------     (similar for atf);
      a                          aft

                               (R [TU])
                      1\ --------------
                             ([Rf] [TU])
                     hs↓ --------------
   (R [TU])                  [(RT) f U]
 -----------   →     1/ ------------- ;
   [(RT) U]                   [(RT) U]

   [(RU) (TV)]                [(RU) (TV)]
 --------------   →    hs↓ -------------- .
   ([RT] [UV])                ([RT] [UV])
```

Alternative Sets of Rules

For propositional classical logic, the maximal set of sound triples is the following:

```
   C´ = { []-[]-[] ,
          []-a-[]  ,
          []-a-a   ,
          []-()-[] ,
          a-[]-[]  ,
          a-a-[]   ,
          a-b-[]   ,
          a-a-a    ,
          b-a-a    ,
          a-()-[]  ,
          a-()-a   ,
          a-()-b   ,
          a-()-()  ,
          ()-[]-[] ,
          ()-a-[]  ,
          ()-a-a   ,
          ()-()-[] ,
          ()-()-a  ,
          ()-()-() } ,
```

where a and b are different atoms. It is straightforward (but tedious) to verify that this is the maximal set of triples that only generate sound rules. Other sound and complete sets of inference rules are, of course, possible.

We can describe C´ in a compact way if we consider logical relations organised in a partial order

```
     ( )
    / |
   a  b  ...
    \ |
     [ ]
```

This means that () > a > [] for every atom a. In addition, an involution ~ is defined such that ~() = [], ~[] = (), ~a = a, ~b = b, ...

In this case, one can see that

$$C' = \{ \alpha\text{-}\beta\text{-}\gamma \mid \beta \geq \gamma \text{ and } (\alpha \geq \beta \text{ or } \alpha > {\sim}\beta \text{ or } \beta \leq {\sim}\gamma)\} \ ,$$

and it is of course possible to give this a semantic justification.

I am still working to these implicit definitions, because their optimal description depends on many logics and, above all, on the specifics of the general cut elimination proof I'm working at. Of course, such specifications are crucial for keeping the cut elimination proof simple and not having to resort to the usual boring analysis of tens of cases, what would partly cancel the benefits of subatomic logic.

<u>Observation</u>

It is important to note that the following equivalences are provable in the system above for classical logic (complete of its up fragment):

$$a\mathbf{ff} \equiv \mathbf{f} \ ,$$
$$a\mathbf{tt} \equiv \mathbf{t} \ .$$

The second equation, from right to left, is proved above in the derivation corresponding to the interaction axiom. It suffices to prove the following:

```
              f
       3\ --------
            (f f)
   2\;2\ ------------
          (t t f f)
     hs↓ ------------
          (t t aff)
    2/;2/ ---------- .
             aff
```

References

[LS] Lutz Straßburger. A local system for linear logic. In M. Baaz and A. Voronkov, editors, *LPAR 2002*, volume 2514 of *Lecture Notes in Artificial Intelligence*, pages 388-402. Springer-Verlag, 2002. URL: http://www.ki.inf.tu-dresden.de/~lutz/lls-lpar.pdf.

Web Site

[WS] http://alessio.guglielmi.name/res/cos.