

The three dimensions of proofs

Yves Guiraud

This document outlines a 3-categorical translation of the proofs of SKS, a deep inference formal system for classical propositional logic. This interpretation identifies two proofs that only differ by bureaucratic considerations: two proofs using the same inference rules, but in different order. Then, the notion of Penrose diagrams is extended to provide 3-dimensional representations for proofs. Finally, the 3-categorical setting allows one to express local transformations of proofs as 4-dimensional computations; this yields the first concrete example of 4-dimensional rewriting and promises new tools to prove normalization and factorisation results.

1 The two dimensions of formulas

This section is about a 2-dimensional translation of SKS formulas, heavily inspired by the one already known for terms, described and studied in (Guiraud 2004a, 2004b).

1.1 The formulas of system SKS

System SKS is a deep inference formal system for proofs of propositional logic (Brünnler 2004). It is one of the formalisms expressed in the calculus of structures style, an alternative to sequent calculus where inference rules can be applied at any depth inside formulas (Guglielmi 2005). Here a slightly alternative definition of SKS is used.

Definition 1.1.1. Let us consider a countable set \mathcal{A} . The set \mathcal{A} of *SKS atoms* is the set of terms built on the variables in \mathcal{A} and the signature $\Sigma_{\mathcal{A}}$ made of one unary operator $\overline{(\cdot)}$, *modulo* the congruence $\equiv_{\mathcal{A}}$ generated by the relation $\overline{\overline{a}} \equiv a$. Then, given another countable set \mathcal{V} , the set \mathcal{F} of *SKS formulas* is the set of terms built on \mathcal{V} and the signature $\Sigma_{\mathcal{F}}$ made of two constants \top and \perp , one unary operator from \mathcal{A} to \mathcal{F} and two binary operators \wedge and \vee , *modulo* the congruence $\equiv_{\mathcal{F}}$ generated by the following relations:

$$\begin{array}{ll} (f \wedge g) \wedge h & \equiv f \wedge (g \wedge h) & (f \vee g) \vee h & \equiv f \vee (g \vee h) \\ f \wedge g & \equiv g \wedge f & f \vee g & \equiv g \vee f \\ \top \wedge f & \equiv f & \perp \vee f & \equiv f \\ \perp \wedge \perp & \equiv \perp & \top \vee \top & \equiv \top \end{array}$$

◆

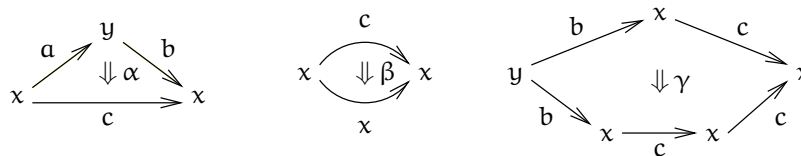
Note that in the original description of system SKS, only closed terms are considered for formulas. However, in order to reduce deduction rules to a *finite* number, it is more convenient to consider all terms. Furthermore, it does not raise any complication, neither does the fact of taking non-linear terms in account.

1. The two dimensions of formulas

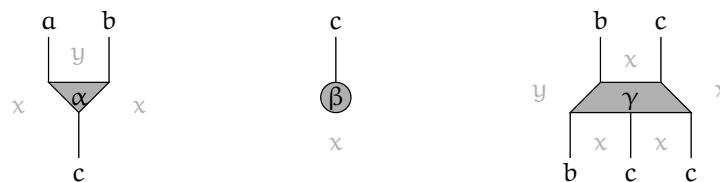
1.2 The 2-polygraph of logical connectives

Here a translation of the aforementioned SKS signature into a 2-polygraph is given. Defining the structure of polygraph can be quite cumbersome, but giving an idea is rather simple: it is a cellular presentation of a 2-graph with no limitation on the shapes of the 2-cells. Such an object is built inductively:

- First, one considers a set Σ_0 of 0-cells, pictured as points in the plane.
- Then, one takes a set Σ_1 of 1-cells, pictured as arrows between these points; formally, the pair (Σ_0, Σ_1) is equipped with a structure of (1-)graph.
- From these data, one can build a (1-)category, which is the graph $(\Sigma_0, \langle \Sigma_1 \rangle)$ of finite paths in (Σ_0, Σ_1) , equipped with the concatenation operation. Two paths starting at the same point and ending at the same point are *parallel*.
- Finally, one chooses a set of 2-cells on the category $(\Sigma_0, \langle \Sigma_1 \rangle)$: this is a set Σ_2 of arrows between parallel paths. They are pictured as directed surfaces, like in the following examples:



For a formal definition of 2-polygraph, see (Burrioni 1993), where these objects were introduced, or the first pages of (Métayer 2003). There is another, more useful representation of 2-cells: *Penrose diagrams*, already used in (Lafont 2003; Guiraud 2004a, 2004b), make 2-cells appear as *circuits*. For example, the three aforementioned 2-cells become:



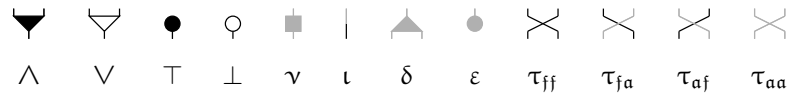
To build these diagrams, the following correspondance is used:

- Each 0-cell is pictured as a label (or colour) for parts of the plane; in the given representations, these labels have been grayed out for sake of clarity.
- Each 1-cell is drawn using a vertical wire, separating the plane in two parts, the leftmost one labelled with the name of its source and the rightmost one with the name of its target; the wire itself is labelled (or coloured) with the name of the 1-cell.
- Each 2-cell is pictured by a small circuit component, a bit like logical gates, with inputs and outputs corresponding to its source and its target.

In most of the concrete examples encountered so far, there has been only one 0-cell and at most two 1-cells. Therefore, the labels in the parts of the plane can be dropped (they are all equal) and one uses only one or two colours for the wires; this simplifies diagrams, but considering the general case does not yield any difficulty since it is easily handled in pictures.

This being defined, let us build a 2-polygraph Σ from the signature defining SKS formulas. We follow the same idea as the one used in (Guiraud 2004a, 2004b) to translate signatures of terms:

- The 2-polygraph Σ has only one 0-cell, denoted by $*$ (no label in parts of the plane).
- Then, we add two 1-cells, denoted by \mathfrak{a} and \mathfrak{f} , respectively corresponding to the sort of atoms (gray wires) and to the sort of formulas (black wires).
- Finally, we consider the following twelve 2-cells:



The interpretation of each 2-cell is given below each diagram. Each one is seen as an operation on formulas, such as $(f, g) \mapsto f \wedge g$ for \wedge . The cells ι and ν stand respectively for the inclusion of atoms into formulas and for the negation on atoms. To these ones, some so-called *ressource management operators* are added; their role is to locally handle duplication, erasement and permutation operations:

- The 2-cell δ is a local duplicator of atoms. Note that, in what follows, duplication of formulas is not necessary; however, one could add a local duplicator of formulas, with no increased difficulty, but at the cost of a few more equations; in that case, the duplicators would be, if necessary, distinguished by notations such as $\delta_{\mathfrak{a}}$ and $\delta_{\mathfrak{f}}$.
- The 2-cell ε is a local eraser of atoms. The same remark applies, eventually leading to the addition of another local eraser $\varepsilon_{\mathfrak{f}}$ for formulas, the first one becoming $\varepsilon_{\mathfrak{a}}$.
- The four remaining 2-cells are local permutations; there is one for each possible pair of colours, but generally all of them are simply denoted by τ .

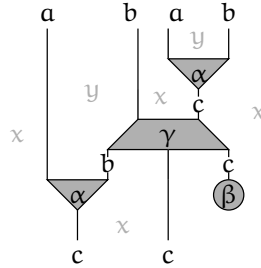
1.3 The 2-category of formulas

In the same way as the signature operators are building blocks for formulas, these circuit components are used to craft something bigger, in which SKS formulas can be interpreted.

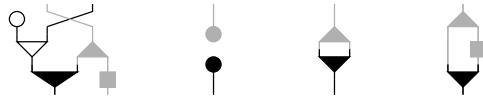
From a 2-polygraph, one considers all the circuits one can build, by plugging together any number of copies of its 2-cells. This is the same idea as the one leading from logical gates to boolean circuits, which are all the constructions one can make using any number of AND-gates, OR-gates, etc.

1. The two dimensions of formulas

For example, with the three aforementioned 2-cells α , β and γ , one can build the following circuit:



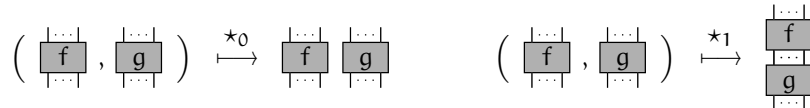
In the case of the polygraph Σ , built from the signature of SKS, one can get the following circuits:



In what follows, we give the following interpretations to these circuits:

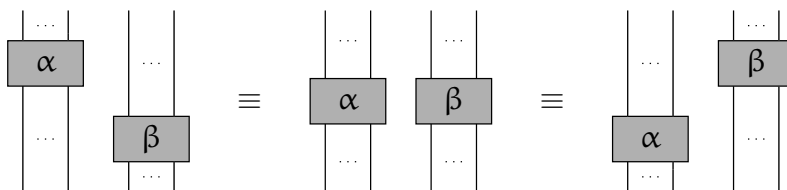
- The first one corresponds to the map $\mathcal{A} \times \mathcal{F} \rightarrow \mathcal{F} \times \mathcal{A}$ sending (a, f) to $((\perp \vee f) \wedge a, \bar{a})$.
- The second one is the constant map $\mathcal{A} \rightarrow \mathcal{F}$ sending every atom to \top .
- The third one is the map $\mathcal{A} \rightarrow \mathcal{F}$ sending a to the formula $a \wedge a$.
- The fourth one is the map $\mathcal{A} \rightarrow \mathcal{F}$ sending a to the formula $a \wedge \bar{a}$.

There are two kinds of *pastings* used to build these circuits: either horizontally or vertically. These operations, called *compositions*, are pictured this way:



The first one, \star_0 , is only defined if the plane labels match (this is always the case here, since there is only one 0-cell). The second one, \star_1 , is only defined if the wires labels match (here, there must be the same number of wires, with the same colours, in the same order). The fact that there are two different ways of pasting circuits together, and only two, is the reason why we say that they are 2-dimensional objects.

These two dimensions are not totally independent. Indeed, the circuits are seen as genuine topological objects. This means that they are to be considered modulo *isotopy*, or homeomorphical deformation: one can deform wires and move components, provided no crossing is created. This identification is generated by the following *local isotopy relations*, given for each pair of components α and β :



Definition 1.3.1. The object $\langle \Sigma \rangle$ consisting of all circuits *modulo* isotopy is called the *free 2-category* generated by the 2-polygraph Σ ; the circuits are the *2-arrows* of $\langle \Sigma \rangle$. If f is one of them, then $s_1(f)$ and $t_1(f)$ respectively denote the source and target 1-arrows of f . \blacklozenge

One can find in (MacLane 1998) more information about 2-categories and (Chang and Lauda 2004) is a really comprehensive survey about the zoology of n -categories. Now, we sketch how SKS formulas are related to the 2-arrows of $\langle \Sigma \rangle$. First, let us build a map from 2-arrows to (families) of formulas.

Since both \mathcal{A} and \mathcal{V} , the sets of atoms and formulas variables, are countable, one is allowed to choose a strict linear order on each set, so that the elements of \mathcal{A} are denoted by a_1, a_2, a_3 , etc. and the elements of \mathcal{V} are denoted by x_1, x_2, x_3 , etc.

We define, for each circuit f , a map $\pi(f)$ acting on atoms and formulas. Let us start by giving its source and target:

- First, one defines $\pi(a) = \mathcal{A}$ and $\pi(f) = \mathcal{F}$.
- Then, one extends $\pi(x \star_0 y) = \pi(x) \times \pi(y)$, for any pair (x, y) of 1-arrows in $\langle \Sigma \rangle$.
- Finally, for any circuit f , $\pi(f)$ has source and target given by:

$$\pi(f) : \pi(s_1(f)) \rightarrow \pi(t_1(f)).$$

Then, we define $\pi(f)$ inductively:

- If f is the identity of a , then $\pi(f)$ is the identity of \mathcal{A} ; both identities are abusively denoted by a and \mathcal{A} .
- If f is (the identity of) f , then $\pi(f)$ is (the identity of) \mathcal{F} .
- $\pi(\wedge) : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{F}$ is defined by $\pi(\wedge)(x, y) = x \wedge y$, and similarly for \vee .
- $\pi(\top) : * \rightarrow \mathcal{F}$ is the constant map \top , and similarly for \perp .
- $\pi(\iota) : \mathcal{A} \rightarrow \mathcal{F}$ is the inclusion of \mathcal{A} into \mathcal{F} .
- $\pi(\nu) : \mathcal{A} \rightarrow \mathcal{A}$ sends each atom a to \bar{a} .
- $\pi(\delta) : \mathcal{A} \rightarrow \mathcal{A} \times \mathcal{A}$ sends a to (a, a) .
- $\pi(\varepsilon) : \mathcal{A} \rightarrow *$ is the terminal map of \mathcal{A} .
- $\pi(\tau_{f,f}) : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{F} \times \mathcal{F}$ send (x, y) to (y, x) , and similarly for the other local permutators.
- If f has the shape $f = g \star_0 h$, then $\pi(f) = (\pi(g), \pi(h))$, the juxtaposition of both maps; this means that $\pi(g)$ acts on the first inputs and $\pi(h)$ on the last inputs.
- If f has the shape $f = g \star_1 h$, then $\pi(f) = \pi(h) \circ \pi(g)$, the composition of $\pi(g)$ and $\pi(h)$.

2. The three dimensions of proofs

Then, one has to check that π is well-defined: it is compatible with the isotopy relation, since the cartesian product is a bifunctor. Finally, one associates a formula $\bar{\pi}(f)$ to each circuit f by application of $\pi(f)$ on some variables:

- One defines $v_n(a) = a_n$, the n^{th} atom variable, and $v_n(f) = x_n$, the n^{th} formula variable.
- One inductively extends v_n to $v_n(x \star_0 u) = (v_n(x), v_{n+1}(u))$, for every 1-cell x and 1-arrow u .
- Finally, one defines $\bar{\pi}(f) = \pi(f)(v_1(s_1(f)))$.

For example, the four aforegiven circuits are respectively sent to the formulas:

$$((\perp \vee x_1) \wedge a_1, \bar{a}_1), \quad \top, \quad a_1 \wedge a_1, \quad a_1 \wedge \bar{a}_1.$$

Hence, we have built a projection $\bar{\pi}$ from circuits to formulas. But, for the moment, we cannot define a canonical section φ , since the projection $\bar{\pi}$ is not compatible with the congruences \equiv_A nor \equiv_F . Indeed, for example, both circuits $\nu \star_1 \nu$ and a are sent to the atom a_1 , since $\bar{a} \equiv_A a$, yet they are two distinct circuits.

Furthermore, $\bar{\pi}^{-1}(\equiv_A)$ and $\bar{\pi}^{-1}(\equiv_F)$ do not constitute the whole kernel of $\bar{\pi}$. For example, both circuits $(\tau \star_1 \tau)$ and $f \star_0 f$ are sent to the pair (x_1, x_2) ; however, they are once again different circuits. These additional equalities are called *resource management equations* and the congruence they generate on circuits is denoted by \equiv_Δ .

There exist more than one possibility to solve this problem, the choice being left to the user. On a first approach, one can translate all the equalities defining \equiv_A and \equiv_F to equalities on circuits; then, add the equalities defining \equiv_Δ ; finally, consider circuits *modulo* the congruence generated by all of these equalities. Then, equivalence classes of circuits would correspond to families of formulas.

But equalities and computational objects rarely live together peacefully. Another idea consists in the replacement of equalities by computations proving these equalities; this point of view is more in adequation with the computational flavour of the considered objects. Then, both (Burroni 1993) and (Baez and Dolan 1998) tell us that this calculus can be well-represented by objects in the dimension above.

2 The three dimensions of proofs

2.1 The proofs of system SKS

We give here a slightly different definition than the original one from (Brünnler 2004); non-closed formulas allow one to remove contexts in the rules, reducing them to a finite number.

Definition 2.1.1. The *inference rules* of system SKS are given by:

$$\begin{array}{llll} \top \longrightarrow a \vee \bar{a} & (x \vee y) \wedge z \longrightarrow (x \wedge z) \vee y & \perp \longrightarrow a & a \vee a \longrightarrow a \\ a \wedge \bar{a} \longrightarrow \perp & (x \wedge y) \vee (z \wedge t) \longrightarrow (x \vee z) \wedge (y \vee t) & a \longrightarrow \top & a \longrightarrow a \wedge a \end{array}$$

2.1. The proofs of system SKS

From left to right, top to bottom, these rules are called *introduction* (or *axiom*), *switch*, *weakening*, *contraction*, *cointroduction* (or *cut*), *medial*, *coweakening* and *cocontraction*. \blacklozenge

From these elementary blocks, one builds *SKS proofs* exactly the same way one builds rewriting paths from rewriting rules: each inference rule α generates a proof step from $C[s(\alpha) \cdot \sigma]$ to $C[t(\alpha) \cdot \sigma]$, for every context C and every substitution σ - see (Baader and Nipkow 1998) or (Guiraud 2004a) for more information about these notions.

Then one can compose proof steps, provided the target of each one matches the source of the next one. Such a path going from a formula u to a formula v is called a *proof (from u to v)*; when $u = \top$, then the path is a *complete proof (of v)*.

Example 2.1.2. Here is a complete proof of $(a \wedge b) \vee (\bar{a} \vee \bar{b})$:

$$\begin{aligned} \top &\equiv_F \top \wedge \top \rightarrow (a \vee \bar{a}) \wedge \top \\ &\rightarrow (a \vee \bar{a}) \wedge (b \vee \bar{b}) \\ &\rightarrow (a \wedge (b \vee \bar{b})) \vee \bar{a} \equiv_F ((b \vee \bar{b}) \wedge a) \vee \bar{a} \\ &\rightarrow ((b \wedge a) \vee \bar{b}) \vee \bar{a} \equiv_F (a \wedge b) \vee (\bar{a} \vee \bar{b}). \end{aligned}$$

The main problem coming from using term-flavoured notations for formulas is that it raises bureaucracy, that can hardly be controlled by equations.

A first example of bureaucracy is illustrated by the aforegiven SKS proof. Indeed, there are two possible proofs from $\top \wedge \top$ to $(a \vee \bar{a}) \wedge (b \vee \bar{b})$, generating two different complete proofs of $(a \wedge b) \vee (\bar{a} \vee \bar{b})$:

$$\begin{array}{ccc} \top \wedge \top & \longrightarrow & (a \vee \bar{a}) \wedge \top \\ \downarrow & & \downarrow \\ \top \wedge (b \vee \bar{b}) & \longrightarrow & (a \vee \bar{a}) \wedge (b \vee \bar{b}) \end{array}$$

But the user may desire to identify these two proofs: indeed, they only differ by the order of application of two introduction rules on different subformulas; so, in essence, they could be considered the same. Yet, they are distinct in the SKS formalism.

For this kind of bureaucracy, equations remain easy to craft. But there is another kind of bureaucracy that can hardly be defined equationally on formulas expressed as terms. Let us assume that F is a proof from u to v and that x and y are two formulas variables; then, one gets two different proofs from $(u \wedge x) \vee y$ to $(v \vee y) \wedge x$:

$$\begin{array}{ccc} (u \wedge x) \vee y & \longrightarrow & (v \wedge x) \vee y \\ \downarrow & & \downarrow \\ (u \vee y) \wedge x & \longrightarrow & (v \vee y) \wedge x \end{array}$$

2. The three dimensions of proofs

Once again, these two proofs could be considered as equal. But here, equations are painful to define. This is because the term notation makes the two different compositions \star_0 and \star_1 appear different, yet they are similar in essence.

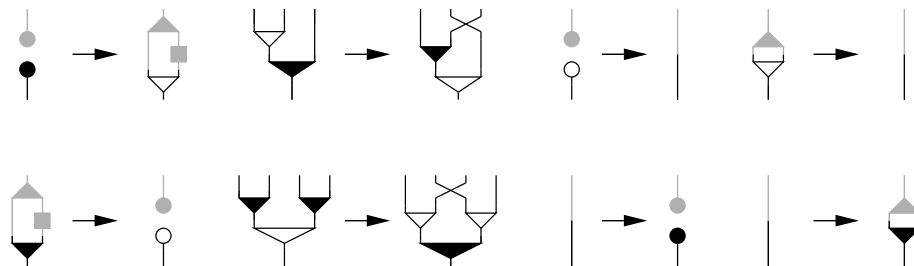
The two-dimensional interpretation of proofs makes it possible to produce automatically the equations corresponding to these two kinds of bureaucracy, provided the proofs are seen as three-dimensional objects.

2.2 The 3-polygraph of inference rules

Here, we associate a 3-polygraph to the inference rules of SKS. Let us give an informal definition of what is a 3-polygraph - the formal one can still be found in (Burroni 1993) or (Métayer 2003).

Definition 2.2.1. A 3-polygraph is a family $\Sigma = (\Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3)$ where $(\Sigma_0, \Sigma_1, \Sigma_2)$ is a 2-polygraph and Σ_3 is an additional set of 3-cells: each one is an arrow between two parallel circuits f and g , built from the 2-polygraph. \blacklozenge

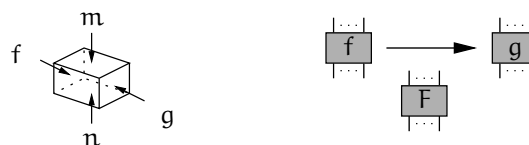
These 3-dimensional cells can be represented as computations from one circuit to another. The ones corresponding to the rules of SKS are the following eight 3-cells:



For the moment, we use this representation for proofs, but another one is used later in order to give a genuine three-dimensional vision of these objects.

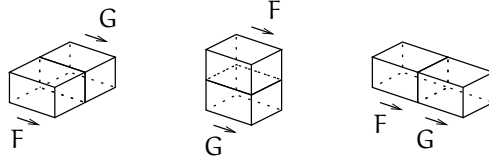
2.3 The 3-category of proofs

These 3-cells are seen as elementary components that are used to build bigger objects, representing (families of) proofs. To give an idea, let us imagine a proof as a block. Here are two different possible representations:

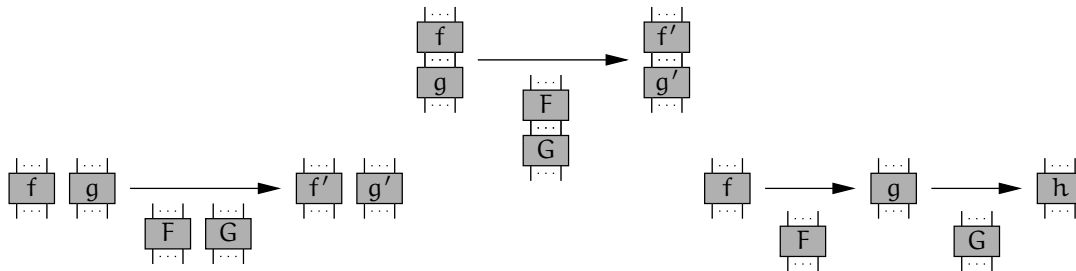


This is a proof F from a circuit f to a circuit g , both having m inputs and n outputs. To compare both representations, one can see the second one as composed of three vertical slices of the block: one before the block (f), one in the middle of the block (F) and one after the block (g).

These blocks can be pasted in three different ways, each one corresponding to one of its dimensions:



If one makes slices for each of these three constructions, one gets:



- The first composition produces a proof $F \star_0 G$ composed of two juxtaposed subproofs (one in each subformula); it is a new composition of proofs, linked to another one revealed by the two-dimensional interpretation of formulas (the juxtaposition of formulas).
- The second one yields a proof $F \star_1 G$ made of one subproof plugged into another one; it is also a new composition corresponding to the composition of formulas.
- The third composition is the usual composition $F \star_2 G$ of proofs, one after the other.

When these constructions are defined, one has to identify some of them along some relations in order to get the 3-arrows of the free 3-category. Among these relations are the *monoidal* ones:

- For any $i \in \{0, 1, 2\}$ and any 3-arrows F, G, H , the following *associativity relation* holds whenever one of its members is defined:

$$(F \star_i G) \star_i H = F \star_i (G \star_i H).$$

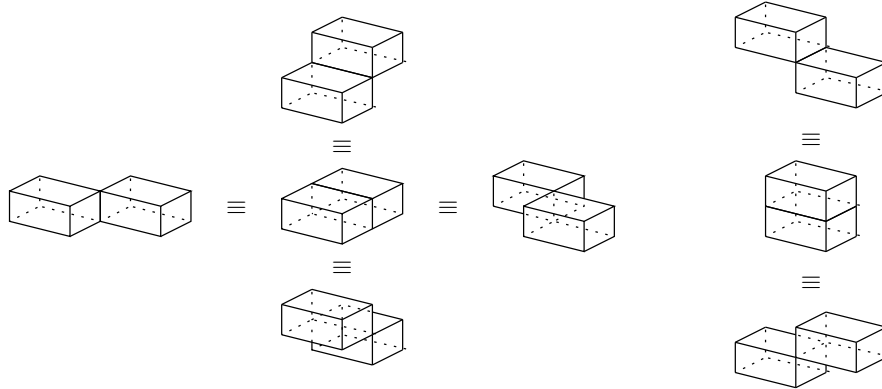
- For any 3-arrow F from f to g , with m inputs and n outputs, the following *unit relations* hold:

$$F \star_0 * = F = * \star_0 F, \quad F \star_1 n = F = m \star_1 F, \quad F \star_2 g = F = f \star_2 F.$$

To these families of equations, one adds the *isotopy relations*. Indeed, 3-dimensional constructions are one again seen as topological objects; this implies that they are identified *modulo* homeomorphic deformation, or isotopy.

3. Isotopy and bureaucracy

The isotopy classes are given by the following equations, given graphically:



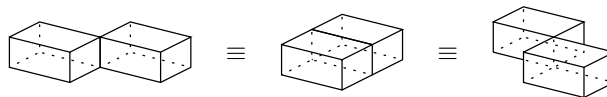
There are three types of isotopy, two of them being detailed thereafter.

3 Isotopy and bureaucracy

Isotopy relations on 3-dimensional arrows are divided into three families. Two of them are deeply linked with the two types of bureaucracy one encounters when dealing with SKS proofs or, more generally, with proofs expressed in the calculus of structures; each one is detailed thereafter. The third family comes from the only isotopy type one gets in dimension 2 (on circuits): it is a degenerated form of this one.

3.1 Isotopy and bureaucracy A

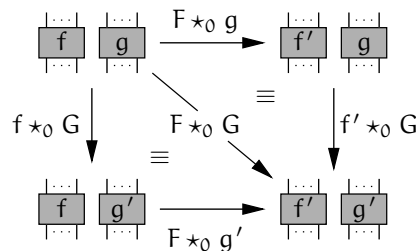
The first family of isotopy relations is given graphically by:



In equational form, it is expressed as:

$$(F \star_0 g) \star_2 (f' \star_0 G) \equiv F \star_0 G \equiv (f \star_0 G) \star_2 (F \star_0 g'),$$

for any $F : f \rightarrow f'$ and $G : g \rightarrow g'$. Thus, this isotopy identifies the three following proofs:



Now, let us see how this is linked to *type A* bureaucracy: in SKS, the applications of two proofs in two different subformulas must be done one after the other; and the two possibilities correspond to two different proofs.

For example, as stated earlier, one gets two different SKS proofs from $\top \wedge \top$ to $(a \vee \bar{a}) \wedge (b \vee \bar{b})$, depending on which of a or b is introduced first. In SKS this yields two different proofs; if one wants to identify them, then a canonical representative has to be chosen, which is not possible, for either choice would be arbitrary.

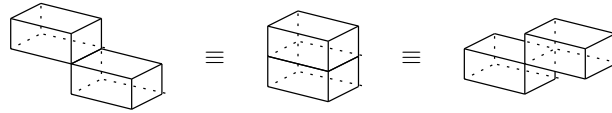
But, in the 3-category, a third proof exists, corresponding to the simultaneous introduction of a and b : this one can be chosen as a canonical representative.

Even more important, one may want to have equations describing this type of bureaucracy. This is not really hard to express in the SKS style, but it is definitely easier in the 3-dimensional framework.

Furthermore, it is only a SKS feature that the other type of bureaucracy appears to be totally different in essence than the *type A* one. Indeed, as described in the next paragraph, the two types are completely similar in the 3-dimensional language.

3.2 Isotopy and bureaucracy B

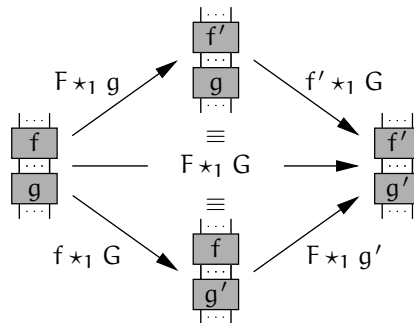
The second family of isotopy relations is:



And, in equational form:

$$(F \star_1 g) \star_2 (f' \star_1 G) \equiv F \star_1 G \equiv (f \star_1 G) \star_2 (F \star_1 g'),$$

for any $F : f \rightarrow f'$ and $G : g \rightarrow g'$, provided the proof $F \star_1 G$ is defined. Thus, this isotopy type identifies the three following paths:



This is deeply linked to *type B* bureaucracy: in SKS, one cannot apply at the same time one proof inside another one; one of them must be done before the other, yielding two different proofs.

3. Isotopy and bureaucracy

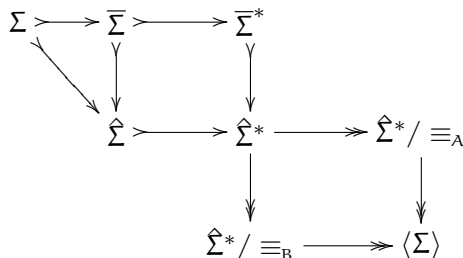
For example, let us consider a proof F from f to g , together with two variables x and y . Then, one gets two different SKS proofs from $(f \vee x) \wedge y$ to $(g \wedge y) \vee x$, depending on whether F is applied before or after the switch rule. Once again, there is no canonical choice between them, whereas in the 3-category there is one.

But, the main interest lies elsewhere: in the SKS style, it is quite hard to express equationally this bureaucracy type B, while in the 3-category, it is not painful at all, since both bureaucracy types are completely similar; indeed, each one corresponds to an isotopy relation, between \star_2 and \star_0 for type A, between \star_2 and \star_1 for type B.

3.3 Some geography

There is a point in which higher-dimensional rewriting excels: the classification of equational and deductive theories. And this is also the case for the calculus of structures and its offsprings, formalisms A and B - when concerned with classical propositional logic, these three are SKS, SKS *modulo* bureaucracy A and SKS *modulo* bureaucracies A and B. Formalism A is defined in (Guglielmi 2004a) and formalism B is in (Guglielmi 2004b).

Existing descriptions of the three formalisms look quite different, while they can be viewed as parts of a bigger scheme. Let us consider the 2-polygraph of formulas and build the following commutative diagram of 3-polygraphs over the free 2-category of formulas:



Objects in this diagram are encountered when constructing, step by step, the free 3-category generated by a 3-polygraph. Here is an informal description of them:

- One starts with a set Σ of 3-cells: these are rewriting rules on circuits.
- From these, one can consider all the rules, applied in any context, which yields $\bar{\Sigma}$, the set of one-step sequential reductions.
- Alternatively, one can consider all the rules applied in any existing context and possibly in parallel to build $\hat{\Sigma}$, the set of one-step parallel reductions.
- Then, one considers the reduction paths generated by $\bar{\Sigma}$: this produces the set $\bar{\Sigma}^*$ of sequential reductions. This is where the calculus of structures (here SKS) lives.
- Alternatively, the paths generated by $\hat{\Sigma}$ give the set $\hat{\Sigma}^*$ of parallel reductions. This is the biggest set of 3-arrows one can build from Σ : there, bureaucracy is at its highest level, since all the described

proofs differing by the order of application of subproofs are distinguished; furthermore, there is at each time a third possible proof, consisting in the simultaneous application of both subproofs.

- Then one starts the quotients by isotopy relations. The first possibility is to quotient by the first family of isotopy relations, corresponding to bureaucracy type A. This yields $\hat{\Sigma}^* / \equiv_A$, where lives formalism A.
- As an alternative, one can instead quotient by the isotopy relations corresponding to bureaucracy type B, to get $\hat{\Sigma}^* / \equiv_B$ which has no equivalent in calculus of structures-derived formalisms.
- Finally, doing both quotients, one gets the free 3-category $\langle \Sigma \rangle$ generated by Σ , where all the bureaucracy is killed. This is where formalism B lives.

This diagram is indeed a map of where known formalisms are located. But it also encompasses still unknown formalisms that could prove to be useful, like $\hat{\Sigma}^*$ or $\hat{\Sigma}^* / \equiv_B$. This is an example of the freedom higher-dimensional rewriting lets to the user in the exact design of the proofs he wants to consider.

Another example of freedom is given later about the possibilities offered to the user for handling the equations between formulas.

4 A 3-dimensional representation for proofs

This section is a first attempt at representing proofs in 3 dimensions, so that one can view them as the genuine 3-dimensional objects they are.

4.1 The theoretical idea

In order to represent 2-arrows, Penrose diagrams are really convenient; they make 2-arrows appear as circuits, using the following scheme:

- Each 2-dimensional cell is pictured as a vertice in a graph (a 0-dimensional object).
- Each 1-dimensional cell is drawn as an edge in a graph (a 1-dimensional object).
- Each 0-dimensional cell is represented by a part of the plane which boundaries are the edges of the graph (2-dimensional objects).
- Then, the vertices and edges of the graph are thickened until they are 2-dimensional; note that in the circuit representation, wires are not thickened to make drawing easier, but they should be for sake of coherence.

The application of a similar process to a 3-dimensional arrow gives:

- Each 3-dimensional cell is a point.
- Each 2-dimensional cell is a line (either open or between two points).
- Each 1-dimensional cell is a surface (either open or with a line as a boundary).

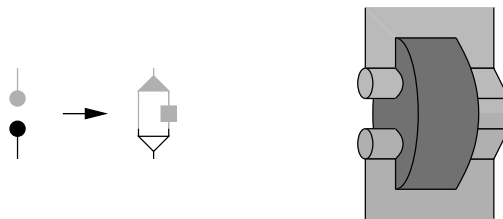
4. A 3-dimensional representation for proofs

- Each 0-dimensional cell is a volume lying between surfaces.
- Finally, every object is thickened, if necessary, until it gets 3-dimensional.

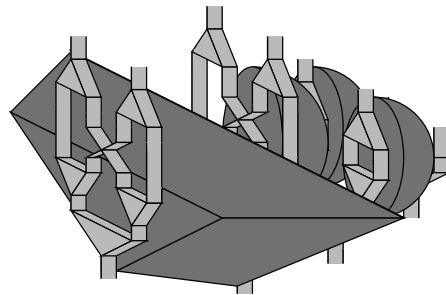
This associates *3-dimensional Penrose diagrams* to 3-arrows.

4.2 One glance at three-dimensional proofs

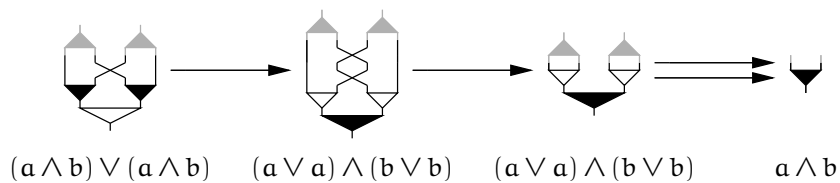
This is time to draw a 3-dimensional proof. First, let us make a Penrose diagram for a rule; the rewriting rule style is also given:



Then, Penrose diagrams for proofs are pastings of such 3-dimensional blocks:

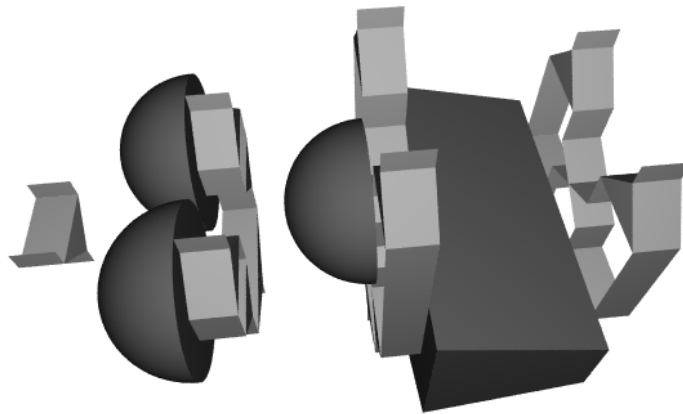
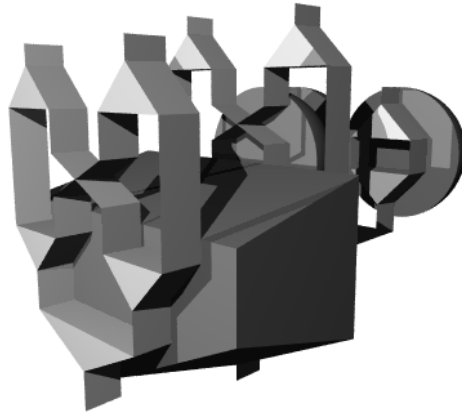


In order to understand how this object is built (and what lies behind some opaque volumes), one can make vertical slices of this object, to produce the following rewriting-style proof:



Since this representation uses only a fake third dimension, one could prefer to use a 3-dimensional modeler. This has many advantages, such as being able to turn around the object and make snapshots from different points of view. For example, two views of the same proof, presented in the next page, were generated using the freely available software POV-Ray¹.

¹<http://www.povray.org>



Two remarks shall be made about the representations:

- On the 3-dimensional ones, less emphasis has been put on circuits, so that different types of wires or different operators appear the same while they should not.
- Some surfaces are not drawn, only for sake of clarity, but these objects should appear somewhat more closed.

This part is quite new and some work will be necessary to produce nice and more usable representations, so that the third dimension can provide more insight on what kind of objects proofs are.

5 The twilight zone

When the third dimension gets involved, one can ask whether this dimensional increase will stop or not. The answer is quite simple: no. Indeed there are, at least, two good reasons to proceed.

The first one is total abstract nonsense. In category theory, there is a proverb saying: *when one wants to study some objects, one should rather study their morphisms*. Let us add that morphisms between 3-arrows are 4-arrows in a 4-category.

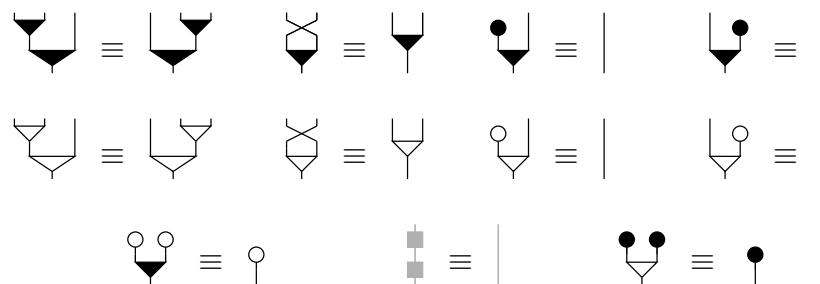
More concretely, there are two kinds of examples that give rise to 4-dimensional arrows: equations between formulas and local transformations on proofs. This section is about a short glance at these two issues.

5.1 Equations between formulas

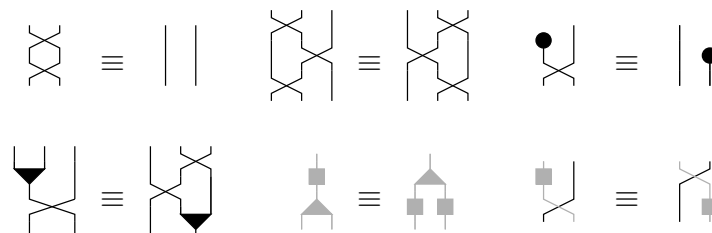
Previously, equations between formulas have been left aside. It has been said that they can be translated as equations between circuits. This is just one possibility, the higher-order rewriting framework allowing one to *choose* between many possible considerations. Here are three of them, but one can at least take any desired combination of them.

Equations are equations. The first possibility is, as stated before, to translate equations between formulas into equations between circuits. In that case, one considers circuits *modulo* two families of equations.

The first one is a faithful translation of the equations on formulas, so that, for example, one can recognize associativity of \wedge and \vee among them:



The second family purpose is to give the resource management operators their real meaning, so that, for example, δ really is a local duplicator; among others, one gets the following equations:



From equations to 3-dimensional isomorphisms. Rather than considering equations on formulas as equations on circuits, one can treat them as invertible computations. Indeed, equations are often clashing with computational considerations, so that, whenever possible, they are replaced by local computations.

Hence, one could replace the two aforementioned families of equations by two families of invertible 3-cells. For example, the equation enforcing the associativity of \wedge is split into two 3-cells:

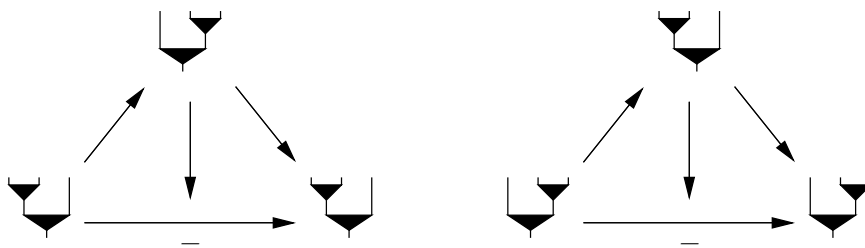


Then, in order to ensure that they are 3-dimensional isomorphisms, one adds equations between proofs: both possible composites are equal to the corresponding identity. Hence, this leaves no equation between objects of dimension 2, while two of them appear between objects of dimension 3 for each equation on formulas.

From equations to 4-dimensional computations. There is no reason to stop the process of lifting up equations. In order to achieve this, the pairs of 3-dimensional cells replacing equations are kept, but equations between 3-dimensional composites are lifted up.

Hence, instead of considering commutative diagrams between 3-dimensional arrows, one defines 4-dimensional cells: each one represents a *computation* from one composite to the identity 3-cell.

For example, the equation about the associativity of \wedge is finally replaced by two 3-cells, together with the following two 4-cells:

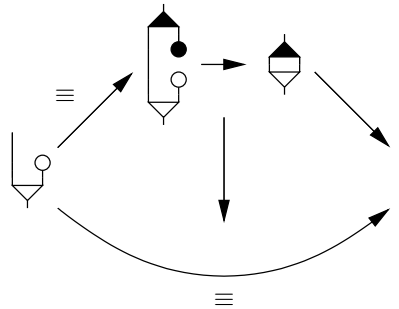


5.2 Local computations on proofs

The next example of 4-dimensional cells is in fact a generalization of the former one. Indeed, it arises whenever one wants to compute normal forms for proofs, *modulo* some user-specified equations. This encompasses the former example, since these equations can be the ones stating that two 3-cells are inverse one another.

5. The twilight zone

As an example of generalized computation, the following 4-cell can be introduced in order to simplify proofs with a weakening followed by a contraction, both acting on the same atom:



In fact, any local computation on proofs can be replaced by a 4-cell. All the 4-cells being given, the computations they generate are the 4-dimensional arrows of a free 4-category.

5.3 Some temporary relief

This point of view immediately arises the following question: how can one use the fact that these computations are 4-dimensional objects? This comes with the subsidiary question: how can one represent 4-dimensional objects? In fact, this is not necessary at this point.

To explain this answer, let us step back by one dimension. Term rewriting is about some properties (termination and confluence) of computations on 2-dimensional objects. While considering the whole 2-dimensional structure of terms is really useful, the computations need not be seen as genuine 3-dimensional objects: the only purpose of doing so would be to identify reduction paths *modulo* bureaucracy. But term rewriting is not concerned with the classification of reduction paths (only their existence) and neither termination nor confluence are modified by bureaucracy.

Then comes proof theory which, with the higher-dimensional point of view, studies 3-dimensional objects, or rather computations between them. Hence, with the same arguments as above, considering the whole 3-dimensional structure of proofs shall prove to be useful. But the four dimensions of computations on proofs are not involved if one only wants to prove termination or confluence of proof normalization processes.

In conclusion, if it is only about (normalization of) proofs, then one can live with rewriting paths on 3-dimensional arrows. But when times will come when the classification of rewriting paths on proofs is concerned, then the fourth dimension will be useful.

For example to manage the six types of bureaucracy lurking in dimension 4.

6 Future directions

This paper presents a higher-dimensional rewriting point of view for the deep inference system SKS. Its main benefit is to provide a uniform setting for many possible systems, depending on what the user wants to emphasize; indeed, much freedom is left on how to consider bureaucracy or how to see equations.

Furthermore, higher-dimensional rewriting provides a common view on equations and computations between proofs: they are seen as 4-dimensional cells between proofs. So one just has to choose the local computations he wants to study, then the theory can be used to see if the generated calculus is terminating or not, confluent or not.

However, some work will be necessary here to provide the required tools, such as a recipe to craft termination orders like the one in (Guiraud 2004b) for 3-dimensional rewriting. Another tool will concern the study of 4-dimensional critical pairs; this will be an adaptation of one that is still under development for 3-dimensional critical pairs and will be described in a subsequent paper.

Aside from these computational issues, proofs seen as 3-dimensional objects are naturally equipped with a geometric representation, using 3-dimensional Penrose diagrams. The links between these pictures and proof nets still have to be explored. For the moment, we can at least say that the proposed 3-dimensional representations provide a completely different way to look at proofs.

Here, only system SKS has been mentioned, mainly because it is smaller than, for example, system SLLS for linear logic (Straßburger 2003). However, it is not very dangerous to conjecture that this translation shall work as well for SLLS.

A bit more risky is the assertion that the translation should also work with any deep inference-style formalism, provided there is no quantifier in the syntax of the formulas. Indeed, binders such as quantifiers or the abstraction in the λ -calculus are yet to be understood in the higher-dimensional point of view.

Finally, a bigger conjecture would be that such a translation can be made for sequent-style proofs (still with no binder allowed in the formulas). If this result holds, then one will be able to say that, in essence, proof theory studies 4-dimensional rewriting systems.

References

Franz Baader and Tobias Nipkow, *Term rewriting and all that*, Cambridge University Press, 1998.

John Carlos Baez and James Dolan, *Categorification*, ArXiv preprint, 1998.

Kai Brännler, *Deep inference and symmetry in classical proofs*, Logos Verlag, 2004.

Albert Burroni, *Higher-dimensional word problems with applications to equational logic*, Theoretical Computer Science 115(1), 1993.

Eugenia Chang and Aaron Lauda, *Higher-dimensional categories: an illustrated guide book*, 2004.

Alessio Guglielmi, *Formalism A*, note, 2004(a).

Alessio Guglielmi, *Formalism B*, note, 2004(b).

Alessio Guglielmi, *Deep inference and the calculus of structures*, project report, 2005.

Yves Guiraud, *Présentations d'opérades et systèmes de réécriture [Operad presentations and rewriting systems]*, thèse de doctorat, Université Montpellier 2, 2004 (a).

Yves Guiraud, *Termination orders for 3-dimensional rewriting*, submitted preprint, 2004 (b).

Yves Lafont, *Towards an algebraic theory of boolean circuits*, Journal of Pure and Applied Algebra 184, 2003.

Saunders MacLane, *Categories for the working mathematician*, Springer, 1998.

François Métayer, *Resolutions by polygraphs*, Theory and Applications of Categories 11(7).

Lutz Straßburger, *Linear logic and noncommutativity in the calculus of structures*, PhD thesis, Technischen Universität Dresden, 2003.