Efficient and complete resource consumption in Lolli

Iliano Cervesato

August 1, 1995

The design of linear logic programming languages opens a number of new implementation challenges not present in the case of more traditional logic languages such as *Prolog* or λ*Prolog*. Among these, the problem of efficiently managing the linear context when solving a multiplicative goal is of crucial importance for the use of these languages in large-scale applications. This document studies this problem in the case of *Lolli* [1]. We first give a proof-theoretic presentation of the operational semantics of *Lolli* as a *resolution system*. The non-deterministic distribution of the linear context featured by this system is overcome by a series of *resource management systems* (*I/O models* in [1]), with version numbers that range from 0.6 to 0.94. Systems 0.6 and 0.7 were devised by Hodas and Miller [1] and constitute the core of their *Prolog* prototype and released *ML* implementation of *Lolli* respectively.

Most of the resource management systems we review in this document have been implemented. The successive versions represent formal models for solving conceptual problems present in their predecessors, or simply to provide a view on better implementation techniques.

We did not prove most of the soundness and completeness theorems that relate these systems. In particular, no proof is included here. We believe these results are simple enough to be reconstructed by the reader.

Contents

List of Figures

$$\frac{D \gg a \setminus G \qquad \Gamma; \Delta \stackrel{r}{\Longrightarrow} G}{\Gamma; \Delta, D \stackrel{r}{\Longrightarrow} a}\ \mathbf{d/r} \qquad\qquad \frac{D \gg a \setminus G \qquad \Gamma, D; \Delta \stackrel{r}{\Longrightarrow} G}{\Gamma, D; \Delta \stackrel{r}{\Longrightarrow} a}\ \mathbf{!d/r}$$

$$\frac{}{\Gamma; \Delta \stackrel{r}{\Longrightarrow} a \doteq a}\ \mathbf{Ir}$$

$$\frac{}{\Gamma; \Delta \stackrel{r}{\Longrightarrow} \top}\ \top\mathbf{r} \qquad\qquad \frac{}{\Gamma; \cdot \stackrel{r}{\Longrightarrow} \mathbf{1}}\ \mathbf{1r}$$

$$\frac{\Gamma; \Delta \stackrel{r}{\Longrightarrow} G_1 \qquad \Gamma; \Delta \stackrel{r}{\Longrightarrow} G_2}{\Gamma; \Delta \stackrel{r}{\Longrightarrow} G_1 \,\&\, G_2}\ \&\,\mathbf{r} \qquad \frac{\Gamma; \Delta_1 \stackrel{r}{\Longrightarrow} G_1 \qquad \Gamma; \Delta_2 \stackrel{r}{\Longrightarrow} G_2}{\Gamma; \Delta_1, \Delta_2 \stackrel{r}{\Longrightarrow} G_1 \otimes G_2}\ \otimes\,\mathbf{r}$$

$$\left\{ \frac{\Gamma; \Delta \stackrel{r}{\Longrightarrow} G_1}{\Gamma; \Delta \stackrel{r}{\Longrightarrow} G_1 \oplus G_2}\ \oplus\,\mathbf{r_1} \qquad\qquad \frac{\Gamma; \Delta \stackrel{r}{\Longrightarrow} G_2}{\Gamma; \Delta \stackrel{r}{\Longrightarrow} G_1 \oplus G_2}\ \oplus\,\mathbf{r_2} \right\}$$

$$\frac{\Gamma; \Delta, D \stackrel{r}{\Longrightarrow} G}{\Gamma; \Delta \stackrel{r}{\Longrightarrow} D \multimap G}\ \multimap\mathbf{r} \qquad\qquad \frac{\Gamma, D; \Delta \stackrel{r}{\Longrightarrow} G}{\Gamma; \Delta \stackrel{r}{\Longrightarrow} D \rightarrow G}\ \rightarrow\mathbf{r}$$

$$\frac{\Gamma; \cdot \stackrel{r}{\Longrightarrow} G}{\Gamma; \cdot \stackrel{r}{\Longrightarrow} !G}\ \mathbf{!r}$$

$$\frac{\Gamma; \Delta \stackrel{r}{\Longrightarrow} [c/x]G}{\Gamma; \Delta \stackrel{r}{\Longrightarrow} \forall x.G}\ \forall\mathbf{r}^{*} \qquad\qquad \frac{\Gamma; \Delta \stackrel{r}{\Longrightarrow} [t/x]G}{\Gamma; \Delta \stackrel{r}{\Longrightarrow} \exists x.G}\ \exists\mathbf{r}$$

Figure 1: Resolution for linear hereditary Harrop formulas

$$\frac{}{a' \gg a \setminus a' \doteq a}\ \mathbf{Id}$$

$$\frac{}{\top \gg a \setminus \mathbf{0}}\ \top\mathbf{d} \qquad \frac{D_1 \gg a \setminus G_1 \qquad D_2 \gg a \setminus G_2}{D_1 \,\&\, D_2 \gg a \setminus G_1 \oplus G_2}\ \&\,\mathbf{d}$$

$$\frac{D \gg a \setminus G'}{G \multimap D \gg a \setminus G' \otimes G}\ \multimap\mathbf{d} \qquad \frac{D \gg a \setminus G'}{G \rightarrow D \gg a \setminus G' \otimes !G}\ \rightarrow\mathbf{d}$$

$$\frac{D \gg a \setminus G}{\forall x.D \gg a \setminus \exists x.G}\ \forall\mathbf{d}$$

Figure 2: Resolution for linear hereditary Harrop formulas (Continued)

1 Resolution for linear hereditary Harrop formulas

The *resolution system* [2] for *linear hereditary Harrop formulas* (*LHHF* for short), displayed in figures 1 and 2, expresses the operational semantics of *Lolli* at a level of abstraction such that:

- the generated proofs are *goal-directed* and *focused* (and therefore *uniform*);
- when a program formula D is selected in order to solve an atomic goal a, it is transformed into an appropriate but equivalent goal G before the computation is allowed to proceed (*resolution*);
- the following issues are not addressed within the proof system and therefore constitute *non-deterministic choices*:
  - the selection of the program formula to use when a goal is atomic, i.e. in rules **d/r** and **!d/r** (*disjunctive non-determinism*),
  - the sequentialization of goals in the binary rules, $\&$ **r** and $\otimes$ **r** in particular (*conjunctive non-determinism*),
  - the term t to use in rule $\exists$**r** when solving an existential goal (*existential non-determinism*), and
  - the way to split the linear context in rule $\otimes$**r** when solving a multiplicative conjunction (*resource distribution non-determinism*).

This system is different but equivalent to the *uniform provability system with backchaining* $\mathcal{L}'$ of Hodas and Miller, presented in [1]. Indeed, we have that the judgment $\Gamma; \Delta \overset{r}{\Longrightarrow} G$ is provable if and only if the sequent $\Gamma; \Delta \longrightarrow G$ is derivable in $\mathcal{L}'$. We will now point out the main differences between these two systems.

The rules in figure 1 correspond exactly to the right introduction rules in [1] with the only exception of **d/r**, **!d/r** and **I r**, which have to do with the backchaining rule these authors rely upon. When the goal formula a is atomic, a program formula D is selected from either the intuitionistic context (rule **d/r**) or from the linear context (rule **!d/r**). In both cases, it is passed to the *formula decomposition judgment* ($D \gg a \setminus G$) together with the atomic formula a in order to extract a goal formula G equivalent to D, in a sense to be explained shortly. At this point, the computation proceeds by solving G; the program formula D is removed from the context in the case it was a linear formula. Goals of the form $a \doteq a'$ (rule **I r**) are technical devices needed during the formula decomposition process. They can be viewed as identity tests for atomic formulas.

Program formulas can be seen as partial definitions of the atomic propositions (or the corresponding predicate symbol) that can appear in a goal position. Therefore, when selecting a program formula D from the context in order to solve an atomic goal a, we would like to transform D into a *clause* of the form $G \multimap a$, that can be immediately used to reduce the problem of proving a to that of solving G. This is exactly what is achieved by the formula decomposition judgment $D \gg a \setminus G$. In order to make this transformation, we need to extend the language of *LHHF* with two new constructs: the equality test $a \doteq a'$ among atomic formulas, and additive falsity $\mathbf{0}$ (this logical constant could be admitted without problems in the language of *LHHF* for goals).

The semantics of the formula decomposition judgment correspond to Hodas and Miller's specifications for the set $||\cdot||$ associated with the backchaining rule they use in [1]. Our description is more algorithmic and does not rely on infinite objects such as $||\cdot||$. We will now briefly describe the rules for this judgment, displayed in figure 2. Remember that only $\top$, $\&$, $\multimap$, $\rightarrow$ or $\forall$ can appear as the top connective of a program formula.

The presence of additive conjunction in the context permits to have a single program formula D partially defining several atomic propositions. If, among these, we are interested in the atomic formula a and we want to rewrite D as $G \multimap a$, we need to replace each defined atom a' with a *test* against a (rule $\mathbf{I\ d}$). We use a formula of the form $a \doteq a'$ for this purpose. As expected, this formula succeeds as a goal only if a and a' are the same atomic proposition (rule $\mathbf{I\ r}$).

In a program position, the logical constant $\top$ represents a total lack of information. Its operational behavior is not defined in such a context since it should not have one. The equivalent goal connective should represent failure. We use the logical constant $\mathbf{0}$ of linear logic, which does not have a right sequent rule, for this purpose (rule $\top\mathbf{d}$).

When using a program formula which main connective is $\&$, one of the conjuncts is kept while the other is thrown away; this corresponds to choosing the definition for one atomic goal formula out of multiple possibilities. A similar behavior in a goal position is obtained by using the additive disjunction of linear logic, $\oplus$ (rule $\&\mathbf{d}$).

A linear implication $G \multimap D$ in a program position is processed by solving its antecedent G as a subgoal and then continuing with whatever computation is specified by its consequent. This is clearly a conjunctive behavior. Since the context is split between the branch of the proof-tree that solves G, and the remainder of the computation, that decomposes D, this behavior is caught by multiplicative conjunction, $\otimes$, as the emulation of $\multimap$ in a goal position (rule $\multimap\mathbf{d}$).

The case of intuitionistic implication is handled in a similar way. Indeed, remember that $A \rightarrow B$ is defined as $!A \multimap B$. Therefore, multiplicative conjunction is used also in this case (rule $\rightarrow\mathbf{d}$).

Whenever a program formula D is enclosed by a universal quantification, we need to instantiate the quantified variables with an appropriate term in order to proceed with the computation (this is achieved concretely by unifying the current goal with the formula defined in D). We can emulate this behavior by using an existential quantifier in the corresponding goal position (rule $\forall\mathbf{d}$).

It is interesting to notice that the resulting goal connectives are the De Morgan duals of the program operators they origin from, with the exception of the test operator. Duality is obtained with respect to linear negation $(\cdot^{\perp})$. This result is indeed sensible: from a sequent calculus point of view, a program formula is an object that lives in the left-hand side of the sequent arrow while a goal resides on the right-hand side. Therefore, if two such entities are to be equivalent, in some sense, they must be dual of each other. We have indeed the following correspondence:

$$\begin{array}{rcl}
\top^{\perp} & = & \mathbf{0} \\
(D_1 \& D_2)^{\perp} & = & {D_1}^{\perp} \oplus {D_2}^{\perp} \\
(G \multimap D)^{\perp} & = & G \otimes D^{\perp} \\
(G \rightarrow D)^{\perp} & = & !G \otimes D^{\perp} \\
(\forall x.D)^{\perp} & = & \exists x.D^{\perp}
\end{array}$$

$$\frac{D \gg a \setminus G \qquad \Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G}{\Gamma; \Delta^I, D \setminus \Delta^O \Longrightarrow a}\ \mathbf{d/io.6} \qquad \frac{D \gg a \setminus G \qquad \Gamma, D; \Delta^I \setminus \Delta^O \Longrightarrow G}{\Gamma, D; \Delta^I \setminus \Delta^O \Longrightarrow a}\ \mathbf{!d/io.6}$$

$$\frac{}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow a \doteq a}\ \mathbf{I\ io.6}$$

$$\frac{}{\Gamma; \Delta, \Delta^O \setminus \Delta^O \Longrightarrow \top}\ \top\ \mathbf{io.6} \qquad \frac{}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow \mathbf{1}}\ \mathbf{1\ io.6}$$

$$\frac{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G_1 \qquad \Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G_2}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G_1 \,\&\, G_2}\ \&\ \mathbf{io.6} \qquad \frac{\Gamma; \Delta^I \setminus \Delta' \Longrightarrow G_1 \qquad \Gamma; \Delta' \setminus \Delta^O \Longrightarrow G_2}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G_1 \otimes G_2}\ \otimes\ \mathbf{io.6}$$

$$\left\{ \frac{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G_1}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G_1 \oplus G_2}\ \oplus_1 \mathbf{io.6} \qquad \frac{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G_2}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G_1 \oplus G_2}\ \oplus_2 \mathbf{io.6} \right\}$$

$$\frac{\Gamma; \Delta^O, \Delta, D \setminus \Delta^O \Longrightarrow G}{\Gamma; \Delta^O, \Delta \setminus \Delta^O \Longrightarrow D \multimap G}\ \multimap \mathbf{io.6} \qquad \frac{\Gamma, D; \Delta^I \setminus \Delta^O \Longrightarrow G}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow D \rightarrow G}\ \rightarrow \mathbf{io.6}$$

$$\frac{\Gamma; \cdot \setminus \cdot \Longrightarrow G}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow !G}\ !\ \mathbf{io.6}$$

$$\frac{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow [c/x]G}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow \forall x.G}\ \forall\ \mathbf{io.6}^* \qquad \frac{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow [t/x]G}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow \exists x.G}\ \exists\ \mathbf{io.6}$$

Figure 3: Resource consumption for *LHHF*, version 0.6

2 Resource consumption for LHHF, version 0.6

As stated earlier, the resolution calculus presented in the previous section does not commit to any strategy in order to split the linear context when processing multiplicative goals. As we will see shortly, the kind of non-determinism involved in this open choice can be computationally harmful unless we devise a sound and complete method to split the linear context. Let us state the problem. The resolution rule for the connective $\otimes$ is as follows:

$$\frac{\Gamma; \Delta_1 \overset{r}{\Longrightarrow} G_1 \qquad \Gamma; \Delta_2 \overset{r}{\Longrightarrow} G_2}{\Gamma; \underbrace{\Delta_1, \Delta_2}_{\Delta} \overset{r}{\Longrightarrow} G_1 \otimes G_2} \otimes \mathbf{r}$$

In order to solve the goal $G_1 \otimes G_2$, we need to split the original linear context Δ into Δ_1 and Δ_2, solve G_1 in Δ_1 and G_2 in Δ_2. Assume that Δ contains n formulas. Then there are 2^n possible splits. Guessing the right one will require to try them all, in the worst case. This is clearly a major efficiency problem: a language based on a generate-and-test implementation of this kind could be used for toy problems, but would never scale up to real applications. This is particularly problematic since multiplicative conjunction is frequently needed in practice (it is the most common operator in *Lolli* programs).

This problem was given a deterministic solution by Hodas and Miller in [1] in what they called the *I/O model* of execution for *Lolli*. We will instead use the name *resource consumption system* and label this deduction system as *version 0.6* since we will come up with several other versions of this calculus.

The rule above, $\otimes$ **r**, attempts to split the context Δ at a stage when the resources needed to prove the two subgoals G_1 and G_2 are completely unknown. There is no way of splitting Δ correctly until we have solved G_1 and G_2. However, if the original goal is to succeed, all resources not used to prove G_1 will be used to solve G_2, and vice versa. The key idea behind the resource consumption model is, therefore, to upgrade the passive role of goal formulas as *resource users* to the more active role of *resource consumers*. Under this view, we will give one of the subgoals, G_1 say, the whole linear context Δ, it will consume part of it and return the remaining context Δ_2 that will be used by G_2.

This basic idea is formalized by means of judgments of the form $\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G$, where Δ^I is the linear part of the context that is given as *input* in order to solve G. In general, G will be just one of the subgoals produced during the derivation a top-level goal A. The proof of G will consume part of Δ^I and return the portion it did not use as the *output context* Δ^O, that will be consumed by some other subgoal derived from A. Clearly the output context for the original goal A should be empty. Indeed, the soundness and completeness theorem for resource consumption states that $\Gamma; \Delta \overset{r}{\Longrightarrow} G$ is derivable if and only if $\Gamma; \Delta \setminus \cdot \Longrightarrow G$ is derivable, where $\cdot$ represents the empty context. An important invariant of this proof system, as well as the variants that will be displayed in the subsequent, is that, when the judgment $\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G$ is derivable, the output context Δ^O is always a submultiset of the input context Δ^I.

In their original paper [1], Hodas and Miller write this judgment $I\{G\}O$, with G, I and O being the goal formula, the input and the output context respectively. The main difference with respect to our judgment is that I and O are lists of items that can be either program formulas or the special constant `del`. This is very close to the original *ML* implementation of *Lolli*. Our Δ^I and Δ^O are instead multisets of formulas. This is consistent with the resolution judgment presented in §1, and permits easier proofs of soundness and completeness.

We will now go through the rules in figure 3 and explain how the context management scheme outlined above is concretely realized, and what the impact it has on the non-multiplicative rules. First consider the rule that deals with multiplicative conjunction, reported below:

$$\frac{\Gamma; \Delta^I \setminus \Delta' \Longrightarrow G_1 \qquad \Gamma; \Delta' \setminus \Delta^O \Longrightarrow G_2}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G_1 \otimes G_2} \otimes \textbf{io.6}$$

The input context in the conclusion of the rule, Δ^I, is passed entirely to the left premiss in order to solve G_1. This computation will consume part of Δ^I and return the rest as the output context Δ', that will be piped over to the right premiss in order to solve G_2. This goal will in turn return an output context Δ^O that corresponds to the output context of the overall goal $G_1 \otimes G_2$. There are two remarks to make at this point: first, when considering the judgment $\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow G$, we are adopting a computational point of view according to which the schematic variables Γ, Δ^I and G are given as input to the rules while Δ^O is returned as an output value from the resolution of the goal. Second this vision imposes to solve first the left subgoal and then the right branch in rule $\otimes$ **io.6**.

Most of the remaining rules, $\forall$ **io.6** for instance, do not have a local effect on the linear context. Therefore, in order to solve the goal formula in their conclusion, $\forall x.G$ say, they simply take whatever input context Δ^I they are provided with and forward it as such to their premiss to solve the subgoal, $[c/x]G$ in this case. This computation will produce an output context Δ^O that is returned as the result of the overall computation performed by the rule. Other rules with this property are **!d/io.6**, $\oplus_i$ **io.6**, $\rightarrow$ **io.6** and $\exists$ **io.6**.

When selecting a program formula D from the linear input context (rule **d/io.6**) in order to solve an atomic goal a, we need to retract D from the input context Δ^I that we pass over in order to solve the goal G obtained by decomposing D with respect to a. Indeed, since D has been consumed during this process, it must not be available to the resolution of G, nor of any successive goal. Notice that Δ^O cannot contain D (or better the instance of D used by the rule) since it has been cancelled from Δ^I. Therefore successive subgoals, which will use Δ^O cannot access it. No such devices are needed in case the selected formula is intuitionistic (rule **!d/io.6**).

The resolution rules for the equality test (**I io.6**) and for the multiplicative unit (**1 io.6**) require an empty linear context. Therefore, solving these goals does not consume resources. We model this behavior by returning as output the same context these rules received as input. The case of the connective $\top$ (rule $\top$ **io.6**) is different: it is allowed to consume a portion Δ of its input context Δ^I. If we write the latter as Δ, Δ^O, the output context returned by this rule is Δ^O.

The operational behavior of additive conjunction $\&$ requires to solve both subgoals G_1 and G_2 in the same linear context. This is modelled in our resource consumption system by giving the original input context to both G_1 and G_2, and expecting them to return the same output context, Δ^O, that will be the output context of the original formula $G_1 \& G_2$. This behavior is caught in rule $\&$ **io.6**.

The rule for multiplicative implication, $\multimap$ **io.6**, requires some attention. Let Δ^I be the original input context. In order to process this connective, we need to augment Δ^I with the antecedent D of the implication. Let Δ^O be the context returned after solving its consequent G. We could pass Δ^O over as the result of the resolution of $D \multimap G$ if we were sure that D does not appear in Δ^O. Indeed, D must have been consumed during the resolution of G. We enforce this constraint by writing Δ^I as Δ, Δ^O and passing Δ, D, Δ^O as the input context to the premiss of the rule, where Δ, D represents the portion of the input context that will be consumed while solving G. The rule dealing with intuitionistic implication $\rightarrow$ **io.6** is not concerned with this problems.

Finally, rule ! **io.6** describes how to solve a banged goal. Since this connective requires it subgoal to hold in the empty linear context, we return the whole input context as output. Moreover, we prove this subgoal with the empty linear context, and, of course, expect the empty linear context as output.

$$\frac{D \gg a \setminus G \qquad \Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_v G}{\boxed{\Gamma; \Delta^I, D}^1 \setminus \Delta^O \Longrightarrow_v a}\ \mathbf{d/io.7} \qquad \frac{D \gg a \setminus G \qquad \Gamma, D; \Delta^I \setminus \Delta^O \Longrightarrow_v G}{\boxed{\Gamma, D; \Delta^I}^1 \setminus \Delta^O \Longrightarrow_v a}\ \mathbf{!d/io.7}$$

$$\frac{}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow_0 a \doteq a}\ \mathbf{I\ io.7}$$

$$\frac{}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow_1 \top}\ \top\ \mathbf{io.7} \qquad \frac{}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow_0 \mathbf{1}}\ \mathbf{1\ io.7}$$

$$\left\{ \begin{array}{c}
\dfrac{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_0 G_1 \qquad \Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_0 G_2}{\Gamma; \Delta^I \setminus \boxed{\Delta^O}^2 \Longrightarrow_0 G_1 \,\&\, G_2}\ \&_1\mathbf{io.7} \\[2ex]
\dfrac{\Gamma; \Delta^I \setminus \Delta_1, \Delta^O \Longrightarrow_1 G_1 \qquad \Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_0 G_2}{\Gamma; \Delta^I \setminus \boxed{\Delta^O}^3 \Longrightarrow_0 G_1 \,\&\, G_2}\ \&_2\mathbf{io.7} \\[2ex]
\dfrac{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_0 G_1 \qquad \Gamma; \Delta^I \setminus \Delta_2, \Delta^O \Longrightarrow_1 G_2}{\Gamma; \Delta^I \setminus \boxed{\Delta^O}^3 \Longrightarrow_0 G_1 \,\&\, G_2}\ \&_3\mathbf{io.7} \\[2ex]
\dfrac{\Gamma; \Delta^I \setminus \Delta_1^O \Longrightarrow_1 G_1 \qquad \Gamma; \Delta^I \setminus \Delta_2^O \Longrightarrow_1 G_2}{\Gamma; \Delta^I \setminus \boxed{\Delta_1^O \cap \Delta_2^O}^4 \Longrightarrow_1 G_1 \,\&\, G_2}\ \&_4\mathbf{io.7}
\end{array} \right\}$$

Figure 4: Resource consumption for *LHHF*, version 0.7

[1] `pick_resource`
[2] `equalprog`
[3] `subcontext`
[4] `context_intersection`
[5] `d_scan`
[6] `d_thinnable`

$$\dfrac{\Gamma;\Delta^I\setminus\Delta' \Longrightarrow_{v_1} G_1 \qquad \Gamma;\Delta'\setminus\Delta^O \Longrightarrow_{v_2} G_2}{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_{v_1\vee v_2} G_1 \otimes G_2}\ \otimes\,\mathbf{io.7}$$

$$\left\{\begin{array}{c}\dfrac{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_v G_1}{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_v G_1 \oplus G_2}\ \oplus_1\mathbf{io.7} \\[2em] \dfrac{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_v G_2}{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_v G_1 \oplus G_2}\ \oplus_2\mathbf{io.7}\end{array}\right\} \qquad \left\{\begin{array}{c}\dfrac{\boxed{\Gamma;\Delta^O,\Delta,D}^5\setminus\Delta^O \Longrightarrow_0 G}{\Gamma;\Delta^O,\Delta\setminus\boxed{\Delta^O}^6 \Longrightarrow_0 D \multimap G}\ \multimap_1\mathbf{io.7} \\[2em] \dfrac{\boxed{\Gamma;\Delta^I,D}^5\setminus\Delta^O \Longrightarrow_1 G}{\Gamma;\Delta^I\setminus\Delta^I\cap\Delta^O \Longrightarrow_1 D \multimap G}\ \multimap_2\mathbf{io.7}\end{array}\right\}$$

$$\dfrac{\boxed{\Gamma,D;\Delta^I}^5\setminus\Delta^O \Longrightarrow_v G}{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_v D \rightarrow G}\ \rightarrow\mathbf{io.7} \qquad \dfrac{\Gamma;\Delta^I\setminus\Delta^I \Longrightarrow_v G}{\Gamma;\Delta^I\setminus\boxed{\Delta^I}^2 \Longrightarrow_0 !G}\ !\,\mathbf{io.7}$$

$$\dfrac{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_v [c/x]G}{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_v \forall x.G}\ \forall\,\mathbf{io.7}^* \qquad \dfrac{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_v [t/x]G}{\Gamma;\Delta^I\setminus\Delta^O \Longrightarrow_v \exists x.G}\ \exists\,\mathbf{io.7}$$

Figure 5: Resource consumption for *LHHF*, version 0.7 (Continued)

3 Resource consumption for LHHF, version 0.7

The context management policy enforced by the system presented in section 2 is not fully deterministic yet. This is due to the operational semantics of the logical constant $\top$. As presented in rule $\top$ **io.6**, this goal is allowed to consume any portion Δ of its input context Δ^I. If Δ^I contains n formulas, we are still left with 2^n possible output contexts Δ^O to be passed to the remaining computation. Hodas and Miller [1] have provided a variant of their I/O model that takes care of this problem. We report their solution using our notation in figures 4 and 5. We use the name *version 0.7* for the resulting rule set. The original *ML* implementation by Josh Hodas was based on this system. We provide hooks to the code by framing parts of the context which management requires calling specialized functions. The names of these functions are given as footnotes.

Roughly, the idea is that once a $\top$ has been encountered, the remaining subgoals do not need to consume all of the input context since the unused formulas could be "pumped back" to the place in the proof tree where $\top$ was first seen. We obtain this behavior by adding an extra parameter to the resource consumption judgment of version 0.6: we now rely on judgments of the form $\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_v G$ where v is a boolean flag (the $\top$-*flag* or *slack indicator*) to be considered as another output argument of the resolution of the goal G. Whenever $v = 0$, the resolution of G uses exactly the resources in $\Delta^I - \Delta^O$. If instead this flag has value 1, G uses $\Delta^I - \Delta^O$ for sure, but may absorb also part or all of the output context Δ^O. In this case, we say that Δ^O is the *slack* of that branch of the proof tree. When $v = 0$, the computation has no slack. The resulting deduction system, labelled version 0.7, is presented in figures 4 and 5.

The main changes with respect to the previous system concern the rules that close the proof trees and the binary rules. Rules **I io.7** and **1 io.7** pass their linear context as the output context for the remainder of the computation. These rules set the $\top$-flag to 0 since no occurrence of $\top$ can have been encountered during the proof of either **1** or the identity test. When this connective is processed as a goal, it passes its input context as output, but raises the $\top$-flag. The subsequent computation will use this information for context management.

Rule $\&$ **io.6** is split into four rules ($\&_i$ **io.7** with i ranging from 1 to 4) according to the status of the $\top$-flags returned by its two premisses. If no $\top$ was encountered while solving either G_1 nor G_2 (rule $\&_1$ **io.7**), then the context is managed as in the previous system and the $\top$-flag for $G_1 \& G_2$ is set to 0. If exactly one of the two premisses sets the $\top$-flag (rules $\&_2$ **io.7** and $\&_3$ **io.7**), then the behavior of the rule is determined by the other premiss. Consider the first of these rule, for instance, and let Δ be the portion of the context used while proving G_2 (clearly, $\Delta^I = \Delta, \Delta^O$). Since both G_1 and G_2 must consume the same portion of the context, the proof of G_1 can use part of Δ but no formula from Δ^O. Since a $\top$ has been encountered, it could indeed use a proper submultiset of Δ. We enforce this constraint by writing the output context of G_1 as Δ_1, Δ^O, where Δ_1 is the actual slack of this branch of the proof tree. The $\top$-flag for $G_1 \& G_2$ is set to zero: since both premisses must consume the same resources and G_2 has no slack, the composed goal cannot have any slack. For a similar reason, the output context of $G_1 \& G_2$ is Δ^O. In the case both premisses return their $\top$-flag set (rule $\&_4$ **io.7**), both subgoal allow arbitrary slack. Therefore, we set the $\top$-flag for the composed formula since in this case exuberant resources can be "pumped back" to both premisses. The resulting output context is the intersection of the linear contexts returned by the premisses: since both branches are constrained to consume the same resources, only what is not used in either can be forwarded.

Slack handling in the rule for multiplicative conjunction (rule $\otimes$ **io.7**) is quite simple since resources are allowed to flow freely from one premiss to the other. We set the $\top$-flag if either subgoal allows slack. Moreover, the overall output context is simply the linear context returned after proving the right premiss.

The two rules for linear implication ($\multimap_1$ **io.7** and $\multimap_2$ **io.7**) are a specialization of rule $\multimap$ **io.6**. In case the $\top$-flag is not set, we essentially proceed as in system 0.6. Otherwise, the possibility of having slack saves us from the duty of checking that the formula D has been consumed in rule $\multimap_2$ **io.7** while proving the goal G: if it has not, it will be part of the slack, but it must not be accessible to the rest of the computation. This is the reason why we take the intersection between the original input context and the output context returned by G. In the concrete implementation, this is handled by pattern matching, so no function call is needed in this case.

Finally, the rule for !, ! **io.7**, resets the $\top$-flag independently of whether $\top$ has been encountered while solving the subgoal G or not. Since the output context coincides with the input context in this rule, there is no place for any slack. This rule reflects the actual implementation: instead of calling G with an empty linear context, the input context is forwarded to this goal, and the returned context is checked for equality against it. This operational behavior fits better the Hodas's implementation strategy than using the empty input context, as illustrated in rule ! **io.6**.

$$\frac{D \gg a \setminus G \qquad \Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_v G}{\boxed{\Gamma; \Delta^I, D}^{1} \setminus \Delta^O \Longrightarrow_v a}\ \mathbf{d/io.8} \qquad \frac{D \gg a \setminus G \qquad \Gamma; \Delta \Longrightarrow' G}{\boxed{\Gamma; \Delta, D}^{1} \Longrightarrow' a}\ \mathbf{d/io.8'}$$

$$\frac{D \gg a \setminus G \qquad \Gamma, D; \Delta^I \setminus \Delta^O \Longrightarrow_v G}{\boxed{\Gamma, D; \Delta^I}^{1} \setminus \Delta^O \Longrightarrow_v a}\ \mathbf{!d/io.8} \qquad \frac{D \gg a \setminus G \qquad \Gamma, D; \Delta \Longrightarrow' G}{\boxed{\Gamma, D; \Delta}^{1} \Longrightarrow' a}\ \mathbf{!d/io.8'}$$

$$\frac{}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow_0 a \doteq a}\ \mathbf{I\ io.8} \qquad \frac{}{\boxed{\Gamma; \cdot}^{2} \Longrightarrow' a \doteq a}\ \mathbf{I\ io.8'}$$

$$\frac{}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow_0 \mathbf{1}}\ \mathbf{1\ io.8} \qquad \frac{}{\boxed{\Gamma; \cdot}^{2} \Longrightarrow' \mathbf{1}}\ \mathbf{1\ io.8'}$$

$$\frac{}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow_1 \top}\ \top\ \mathbf{io.8} \qquad \frac{}{\Gamma; \Delta \Longrightarrow' \top}\ \top\ \mathbf{io.8'}$$

Figure 6: Resource consumption for *LHHF*, version 0.8

[1] `pick_resource`
[2] `consumed`
[3] `complement`
[4] `subset`
[5] `intersect`
[6] `d_scan`
[7] `erase`

$$\left\{\begin{array}{c}
\dfrac{\Gamma;\Delta,\Delta^{O}\backslash\,\Delta^{O}\Longrightarrow_{0} G_1 \qquad \boxed{\Gamma;\Delta}^{3} \Longrightarrow' G_2}{\Gamma;\Delta,\Delta^{O}\backslash\,\Delta^{O}\Longrightarrow_{0} G_1 \,\&\, G_2}\ \&_{1}\mathbf{io.8} \\[2ex]
\dfrac{\Gamma;\Delta^{I}\backslash\,\Delta_1,\Delta^{O}\Longrightarrow_{1} G_1 \qquad \Gamma;\Delta^{I}\backslash\,\Delta^{O}\Longrightarrow_{0} G_2}{\Gamma;\Delta^{I}\backslash\boxed{\Delta^{O}}^{4} \Longrightarrow_{0} G_1 \,\&\, G_2}\ \&_{2}\mathbf{io.8} \\[2ex]
\dfrac{\Gamma;\Delta^{I}\backslash\,\Delta_1^{O}\Longrightarrow_{1} G_1 \qquad \Gamma;\Delta^{I}\backslash\,\Delta_2^{O}\Longrightarrow_{1} G_2}{\Gamma;\Delta^{I}\backslash\boxed{\Delta_1^{O}\cap\Delta_2^{O}}^{5} \Longrightarrow_{1} G_1 \,\&\, G_2}\ \&_{3}\mathbf{io.8}
\end{array}\right\}$$

$$\searrow$$

$$\dfrac{\Gamma;\Delta \Longrightarrow' G_1 \qquad \Gamma;\Delta \Longrightarrow' G_2}{\Gamma;\Delta \Longrightarrow' G_1 \,\&\, G_2}\ \&\,\mathbf{io.8'}$$

$$\dfrac{\Gamma;\Delta^{I}\backslash\,\Delta'\Longrightarrow_{v_1} G_1 \qquad \Gamma;\Delta'\backslash\,\Delta^{O}\Longrightarrow_{v_2} G_2}{\Gamma;\Delta^{I}\backslash\,\Delta^{O}\Longrightarrow_{v_1\vee v_2} G_1 \otimes G_2}\ \otimes\,\mathbf{io.8}$$

$$\left\{\begin{array}{c}
\dfrac{\Gamma;\Delta\backslash\,\Delta'\Longrightarrow_{0} G_1 \qquad \Gamma;\Delta' \Longrightarrow' G_2}{\Gamma;\Delta \Longrightarrow' G_1 \otimes G_2}\ \otimes_{1}\mathbf{io.8'} \\[2ex]
\dfrac{\Gamma;\Delta\backslash\,\Delta'\Longrightarrow_{1} G_1 \qquad \Gamma;\Delta'\backslash\,\Delta''\Longrightarrow_{v} G_2}{\Gamma;\Delta \Longrightarrow' G_1 \otimes G_2}\ \otimes_{2}\mathbf{io.8'}
\end{array}\right\}$$

$$\left\{\begin{array}{c}
\dfrac{\Gamma;\Delta^{I}\backslash\,\Delta^{O}\Longrightarrow_{v} G_1}{\Gamma;\Delta^{I}\backslash\,\Delta^{O}\Longrightarrow_{v} G_1 \oplus G_2}\ \oplus_{1}\mathbf{io.8} \\[2ex]
\dfrac{\Gamma;\Delta^{I}\backslash\,\Delta^{O}\Longrightarrow_{v} G_2}{\Gamma;\Delta^{I}\backslash\,\Delta^{O}\Longrightarrow_{v} G_1 \oplus G_2}\ \oplus_{2}\mathbf{io}
\end{array}\right\}$$

$$\left\{\begin{array}{c}
\dfrac{\Gamma;\Delta \Longrightarrow' G_1}{\Gamma;\Delta \Longrightarrow' G_1 \oplus G_2}\ \oplus_{1}\mathbf{io.8'} \\[2ex]
\dfrac{\Gamma;\Delta \Longrightarrow' G_2}{\Gamma;\Delta \Longrightarrow' G_1 \oplus G_2}\ \oplus_{2}\mathbf{io.8'}
\end{array}\right\}$$

Figure 7: Resource consumption for *LHHF*, version 0.8 (Continued)

$$\left\{\begin{array}{c}
\dfrac{\boxed{\Gamma; \Delta, \Delta^O, D}^{6} \setminus \Delta^O \Longrightarrow_0 G}{\Gamma; \Delta, \Delta^O \setminus \Delta^O \Longrightarrow_0 D \multimap G}\ {\multimap}_1\mathbf{io.8} \\[2em]
\dfrac{\boxed{\Gamma; \Delta^I, D}^{6} \setminus \Delta^O \Longrightarrow_1 G}{\Gamma; \Delta^I \setminus \boxed{\Delta^I \cap \Delta^O}^{2} \Longrightarrow_1 D \multimap G}\ {\multimap}_2\mathbf{io.8}
\end{array}\right\}
\qquad
\dfrac{\boxed{\Gamma; \Delta, D}^{6} \Longrightarrow' G}{\Gamma; \Delta \Longrightarrow' D \multimap G}\ {\multimap}\mathbf{io.8'}$$

$$\dfrac{\boxed{\Gamma, D; \Delta^I}^{6} \setminus \Delta^O \Longrightarrow_v G}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_v D \to G}\ {\to}\,\mathbf{io.8}
\qquad
\dfrac{\boxed{\Gamma, D; \Delta}^{6} \Longrightarrow' G}{\Gamma; \Delta \Longrightarrow' D \to G}\ {\to}\,\mathbf{io.8'}$$

$$\dfrac{\boxed{\Gamma; \cdot}^{7} \Longrightarrow' G}{\Gamma; \Delta^I \setminus \Delta^I \Longrightarrow_0 !G}\ !\,\mathbf{io.8}
\qquad
\dfrac{\Gamma; \cdot \Longrightarrow' G}{\boxed{\Gamma; \cdot}^{2} \Longrightarrow' !G}\ !\,\mathbf{io.8'}$$

$$\dfrac{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_v [c/x]G}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_v \forall x.G}\ \forall\,\mathbf{io.8}^{*}
\qquad
\dfrac{\Gamma; \Delta \Longrightarrow' [c/x]G}{\Gamma; \Delta \Longrightarrow' \forall x.G}\ \forall\,\mathbf{io.8}'^{*}$$

$$\dfrac{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_v [t/x]G}{\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_v \exists x.G}\ \exists\,\mathbf{io.8}
\qquad
\dfrac{\Gamma; \Delta \Longrightarrow' [t/x]G}{\Gamma; \Delta \Longrightarrow' \exists x.G}\ \exists\,\mathbf{io.8'}$$

Figure 8: Resource consumption for *LHHF*, version 0.8 (Continued)

4 Resource consumption for LHHF, version 0.8

Version 0.7 of the resource consumption model, presented in §3, achieves determinism in context management. A close examination of the rules reveals that some efficiency and completeness problems still remain. In particular, the rules concerning $\&$ are unsatisfactory. The problems already appears in version 0.6. We had the following rule:

$$\frac{\Gamma; \Delta^I \backslash \Delta^O \Longrightarrow G_1 \qquad \Gamma; \Delta^I \backslash \Delta^O \Longrightarrow G_2}{\Gamma; \Delta^I \backslash \Delta^O \Longrightarrow G_1 \,\&\, G_2} \,\&\,\mathbf{io.6}$$

Assuming a sequential execution order for the two premisses, this rules requires to solve first G_1, say, obtaining an output context Δ_1^O. Then G_2 will be proved and return the output context Δ_2^O. Operationally, at this point, and only at this point we check that Δ_1^O and Δ_2^O are equal. In principle this test can be done efficiently (for example by having a bit vector where each position records whether the corresponding resource has been used). Assume however a situation where the proof of G_2 attempts to use a formula D present in Δ_1^O (and there are no $\top$ around to cushion it). The consequent failure could be discovered as soon as D is accessed. Our current model postpones recognizing this situation until G_2 has been completely solved. In a computation model based on backtracking, early identifications of illegal uses of resources of the kind described above are highly beneficial: after D has been accessed, the resolution of G_2 may involve heavy computations that might be completely avoided (efficiency issue); moreover, this computation might be diverging (completeness issue).

Consider the following example, written using *Lolli*'s concrete syntax:

```
test :- (a & b), c.

a.

b :- c, write_sans "I may diverge".  % Fails
b.                                   % Succeeds

LINEAR c.
```

According to the current execution model, the goal '`?- test`' is solved by first proving `a` without consuming any resource, then attempting to prove `b`. The first clause for this goal is attempted first and has `b` succeed (the message is printed). Upon returning, the context used for solving `a` and `b` are compared and a failure is reported since the latter uses `c` while the former does not. This triggers backtracking and the second clause for `b` is accessed, making the overall computation to succeed eventually. As we said above, we would like the first attempt to solve `b` to fail as soon as `c` is accessed (so that the error message does not get printed).

In order to recognize early failures due to the fact that the second goal incorrectly accesses resources unused by the first, we modify the rule $\&$ **io.6** as follows:

$$\frac{\Gamma; \Delta^I \backslash \Delta^O \Longrightarrow G_1 \qquad \Gamma; \Delta^I - \Delta^O \backslash \cdot \Longrightarrow G_2}{\Gamma; \Delta^I \backslash \Delta^O \Longrightarrow G_1 \,\&\, G_2} \,\&\,\mathbf{io.6'}$$

In this rule, we give G_2 exactly the portion of the linear context that it must use and expect the empty context as an output. In this way, the resources not consumed by G_1 are inaccessible to G_2; this achieves our purposes.

Version 0.8 of the resource consumption system develops this idea and integrates the technique presented in version 0.7 to handle $\top$. We rely on two mutually recursive judgments: $\Gamma; \Delta^I \setminus \Delta^O \Longrightarrow_v G$ and $\Gamma; \Delta \Longrightarrow' G$, called the *partial* and the *total consumption judgment* respectively. The first shares the semantics as the similar judgment of version 0.7: the goal G is provable from the context $\Gamma; \Delta^I$ returning the linear context Δ^O as output, where v is the slack indicator. Total consumption behaves like the resolution judgment described in §1: G is provable in the context $\Gamma; \Delta$, and in particular consumes completely the linear part of the context, Δ.

Most of the rules for this system, presented in figures 6, 7 and 8, correspond exactly to the equivalent rules of either system 0.7 or of the resolution calculus. The two judgments communicate in the rules that deal with $\&$, $\otimes$ and !. More precisely, only rules $\&_1$ **io.8**, $\otimes_i$ **io.8**$'$ and ! **io.8** introduce new features.

The first of these rules expresses faithfully the idea described above as $\&$ **io.6**$'$ as soon as we write $\Delta^I - \Delta^O$ as Δ. If the resolution of G_1 does not allow any slack, then Δ must be entirely consumed while solving G_2, independently of whether $\top$ is encountered or not. We obtain this by using total the consumption judgment. In the case solving G_1 permits slack, we must behave similarly to version 0.7.

Since the argument of a ! must be solved in an empty context (and therefore must trivially consume it all), the partial consumption rule dealing with this connective (! **io.8**) can take advantage of the total consumption judgment.

Rules $\otimes_i$ **io.8**$'$ describe how to solve a goal of the form $G_1 \otimes G_2$ so that the linear context gets completely consumed. We give the whole context to G_1 by means of the partial consumption judgment. This goal will consume part of it and possibly raise the $\top$-flag. Depending on the returned value for this flag, two courses of action can be taken. If the $\top$-flag is not set, then G_2 must consume all the resources discarded by G_1; total consumption is used for this purpose. Otherwise, if the slack indicator is set, we pass the output context of G_1 to G_2 as input by means of the partial consumption judgment, and discard the resources that this goal does not consume (we ignore also the $\top$-flag returned by this branch of the proof tree).

$$\left\{ \quad \frac{D \gg a \setminus G \qquad \Gamma;\Xi;\Delta^I \setminus \Delta^O \Longrightarrow_v G}{\boxed{\Gamma;\Xi;(\Delta^I, D)}^{1} \setminus \Delta^O \Longrightarrow_v a}\,\mathbf{d_1/io.9} \qquad \frac{D \gg a \setminus G \qquad \Gamma;\Xi;\Delta^I \setminus \Delta^O \Longrightarrow_v G}{\boxed{\Gamma;(\Xi, D);\Delta^I}^{1} \setminus \Delta^O \Longrightarrow_v a}\,\mathbf{d_2/io.9} \quad \right\}$$

$$\frac{D \gg a \setminus G \qquad (\Gamma, D);\Xi;\Delta^I \setminus \Delta^O \Longrightarrow_v G}{\boxed{(\Gamma, D);\Xi;\Delta^I}^{1} \setminus \Delta^O \Longrightarrow_v a}\,\mathbf{!d/io.9}$$

$$\frac{}{\boxed{\Gamma;\cdot;\Delta^I}^{2} \setminus \Delta^I \Longrightarrow_0 a \doteq a}\,\mathbf{I\ io.9} \qquad \frac{}{\boxed{\Gamma;\cdot;\Delta^I}^{2} \setminus \Delta^I \Longrightarrow_0 \mathbf{1}}\,\mathbf{1\ io.9}$$

$$\frac{}{\boxed{\Gamma;\Xi;\Delta^I}^{3} \setminus \Delta^I \Longrightarrow_1 \top}\,\top\,\mathbf{io.9}$$

$$\left\{ \begin{array}{c} \dfrac{\Gamma;\Xi;\Delta^I \setminus \Delta^O \Longrightarrow_0 G_1 \qquad \boxed{\Gamma;(\Xi, \Delta^I - \Delta^O);\cdot}^{4} \setminus \Delta^\perp \Longrightarrow_v G_2}{\Gamma;\Xi;\Delta^I \setminus \Delta^O \Longrightarrow_0 G_1 \,\&\, G_2}\,\&_1\mathbf{io.9} \\[3ex] \dfrac{\Gamma;\Xi;\Delta^I \setminus \Delta' \Longrightarrow_1 G_1 \qquad \boxed{\Gamma;(\Xi, \Delta^I - \Delta');\Delta'}^{5} \setminus \Delta^O \Longrightarrow_v G_2}{\Gamma;\Xi;\Delta^I \setminus \Delta^O \Longrightarrow_v G_1 \,\&\, G_2}\,\&_2\mathbf{io.9} \end{array} \right\}$$

Figure 9: Resource consumption for *LHHF*, version 0.9

[1] `pick_resource`

[2] `strict`

[3] `erase_strict`

[4] `strict_difference_and_erase`

[5] `strict_difference_and_keep`

$$\left\{\begin{array}{c}
\dfrac{\boxed{\Gamma; \cdot; (\Xi, \Delta^I)}^{6} \setminus \Delta' \Longrightarrow_0 G_1 \qquad \boxed{\Gamma; (\Xi \cap \Delta'); (\Delta^I \cap \Delta')}^{7} \setminus \Delta^O \Longrightarrow_v G_2}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v G_1 \otimes G_2} \otimes_1 \mathbf{io.9} \\[2em]
\dfrac{\boxed{\Gamma; \cdot; (\Xi, \Delta^I)}^{6} \setminus \Delta' \Longrightarrow_1 G_1 \qquad \Gamma; \cdot; \Delta' \setminus \Delta^O \Longrightarrow_v G_2}{\Gamma; \Xi; \Delta^I \setminus \boxed{\Delta^I \cap \Delta^O}^{8} \Longrightarrow_1 G_1 \otimes G_2} \otimes_2 \mathbf{io.9}
\end{array}\right\}$$

$$\left\{ \dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v G_1}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v G_1 \oplus G_2} \oplus_1 \mathbf{io.9} \qquad \dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v G_2}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v G_1 \oplus G_2} \oplus_2 \mathbf{io.9} \right\}$$

$$\dfrac{\boxed{\Gamma; (\Xi, D); \Delta^I}^{9} \setminus \Delta^O \Longrightarrow_v G}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v D \multimap G} \multimap \mathbf{io.9} \qquad \dfrac{\boxed{(\Gamma, D); \Xi; \Delta^I}^{9} \setminus \Delta^O \Longrightarrow_v G}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v D \rightarrow G} \rightarrow \mathbf{io.9}$$

$$\dfrac{\boxed{\Gamma; \cdot; \cdot}^{10} \setminus \Delta^{\perp} \Longrightarrow_v G}{\boxed{\Gamma; \cdot; \Delta^I}^{2} \setminus \Delta^I \Longrightarrow_0 !G} \, !\, \mathbf{io.9}$$

$$\dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v [c/x]G}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v \forall x.G} \forall \mathbf{io.9^*} \qquad \dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v [t/x]G}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v \exists x.G} \exists \mathbf{io.9}$$

Figure 10: Resource consumption for *LHHF*, version 0.9 (Continued)

[6] `make_non_strict`
[7] `intersect_input`
[8] `intersect`
[9] `d_scan`
[10] `erase`

5 Resource consumption for LHHF, version 0.9

Version 0.8 of the proof system forbids the right branch of the rules for $\&$ to access resources that the left branch spared. However, it does not force the right branch to consume all the resources used in the left part of the proof tree, at least not at an early enough stage. Consider the following examples that shows the problem in the presence of nested occurrences of $\&$.

```
test :- ((a,erase) & b), d.

LINEAR a.
LINEAR c.
LINEAR d.

b :- c & (write_sans "I may diverge", a).    % Fails
b :- a, c.                                   % Succeeds
```

We will rely on the abstract syntax of our proof systems to describe the computation originating from the resolution of the query `?- test`. Specifically, let Γ consist of the two clauses for b and of the definition of *test*, and W represent the output subgoal, logically equivalent to $\mathbf{1}$. After one application of rule $\mathbf{d/io.8'}$, we are reduced to solving $\Gamma; a, c, d \Longrightarrow' ((a \otimes \top) \& b) \otimes d$. We do not know yet which of the total consumption rules for the tensor to use, but both require to solve the partial consumption judgment $\Gamma; a, c, d \backslash \Delta_1^O \Longrightarrow_{v_1} (a \otimes \top) \& b$ for still unknown values of Δ_1^O and v_1. At this point the rules for $\&$ look for a proof of $\Gamma; a, c, d \backslash \Delta_2^O \Longrightarrow_{v_2} a \otimes \top$. It is easy to verify that this proof succeeds with $\Delta_2^O = c, d$ and $v_2 = 1$. Therefore, the computation proceeds with either rule $\&_2$ **io.8** or $\&_3$ **io.8**, and the goal $\Gamma; a, c, d \backslash \Delta_3^O \Longrightarrow_{v_3} b$. Notice at this point that this branch of the proof tree must consume a for the goal $(a \otimes \top) \& b$ to succeed, but nothing in this judgment enforces this behavior explicitly; this will cause a failure to be detected only very late. The first clause for b will be attempted, reducing this goal to that of proving $\Gamma; a, c, d \backslash \Delta_3^O \Longrightarrow_{v_3} c \& (W \otimes a)$. The rules for $\&$ reduce this judgment to the goal $\Gamma; a, c, d \backslash \Delta_4^O \Longrightarrow_{v_4} c$ which succeeds with $D_4^O = a, d$ and $v_4 = 0$. Notice that a has not been consumed while solving c, but the computation is allowed to proceed, and it does so by solving $\Gamma; c \Longrightarrow' W \otimes a$ which clearly fails but only after having executed W, i.e. printing the message. At this point, backtracking occurs and the second clause for b is attempted. An easy simulation shows that his attempt succeeds.

This behavior is not satisfactory. In this example a is a resource that must be consumed while solving the goal b while the other linear resources available, c and d, might be consumed while proving this goal, but do not have to. Instead, our current system passes all these resources indiscriminately to b as its input linear context.

We obtain the desired behavior in version 0.9 of the resource management system by considering a judgment of the form $\Gamma; \Xi; \Delta^I \backslash \Delta^O \Longrightarrow_v G$. The input linear context is logically subdivided into two parts: the *strict* context Ξ that must be consumed during the resolution of the goal G, and the *non-strict* context Δ^I which contents might be consumed while solving G; whatever portion of Δ^I is not used for this purpose is returned as the output context Δ^O. This unique relation captures the semantics of both the partial and the total consumption judgment of system 0.8. Indeed the strict context is treated as the linear context of total consumption, while the non-strict context is managed similarly to the input context of the partial consumption rules.

The rules defining the semantics of this judgment are represented in figure 9 and 10. We will now briefly describe their principal characteristics.

Since we have split the linear context, we need to provide two rules for accessing a linear formula when the goal is atomic. These are rules $\mathbf{d}_1$**/io.9** and $\mathbf{d}_2$**/io.9**. Rules **I io.9** and **1 io.9** are straightforward: the strict context must be empty while solving them while the non-strict context is passed over as output; notice that applying these rules forces us to check that the strict context is empty. The rule for $\top$ ($\top$ **io.9**) deletes whatever portion of the strict context it is provided with, and forwards as output its non-strict context.

The rules for $\&$ are more interesting. In order to solve the goal $G_1 \& G_2$ with respect to the linear context $\Xi; \Delta^I$, we first solve G_1 in $\Xi; \Delta^I$, obtaining the output context Δ^O, and then we can take two different courses of action depending on the value of the slack indicator. If no slack is available (rule $\&_1$ **io.9**), G_2 must consume everything that has been used by G_1, i.e. Ξ as well as $\Delta^I - \Delta^O$. These two components are packed into the strict context of G_2. Since this goal is not allowed to consume any other resources, it is given an empty non-strict context. The output context must clearly be empty $(\cdot)$. We use the superscript $^\perp$ to indicate entities which value is either trivial or has to be ignored; we use instead exact values, such as $\cdot$ to indicate that a test has to be performed to verify that the entity has this value (see rule **I io.9** for instance). If the resolution of G_1 has encountered an occurrence of $\top$ and slack is admitted (rule $\&_2$ **io.9**), G_2 must still consume every resource used by G_1 (i.e. $\Xi, \Delta^I - \Delta'$), but is also allowed to access the resources not used by this goal (Δ'), therefore, we give them as the non-strict context of G_2. The output context of this goal is the output context of the overall computation.

When solving a goal of the form $G_1 \otimes G_2$, the strict context Ξ must be consumed by either G_1 or G_2. Since the first of these subgoals may use an arbitrary part of Ξ as well as some portion of the non-strict context Δ^I, we put both Ξ and Δ^I in the non-strict context of G_1 and leave the strict context empty. Again how to solve G_2 depends on the value of the $\top$-flag. If no slack is allowed (rule $\otimes_1$ **io.9**), G_2 must consume whatever portion of Ξ G_1 did not use, and possibly some formulas in Δ^I that this goal did not access. Therefore, we need to restore the remaining parts of Ξ and Δ^I as the strict an non-strict context of G_2 respectively. We intersect these multisets with the output context Δ' of G_1 to prevent G_2 to access used resources. If the slack indicator was set by G_1 (rule $\otimes_2$ **io.9**, all the strict resources in the original Ξ can be pumped back to G_1 in the case G_2 does not use them. Therefore we call this goal with an empty strict context and the output context of G_1, Δ' as its non-strict context. We must be careful, however, not to return strict resources from Ξ as the output context of $G_1 \otimes G_2$, therefore we intersect the context returned by G_2 with the original input context Δ^I of the composed goal.

The rule dealing with linear implication ($\multimap$ **io.9**) provides a reasonable solution to the management of the consequent of a goal of the form $D \multimap G$. Since D has to be used while proving G, we simply put it into the strict context of this subgoal. In this way we avoid complex operations depending on the value of the slack indicator.

Finally, the rule for ! (! **io.9**) expects an empty strict context and has its subgoal solved in a non-strict context that should be empty as well. The original non-strict context is returned as output.

$$\left\{ \frac{D \gg a \setminus G \qquad \Gamma;\Xi;\Delta^I\setminus\Delta^O \Longrightarrow_v^s G}{\boxed{\Gamma;\Xi;(\Delta^I,D)}^{1} \setminus \Delta^O \Longrightarrow_v^s a}\,\mathbf{d_1/io.91} \qquad \frac{D \gg a\setminus G \qquad \Gamma;\Xi;\Delta^I\setminus\Delta^O \Longrightarrow_v^1 G}{\boxed{\Gamma;(\Xi,D);\Delta^I}^{1}\setminus\Delta^O\Longrightarrow_v^s a}\,\mathbf{d_2/io.91}\right\}$$

$$\frac{D \gg a\setminus G \qquad (\Gamma,D);\Xi;\Delta^I\setminus\Delta^O \Longrightarrow_v^s G}{\boxed{(\Gamma,D);\Xi;\Delta^I}^{1}\setminus\Delta^O \Longrightarrow_v^s a}\,\mathbf{!d/io.91}$$

$$\left\{ \frac{}{\Gamma;\Xi^{\perp};\Delta^I\setminus\Delta^I \Longrightarrow_0^0 a \doteq a}\,\mathbf{I_1\ io.91} \qquad \frac{}{\boxed{\Gamma;\cdot;\Delta^I}^{2}\setminus\Delta^I \Longrightarrow_0^1 a \doteq a}\,\mathbf{I_2\ io.91}\right\}$$

$$\left\{ \frac{}{\Gamma;\Xi^{\perp};\Delta^I\setminus\Delta^I \Longrightarrow_0^0 \mathbf{1}}\,\mathbf{1_1\ io.91} \qquad \frac{}{\boxed{\Gamma;\cdot;\Delta^I}^{2}\setminus\Delta^I \Longrightarrow_0^1 \mathbf{1}}\,\mathbf{1_2\ io.91}\right\}$$

$$\left\{ \frac{}{\Gamma;\Xi^{\perp};\Delta^I\setminus\Delta^I \Longrightarrow_1^0 \top}\,\mathbf{\top_1\ io.91} \qquad \frac{}{\boxed{\Gamma;\Xi;\Delta^I}^{3}\setminus\Delta^I \Longrightarrow_1^1 \top}\,\mathbf{\top_2\ io.91}\right\}$$

$$\left\{ \begin{array}{c} \dfrac{\Gamma;\Xi;\Delta^I\setminus\Delta^O \Longrightarrow_0^s G_1 \qquad \boxed{\Gamma;(\Xi,\Delta^I-\Delta^O);\cdot}^{4}\setminus\Delta^{\perp} \Longrightarrow_v^1 G_2}{\Gamma;\Xi;\Delta^I\setminus\Delta^O \Longrightarrow_0^s G_1 \mathbin{\&} G_2}\,\mathbf{\&_1 io.91} \\[3em] \dfrac{\Gamma;\Xi;\Delta^I\setminus\Delta' \Longrightarrow_1^s G_1 \qquad \boxed{\Gamma;(\Xi,\Delta^I-\Delta');\Delta'}^{5}\setminus\Delta^O \Longrightarrow_v^1 G_2}{\Gamma;\Xi;\Delta^I\setminus\Delta^O \Longrightarrow_v^s G_1 \mathbin{\&} G_2}\,\mathbf{\&_2 io.91} \end{array}\right\}$$

Figure 11: Resource consumption for *LHHF*, version 0.91

[1] `pick_resource`

[2] `strict`

[3] `erase_strict`

[4] `strict_difference_and_erase`

[5] `strict_difference_and_keep`

$$\left\{\begin{array}{c}
\dfrac{\Gamma; \cdot; \Delta^I \setminus \Delta' \Longrightarrow^0_{v_1} G_1 \qquad \Gamma; \cdot; \Delta' \setminus \Delta^O \Longrightarrow^0_{v_2} G_2}{\Gamma; \Xi^\top; \Delta^I \setminus \Delta^O \Longrightarrow^0_{v_1 \vee v_2} G_1 \otimes G_2} \otimes_1 \mathbf{io.91} \\[3ex]
\dfrac{\boxed{\Gamma; \cdot; (\Xi, \Delta^I)}^{6} \setminus \Delta' \Longrightarrow^0_0 G_1 \qquad \boxed{\Gamma; (\Xi \cap \Delta'); (\Delta^I \cap \Delta')}^{7} \setminus \Delta^O \Longrightarrow^1_v G_2}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^1_v G_1 \otimes G_2} \otimes_2 \mathbf{io.91} \\[3ex]
\dfrac{\boxed{\Gamma; \cdot; (\Xi, \Delta^I)}^{6} \setminus \Delta' \Longrightarrow^0_1 G_1 \qquad \Gamma; \cdot; \Delta' \setminus \Delta^O \Longrightarrow^0_v G_2}{\Gamma; \Xi; \Delta^I \setminus \boxed{\Delta^I \cap \Delta^O}^{8} \Longrightarrow^1_1 G_1 \otimes G_2} \otimes_3 \mathbf{io.91}
\end{array}\right\}$$

$$\left\{ \dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v G_1}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v G_1 \oplus G_2} \oplus_1 \mathbf{io.91} \qquad \dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v G_2}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v G_1 \oplus G_2} \oplus_2 \mathbf{io.91} \right\}$$

$$\dfrac{\Gamma; (\Xi, D); \Delta^I \setminus \Delta^O \Longrightarrow^1_v G}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v D \multimap G} \multimap \mathbf{io.91} \qquad \dfrac{(\Gamma, D); \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v G}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v D \rightarrow G} \rightarrow \mathbf{io.91}$$

$$\left\{ \dfrac{\boxed{\Gamma; \Xi^\perp; \cdot}^{10} \setminus \Delta^\perp \Longrightarrow^0_v G}{\Gamma; \Xi^\perp; \Delta^I \setminus \Delta^I \Longrightarrow^0_0 !G} !_1\, \mathbf{io.91} \qquad \dfrac{\boxed{\Gamma; \cdot; \cdot}^{10} \setminus \Delta^\perp \Longrightarrow^0_v G}{\boxed{\Gamma; \cdot; \Delta^I}^{2} \setminus \Delta^I \Longrightarrow^1_0 !G} !_2\, \mathbf{io.91} \right\}$$

$$\dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v [c/x]G}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v \forall x.G} \forall\, \mathbf{io.91}^* \qquad \dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v [t/x]G}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow^s_v \exists x.G} \exists\, \mathbf{io.91}$$

Figure 12: Resource consumption for *LHHF*, version 0.91 (Continued)

[6] `make_non_strict`
[7] `intersect_input`
[8] `intersect`
[9] `d_scan`
[10] `erase`

6 Resource consumption for LHHF, version 0.91

Version 0.9 of the proof system for resource management provides satisfactory solutions to all the problems we complained about in the previous sections. Unfortunately, the price to pay for this is rather high since almost all rules involve complex operations on the context (either exhaustive tests or shuffling formulas from the strict to the non-strict context or vice versa). Versions 0.91 to 0.94 attempt to reduce this overhead by keeping track of different kind of information. These systems were defined according to their sequence number and therefore represent the historical development of our ideas. The best and current system is version 0.94, discussed in §9.

The implementation of version 0.9 of resource management represents all three context in the same tree structure, differentiating the role of each formula by means of a tag. Consumed formulas are kept in that same tree. Therefore, each time we perform a test to check if the strict context is empty (rules **I io.9**, **1 io.9** and ! **io.9**), we have to visit all the formulas present in the context. This is clearly an expensive operation. Similarly, rules $\top$ **io.9** and $\otimes_i$ **io.9** visit the context to perform operations on strict formulas.

Version 0.91 (figures 11 and 12) associates with each judgment a boolean flag, s, written as a superscript to the sequent arrow, which indicates that the strict context is empty. If the flag is set, this context might contain some formulas, if it has value 0, then there is for sure no formula in it. This is clearly an approximation of the contents of the strict context. Before doing any operation concerned with this context, we test this flag to check if it is empty. If so, we can save on most operations. Otherwise, we take the same course of actions as in system 0.9.

This technical device is beneficial in the rules that deal with the connectives **1**, $\top$, $\otimes$ and !.

$$\left\{ \frac{D \gg a \setminus G \qquad \Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v^{s,1} G}{\boxed{\Gamma; \Xi; (\Delta^I, D)}^{1} \setminus \Delta^O \Longrightarrow_v^{s,t} a}\ \mathbf{d_1/io.92} \qquad \frac{D \gg a \setminus G \qquad \Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v^{1,t} G}{\boxed{\Gamma; (\Xi, D); \Delta^I}^{1} \setminus \Delta^O \Longrightarrow_v^{s,t} a}\ \mathbf{d_2/io.92} \right\}$$

$$\frac{D \gg a \setminus G \qquad (\Gamma, D); \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v^{s,t} G}{\boxed{(\Gamma, D); \Xi; \Delta^I}^{1} \setminus \Delta^O \Longrightarrow_v^{s,t} a}\ \mathbf{!d/io.92}$$

$$\left\{ \frac{}{\Gamma; \Xi^{\perp}; \Delta^I \setminus \Delta^I \Longrightarrow_0^{0,t} a \doteq a}\ \mathbf{I_1\ io.92} \qquad \frac{}{\boxed{\Gamma; \cdot; \Delta^I}^{2} \setminus \Delta^I \Longrightarrow_0^{1,t} a \doteq a}\ \mathbf{I_2\ io.92} \right\}$$

$$\left\{ \frac{}{\Gamma; \Xi^{\perp}; \Delta^I \setminus \Delta^I \Longrightarrow_0^{0,t} \mathbf{1}}\ \mathbf{1_1\ io.92} \qquad \frac{}{\boxed{\Gamma; \cdot; \Delta^I}^{2} \setminus \Delta^I \Longrightarrow_0^{1,t} \mathbf{1}}\ \mathbf{1_2\ io.92} \right\}$$

$$\left\{ \frac{}{\Gamma; \Xi^{\perp}; \Delta^I \setminus \Delta^I \Longrightarrow_1^{0,t} \top}\ \top\,\mathbf{io.92} \qquad \frac{}{\boxed{\Gamma; \Xi; \Delta^I}^{3} \setminus \Delta^I \Longrightarrow_1^{1,t} \top}\ \top\,\mathbf{io.92} \right\}$$

$$\left\{ \begin{array}{c}
\dfrac{\Gamma; \Xi; \cdot \setminus \Delta^{\perp} \Longrightarrow_{v_1}^{s,0} G_1 \qquad \Gamma; \Xi; \cdot \setminus \Delta^{\perp} \Longrightarrow_{v_2}^{s,0} G_2}{\Gamma; \Xi; \Delta^{\perp} \setminus \Delta^{\perp} \Longrightarrow_{v_1 \wedge v_2}^{s,0} G_1 \,\&\, G_2}\ \&_1\mathbf{io.92} \\[2em]
\dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_0^{s,1} G_1 \qquad \boxed{\Gamma; (\Xi, \Delta^I - \Delta^O); \cdot}^{4} \setminus \Delta^{\perp} \Longrightarrow_v^{1,0} G_2}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_0^{s,1} G_1 \,\&\, G_2}\ \&_2\mathbf{io.92} \\[2em]
\dfrac{\Gamma; \Xi; \Delta^I \setminus \Delta' \Longrightarrow_1^{s,1} G_1 \qquad \boxed{\Gamma; (\Xi, \Delta^I - \Delta'); \Delta'}^{5} \setminus \Delta^O \Longrightarrow_v^{1,1} G_2}{\Gamma; \Xi; \Delta^I \setminus \Delta^O \Longrightarrow_v^{s,1} G_1 \,\&\, G_2}\ \&_3\mathbf{io.92}
\end{array} \right\}$$

Figure 13: Resource consumption for *LHHF*, version 0.92

[1] `pick_resource`
[2] `strict`
[3] `erase_strict`

$$\left\{\begin{array}{c}
\dfrac{\Gamma;\cdot;\Delta^I\backslash\Delta' \Longrightarrow_0^{0,t} G_1 \qquad \Gamma;\cdot;\Delta'\backslash\Delta^O \Longrightarrow_v^{0,t} G_2}{\Gamma;\Xi^{\perp};\Delta^I\backslash\Delta^O \Longrightarrow_v^{0,t} G_1 \otimes G_2}\otimes_1\mathbf{io.92}\\[2em]
\dfrac{\boxed{\Gamma;\cdot;(\Xi,\Delta^I)}^{6}\backslash\Delta' \Longrightarrow_0^{0,1} G_1 \qquad \boxed{\Gamma;(\Xi\cap\Delta');(\Delta^I\cap\Delta')}^{7}\backslash\Delta^O \Longrightarrow_v^{1,t} G_2}{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{1,t} G_1 \otimes G_2}\otimes_2\mathbf{io.92}\\[2em]
\dfrac{\boxed{\Gamma;\cdot;(\Xi,\Delta^I)}^{6}\backslash\Delta' \Longrightarrow_1^{0,1} G_1 \qquad \Gamma;\cdot;\Delta'\backslash\Delta^O \Longrightarrow_v^{0,1} G_2}{\Gamma;\Xi;\Delta^I\backslash\boxed{\Delta^I\cap\Delta^O}^{8} \Longrightarrow_1^{1,t} G_1 \otimes G_2}\otimes_3\mathbf{io.92}
\end{array}\right\}$$

$$\left\{\dfrac{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} G_1}{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} G_1 \oplus G_2}\oplus_1\mathbf{io.92} \qquad \dfrac{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} G_2}{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} G_1 \oplus G_2}\oplus_2\mathbf{io.92}\right\}$$

$$\dfrac{\boxed{\Gamma;(\Xi,D);\Delta^I}^{9}\backslash\Delta^O \Longrightarrow_v^{1,t} G}{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} D \multimap G}\multimap\mathbf{io.92} \qquad \dfrac{\boxed{(\Gamma,D);\Xi;\Delta^I}^{9}\backslash\Delta^O \Longrightarrow_v^{s,t} G}{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} D \rightarrow G}\rightarrow\mathbf{io.92}$$

Figure 14: Resource consumption for *LHHF*, version 0.92 (Continued)

[4] `strict_difference_and_erase`

[5] `strict_difference_and_keep`

[6] `make_non_strict`

[7] `intersect_input`

[8] `intersect`

[9] `d_scan`

[10] `erase`

$$\left\{\begin{array}{cc}
\dfrac{\Gamma;\Xi^\perp;\Delta_1^\perp\backslash\Delta_2^\perp \Longrightarrow_v^{0,0} G}{\Gamma;\Xi^\perp;\Delta_1^\perp\backslash\Delta_2^\perp \Longrightarrow_0^{0,0} !G}\ !_1\ \mathbf{io.92} &
\dfrac{\Gamma;\cdot;\Delta_1^\perp\backslash\Delta_2^\perp \Longrightarrow_v^{0,0} G}{\boxed{\Gamma;\cdot;\Delta_1^\perp}^{2}\backslash\Delta_2^\perp \Longrightarrow_0^{1,0} !G}\ !_2\ \mathbf{io.92} \\[2ex]
\dfrac{\boxed{\Gamma;\Xi^\perp;\cdot}^{10}\backslash\Delta^\perp \Longrightarrow_v^{0,0} G}{\Gamma;\Xi^\perp;\Delta^I\backslash\Delta^I \Longrightarrow_0^{0,1} !G}\ !_3\ \mathbf{io.92} &
\dfrac{\boxed{\Gamma;\cdot;\cdot}^{10}\backslash\Delta^\perp \Longrightarrow_v^{0,0} G}{\boxed{\Gamma;\cdot;\Delta^I}^{2}\backslash\Delta^I \Longrightarrow_0^{1,1} !G}\ !_4\ \mathbf{io.92}
\end{array}\right\}$$

$$\dfrac{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} [c/x]G}{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} \forall x.G}\ \forall\ \mathbf{io.92}^* \qquad \dfrac{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} [t/x]G}{\Gamma;\Xi;\Delta^I\backslash\Delta^O \Longrightarrow_v^{s,t} \exists x.G}\ \exists\ \mathbf{io.92}$$

Figure 15: Resource consumption for *LHHF*, version 0.92 (Continued)

7 Resource consumption for LHHF, version 0.92

The next version of the resource management system (version 0.92) pushes this idea further by associating to each judgment a *non-strict flag t* that has value 0 only if the non-strict context is for sure empty, while is assigned the value 1 if this context might contain some formula.

Testing this flag allows to save even more computation when solving an additive conjunction and an exponential. Moreover it provides a better approximation to the value of the strict flag s.

The rules of this system are presented in figures 13, 14 and 15.

$$
\left\{
\frac{D \gg a \setminus G \qquad \Gamma;\ \overbrace{\Xi;\Delta^I}^{(s,t-1)\mapsto t'}\ \setminus \Delta^O \Longrightarrow_v\ G}{\boxed{\Gamma; \underbrace{\Xi; (\Delta^I, D)}_{(s,t)\mapsto t'}}^{1} \setminus \Delta^O \Longrightarrow_v\ a}\ \mathbf{d_1/io.93}
\qquad
\frac{D \gg a \setminus G \qquad \Gamma;\ \overbrace{\Xi;\Delta^I}^{(s-1,t)\mapsto t'}\ \setminus \Delta^O \Longrightarrow_v\ G}{\boxed{\Gamma; \underbrace{(\Xi, D); \Delta^I}_{(s,t)\mapsto t'}}^{1} \setminus \Delta^O \Longrightarrow_v\ a}\ \mathbf{d_2/io.93}
\right\}
$$

$$
\frac{D \gg a \setminus G \qquad (\Gamma, D);\ \overbrace{\Xi;\Delta^I}^{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_v\ G}{\boxed{(\Gamma, D);\ \underbrace{\Xi;\Delta^I}_{(s,t)\mapsto t'}}^{1} \setminus \Delta^O \Longrightarrow_v\ a}\ \mathbf{!d/io.93}
$$

$$
\frac{}{\boxed{\Gamma;\ \underbrace{\cdot;\Delta^I}_{(s,t)\mapsto t}}^{2} \setminus \Delta^I \Longrightarrow_0\ a \doteq a}\ \mathbf{I\ io.93}
\qquad
\frac{}{\boxed{\Gamma;\ \underbrace{\cdot;\Delta^I}_{(s,t)\mapsto t}}^{2} \setminus \Delta^I \Longrightarrow_0\ \mathbf{1}}\ \mathbf{1\ io.93}
$$

$$
\frac{}{\boxed{\Gamma;\ \underbrace{\Xi;\Delta^I}_{(s,t)\mapsto t}}^{3} \setminus \Delta^I \Longrightarrow_1\ \top}\ \top\ \mathbf{io.93}
$$

$$
\left\{
\begin{array}{c}
\dfrac{\Gamma;\ \overbrace{\Xi;\Delta^I}^{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_0\ G_1 \qquad \boxed{\Gamma; \overbrace{(\Xi, \Delta^I - \Delta^O); \cdot}^{(s+t-t',0)\mapsto t^{\perp}}}^{4}\ \setminus \Delta^{\perp} \Longrightarrow_v\ G_2}{\Gamma;\ \underbrace{\Xi;\Delta^I}_{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_0\ G_1 \,\&\, G_2}\ \&_1\mathbf{io.93}
\\[2em]
\dfrac{\Gamma;\ \overbrace{\Xi;\Delta^I}^{(s,t)\mapsto t'} \setminus \Delta' \Longrightarrow_1\ G_1 \qquad \boxed{\Gamma; \overbrace{(\Xi, \Delta^I - \Delta'); \Delta'}^{(s+t-t',t')\mapsto t''}}^{5}\ \setminus \Delta^O \Longrightarrow_v\ G_2}{\Gamma;\ \underbrace{\Xi;\Delta^I}_{(s,t)\mapsto t''} \setminus \Delta^O \Longrightarrow_v\ G_1 \,\&\, G_2}\ \&_2\mathbf{io.93}
\end{array}
\right\}
$$

Figure 16: Resource consumption for *LHHF*, version 0.93

[1] `pick_resource`
[2] `strict`

$$\frac{\boxed{\Gamma; \cdot; \overbrace{(\Xi, \Delta^I)}^{(0,s+t)\mapsto t'}}^{6} \setminus \Delta' \Longrightarrow_0 G_1 \qquad \boxed{\Gamma; \overbrace{(\Xi\cap\Delta'); (\Delta^I\cap\Delta')}^{(s^7,t^7)\mapsto t''}}^{7} \setminus \Delta^O \Longrightarrow_v G_2}{\Gamma;\ \underbrace{\Xi; \Delta^I}_{(s,t)\mapsto t''} \setminus \Delta^O \Longrightarrow_v G_1 \otimes G_2} \otimes_1\mathbf{io.93}$$

$$\frac{\boxed{\Gamma; \cdot; \overbrace{(\Xi, \Delta^I)}^{(0,s+t)\mapsto t'}}^{6} \setminus \Delta' \Longrightarrow_1 G_1 \qquad \Gamma;\ \overbrace{\cdot; \Delta'}^{(0,t')\mapsto t''} \setminus \Delta^O \Longrightarrow_v G_2}{\Gamma;\ \underbrace{\Xi; \Delta^I}_{(s,t)\mapsto t^8} \setminus \boxed{\Delta^I \cap \Delta^O}^{8} \Longrightarrow_1 G_1 \otimes G_2} \otimes_2\mathbf{io.93}$$

$$\frac{\Gamma;\ \overbrace{\Xi; \Delta^I}^{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_v G_1}{\Gamma;\ \underbrace{\Xi; \Delta^I}_{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_v G_1 \oplus G_2} \oplus_1\mathbf{io.93} \qquad \frac{\Gamma;\ \overbrace{\Xi; \Delta^I}^{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_v G_2}{\Gamma;\ \underbrace{\Xi; \Delta^I}_{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_v G_1 \oplus G_2} \oplus_2\mathbf{io.93}$$

Figure 17: Resource consumption for *LHHF*, version 0.93 (Continued)

[3] `erase_strict`

[4] `strict_difference_and_erase`

[5] `strict_difference_and_keep`

[6] `make_non_strict`

[7] `intersect_input`

[8] `intersect`

[9] `augment`

[10] `take_back`

[11] `erase`

$$\frac{\boxed{\Gamma; \overbrace{(\Xi, D); \Delta^I}^{(s+s^9,t)\mapsto t'}}^{9} \setminus \Delta^O \Longrightarrow_v G}{\Gamma; \underbrace{\Xi; \Delta^I}_{(s,t)\mapsto t'} \setminus \boxed{\Delta^O}^{10} \Longrightarrow_v D \multimap G} \multimap \mathbf{io.93}$$

$$\frac{\boxed{(\Gamma, D);\ \overbrace{\Xi; \Delta^I}^{(s,t)\mapsto t'}}^{9} \setminus \Delta^O \Longrightarrow_v G}{\Gamma; \underbrace{\Xi; \Delta^I}_{(s,t)\mapsto t'} \setminus \boxed{\Delta^O}^{10} \Longrightarrow_v D \rightarrow G} \rightarrow \mathbf{io.93}$$

$$\frac{\boxed{\Gamma;\ \overbrace{\cdot;\cdot}^{(0,0)\mapsto t^{\perp}}}^{11} \setminus \Delta^{\perp} \Longrightarrow_v G}{\boxed{\Gamma;\ \underbrace{\cdot;\Delta^I}_{(s,t)\mapsto t}}^{2} \setminus \Delta^I \Longrightarrow_0 !G} \,!\,\mathbf{io.93}$$

$$\frac{\Gamma;\ \overbrace{\Xi; \Delta^I}^{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_v [c/x]G}{\Gamma;\ \underbrace{\Xi; \Delta^I}_{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_v \forall x.G} \forall\, \mathbf{io.93}^{*}$$

$$\frac{\Gamma;\ \overbrace{\Xi; \Delta^I}^{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_v [t/x]G}{\Gamma;\ \underbrace{\Xi; \Delta^I}_{(s,t)\mapsto t'} \setminus \Delta^O \Longrightarrow_v \exists x.G} \exists\, \mathbf{io.93}$$

Figure 18: Resource consumption for *LHHF*, version 0.93 (Continued)

8 Resource consumption for LHHF, version 0.93

The flags that characterize version 0.92 of the proof system provide correct but imprecise information about the contents of the two parts of the linear context. In particular, an empty context might be labelled as non-empty due to the roughness of the data we can store in our two flags. We can gain in efficiency if we gain in precision.

Version 0.93 of context management achieves total accuracy by substituting the flags of the previous version with two counters that record exactly how many formulas are present in either part of the linear context. These counters (that we continue to call s and t) are integrated into the data structure used to represent the contexts in order to achieve data abstraction. The operations that manipulate the context are in charge of maintaining the value of these counters correct. In particular, each time an output context is produced, we must make available the number of formulas present in it.

The resulting proof system is presented in figures 16, 17 and 18. Notice that it corresponds to version 0.9 of resource management with the addition of annotations specifying how to calculate the number of formulas in the strict, non-strict and output contexts.

9 Resource consumption for LHHF, version 0.94

Rules $\otimes_i$ **io.93** in version 0.93 of the proof system for context management perform a very expensive operation each time a tensor is to be solved (and the strict context is not empty). Multiplicative conjunction is however the most commonly used connective in *Lolli*. Notice that these rules solve left conjunct in an empty strict context. Therefore, if this goal contains nested occurrences of $\otimes$, no context shuffling is required to solve the relative subgoals. We can take advantage of this observation by requiring $\otimes$ to be parsed as a left associative operator. System 0.94 is just a version of system 0.93 in which this requirement is enforced.

References

[1] J.S. Hodas and D. Miller. Logic Programming in a Fragment of Linear Logic. *Journal of Information and Computation*, 110(2):327–365, 1994.

[2] F. Pfenning. *Computation and Deduction*, draft notes for a course given at Carnegie Mellon University during the Spring semester of 1994.