Resource-distribution via Boolean constraints (extended abstract)

Harland, James; Pym, David

ISSN 1369-1961

Department of Computer Science

Technical Report No. 733

Resource-distribution via Boolean constraints (extended abstract)

James Harland & David Pym

QUEEN MARY
AND WESTFIELD COLLEGE
UNIVERSITY OF LONDON

April 1997

To appear: Proc. CADE-14, W. McCune, editor,
LNCS, Springer-Verlag, 1997.

Resource-distribution via Boolean constraints (extended abstract)

James Harland[1] David Pym[2]

[1] Department of Computer Science, Royal Melbourne Institute of Technology
[2] Queen Mary and Westfield College, University of London

Abstract. Proof-search (the basis of logic programming) with multiplicative inference rules, such as linear logic's $\otimes$R and $⅋$L, is problematic because of the required non-deterministic splitting of resources. Similarly, searching with additive rules such as &L and $\oplus$R requires a non-deterministic choice between two formulae. Many strategies which resolve such non-determinism, either locally or globally, are available.
We present a characterization of a range of strategies for distributing and selecting resources in linear sequent calculus proof-search via a sequent calculus annotated with Boolean constraints. Strategies are characterized by calculations of solutions of sets of Boolean equations generated by searches. Our characterization encompasses lazy (or local), eager (or global) and intermediate (mixed local and global) strategies.

1 Introduction

The formulation of linear logic [4] as a sequent calculus [3, 4, 13, 2] makes *essential* use of the *multiplicative* formulation of binary rules, in which the resources available to derive the principal formula in each of the premisses must be combined to form the resources available to derive the principal formula of the conclusion. For example, in each of the rules for $\otimes$ on the right and $⅋$ on the left,

$$\frac{\Gamma_1 \vdash p_1, \Delta_1 \quad \Gamma_2 \vdash p_2, \Delta_2}{\Gamma_1, \Gamma_2 \vdash p_1 \otimes p_2, \Delta_1, \Delta_2} \otimes R \qquad \text{and} \qquad \frac{\Gamma_1, p_1 \vdash \Delta_1 \quad \Gamma_2, p_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2, p_1 ⅋ p_2 \vdash \Delta_1, \Delta_2} ⅋ L,$$

the antecedents Γ_1 and Γ_2 and succedents Δ_1 and Δ_2 must be combined to form, respectively, the antecedent and succedent of the conclusion.

From the point of view of proof-search, in which rules are read as reduction operators from conclusion to premisses, multiplicative rules are problematic because they are (highly) *non-deterministic*.[1] Faced with a sequent $\Gamma \vdash p_1 \otimes p_2, \Delta$, and having decided to apply the $\otimes$R rule, it is necessary to split each of Γ and Δ so as to determine the premisses $\Gamma_1 \vdash p_1, \Delta_1$ and $\Gamma_2 \vdash p_2, \Delta_2$ such that $\Gamma = \Gamma_1, \Gamma_2$ and $\Delta = \Delta_1, \Delta_2$. The number of such splittings is exponential in

[1] Such non-determinism causes the search space to be disjunctive.

the sizes of and Γ and Δ. Similar problems arise with the $\multimap$L and $\parr$L rules. It is clear that there are many possible strategies for determining multiplicative splittings.

A second source of non-determinism in proof-search arises in reducing the $\oplus$R and &L rules,

$$\frac{\Gamma \vdash p_i, \Delta}{\Gamma \vdash p_1 \oplus p_2, \Delta} \; (i = 1, 2) \quad \oplus\text{R} \qquad \text{and} \qquad \frac{\Gamma, p_i \vdash \Delta}{\Gamma, p_1 \,\&\, p_2 \vdash \Delta} \; (i = 1, 2) \quad \&\,\text{L},$$

in which we must choose between p_1 and p_2.

Closely related to each of the two sources of non-determinism we have discussed so far is that arising from the structural rules[2] of *weakening* and *contraction*,

$$\frac{\Gamma \vdash \Delta}{\Gamma, !p \vdash \Delta}\;\text{W!L}, \quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?p, \Delta}\;\text{W?R} \quad \text{and} \quad \frac{\Gamma, !p, !p \vdash \Delta}{\Gamma, !p \vdash \Delta}\;\text{C!L}, \quad \frac{\Gamma \vdash ?p, ?p, \Delta}{\Gamma \vdash ?p, \Delta}\;\text{C?R}.$$

In fact, we cannot fully analyze the behaviour of the multiplicative rules without considering weakening and contraction. Contraction cannot, in general, be permuted upwards past $\otimes$R. For example, consider the following derivation:

$$\cfrac{\cfrac{\Gamma_1, !p \vdash p_1, \Delta_1 \quad \Gamma_2, !p \vdash p_2, \Delta_2}{\Gamma_1, \Gamma_2, !p, !p \vdash p_1 \otimes p_2, \Delta_1, \Delta_2}\;\otimes\text{R}}{\Gamma_1, \Gamma_2, !p \vdash p_1 \otimes p_2, \Delta_1, \Delta_2}\;C!\text{L}.$$

The order of the inferences cannot be reversed because we must put a copy of $!p$ on each multiplicative branch. This aspect of the linear sequent calculus has been discussed elsewhere; there are sequent calculus systems in which such occurrences of weakening and contraction are encoded into the other rules [12, 10]. Hence we do not explicitly consider them here.

Two sources of non-determinism in proof-search remain: (i) the choice of a term t in the quantifier rules $\exists$R and $\forall$L; and (ii) the choice of rule instance, *i.e.*, which rule on which formula. Neither (i) nor (ii) is addressed in this paper: (i) because it can be treated independently via unification and has been addressed by Lincoln and Shankar [8]; and (ii) because it would take is into the realm of permutation theorems, which have been addressed by Pym and Harland [10], Andreoli [1], Miller [9], Lincoln and others, thereby interacting non-trivially with (i).[3] Our focus, then, is the distribution of propositional resources for a proof of fixed shape.[4] The main technical consideration is to make explicit the non-determinism implicit in the sequential rules, $\otimes$R, $\parr$L, $\multimap$L, &L and $\oplus$R.

Our technique for doing this is to attach a Boolean variable to each formula in a sequent. The rules of the calculus are then extended to include the relationships between these Boolean variables. Consequently, we do not need to determine the way in which the formulae are split between multiplicative branches at the time that the rule is applied. Rather, we can specify what constraints must be satisfied in order for the rule to be applicable. Different strategies will then correspond to different methods of solving the resulting set of Boolean equations.

[2] The rules **1**L and $\bot$R also belong to this class of non-determinism.

[3] Such considerations would take us in the direction of a matrix characterization of provability in linear logic.

[4] Note that we generally consider only propositional formulae for simplicity; there no inherent problem with quantifiers (see Definition 5 below).

2 A calculus with constraints for resource-distribution

In this section, we present a sequent calculus for linear logic in which the non-deterministic splitting of resources (side-formulae) at multiplicative reductions is *explicit*. This is done in order to provide an inference system which is independent of any strategy used to distribute the formulae, but which makes the necessary constraints explicit. Such a calculus will facilitate theoretical analyses, and systematic mechanical implementations, of resource-distribution. Whilst the ordinary sequent calculus satisfies the independence requirement, the lack of an explicit specification of the constraints involved makes it impossible to analyse strategies in such a framework. Hence our system will make this aspect of the non-determinism in the sequent calculus explicit, so that it can be analysed and various approaches to it can be compared.

We begin with an informal account of our intended class of resource-distribution strategies and proceed to establish our calculus with explicit constraints. We recover a formal account of our intended class of resource-distribution strategies.

2.1 Strategies for resource-distribution

We give an informal account of our intended class of resource-distribution strategies via a selection of examples. This class divides conveniently into three: *lazy*, *intermediate* and *eager*. We illustrate these three cases by applying each of them to searching for proofs of the following sequents: (1) $p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)$; (2) $p, p, q, q \vdash (p \otimes q) \,\&\, (p \otimes q)$; and (3) $(p \otimes q) \parr (p \otimes q) \vdash p \otimes q, p \otimes q$. These examples are *not* intended to illustrate any efficiency concerns. Rather, their purpose is to illustrate our analysis of different types of strategy in the presence of (1) multiplicativity, (2) additivity, and (3) multiple conclusions. Clearly, there are classes of sequents for which each type of strategy is likely to be the most efficient.

Lazy distribution. Firstly, consider (1). Here we distribute the formulae p, p, q, q in a *lazy* manner, *i.e.*, we first pass the entire collection to a chosen branch, which consumes the formulae needed, and passes on any excess to the next chosen branch (*cf.* [14]). This strategy is the one most commonly used in linear logic programming languages such as Lygon [6, 14] and Lolli [7].

Initially, we divide $(p \otimes q) \otimes (p \otimes q)$ into two copies of $(p \otimes q)$, and then one of these is further divided into p and q. Hence we arrive at the leaf $p, p, q, q \vdash p$, where it is clear that the formulae p, q, q are in excess, and are passed to the next branch, which then becomes $p, q, q \vdash q$. Here it is clear that p, q are the excess formulae, which are then passed to the next branch, giving us $p, q \vdash p \otimes q$.

This process is represented in the diagram below, in which each $\Im_i$ represents an as yet unknown multiset of formulae and $\leadsto$ denotes the *evaluation* consisting of closing a leaf and passing on the unconsumed resources, marked by crossing out with $/$:

$$\dfrac{\dfrac{\cancel{p}, p, \cancel{q}, \cancel{q} \vdash p \quad \Im_2 \vdash q}{p, p, q, q \vdash p \otimes q} \qquad \Im_1 \vdash p \otimes q}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} \quad \leadsto \quad \dfrac{\dfrac{\dfrac{}{p \vdash p} \quad \cancel{p}, \cancel{q}, q \vdash q}{p, p, q, q \vdash p \otimes q} \qquad \Im_1 \vdash p \otimes q}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} \quad \leadsto$$

$$\dfrac{\dfrac{\overline{p \vdash p} \quad \overline{q \vdash q}}{p, q \vdash p \otimes q} \qquad p, q \vdash p \otimes q}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} \quad \rightsquigarrow \quad \ldots$$

Following the remaining uncompleted branch, the leaf — $p, q \vdash p \otimes q$ — of which has now been generated, it remains to calculate a proof of $p, q \vdash p \otimes q$:

$$\ldots \quad \rightsquigarrow \quad \dfrac{p, \cancel{q} \vdash p \quad \Im_3 \vdash q}{p, q \vdash p \otimes q} \quad \rightsquigarrow \quad \dfrac{\overline{p \vdash p} \quad q \vdash q}{p, q \vdash p \otimes q} \quad \rightsquigarrow \quad \dfrac{\overline{p \vdash p} \quad \overline{q \vdash q}}{p, q \vdash p \otimes q} \quad .$$

Such a lazy strategy, as outlined above, amounts to a depth-first traversal of the proof tree, with the constraints on the distribution of formulae being propagated sequentially from one multiplicative branch to the next.

Secondly, turning to an example of an unsuccessful search, we see how our view of strategies handles an additive connective, namely &. Consider (2), the unprovable sequent $p, p, q, q \vdash (p \otimes q) \,\&\, (p \otimes q)$.

$$\dfrac{\dfrac{\cancel{p}, p, \cancel{q}, \cancel{q} \vdash p \quad \Im_1 \vdash q}{p, p, q, q \vdash p \otimes q} \qquad p, p, q, q \vdash p \otimes q}{p, p, q, q \vdash (p \otimes q) \,\&\, (p \otimes q)} \quad \rightsquigarrow \quad \dfrac{\dfrac{\overline{p \vdash p} \quad \cancel{p}, \cancel{q}, q \vdash q}{p, p, q, q \vdash p \otimes q} \qquad p, p, q, q \vdash p \otimes q}{p, p, q, q \vdash (p \otimes q) \,\&\, (p \otimes q)} \quad .$$

At this point the computation (search) terminates with *failure*: all of the leaves on the left-hand branch of the derivation have been closed but unconsumed resources, p, q, remain. A similar problem would arise on the right-hand branch, which the lazy strategy never explores.

Thirdly, we give an example involving multiple conclusions. Consider (3), the provable sequent $(p \otimes q) \parr (p \otimes q) \vdash p \otimes q, p \otimes q$. Solving the left-hand branch, we pass the surplus $p \otimes q$, crossed out with $p \cancel{\otimes} q$, to $\Re_1$:

$$\dfrac{\dfrac{\dfrac{\cancel{q}, p \vdash p, p \cancel{\otimes} q \quad q, \cancel{p} \vdash q, p \cancel{\otimes} q}{q, p \vdash p \otimes q, p \otimes q}}{p \otimes q \vdash p \otimes q, p \otimes q} \qquad p \otimes q \vdash \Re_1}{(p \otimes q) \parr (p \otimes q) \vdash p \otimes q, p \otimes q} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\overline{p \vdash p} \quad \overline{q \vdash q}}{q, p \vdash p \otimes q}}{p \otimes q \vdash p \otimes q} \qquad p \otimes q \vdash p \otimes q}{(p \otimes q) \parr (p \otimes q) \vdash p \otimes q, p \otimes q}$$

It now remains to solve $p \otimes q \vdash p \otimes q$, just as for (1).

Looking ahead to the formal account of resource distribution in § 2.2, the important observation is that this process amounts to finding appropriate assignments to Boolean expressions of the form $v_1.v_2$, where each v_i is either a Boolean variable or its negation. We use the axioms to select which formulae are to appear on which branch and this selection is reflected in the assignments to the Boolean variables. From this point of view, the lazy distribution strategy solves a minimal set of equations. To begin to determine the distribution, we must have found a leaf in the proof tree. However, once we have found one such leaf, we can begin to make progress in the calculation of the distribution. So a lazy distribution takes into account just one multiplicative branch at a time.

Eager distribution. Lazy distribution represents one extreme of our class of strategies, *i.e.*, finding just a minimal set of equations to be solved before propagating the resulting assignments to other multiplicative branches. The opposite extreme is one in which all multiplicative branches are explored in parallel, with the co-ordination of resources between branches being handled by a central manager, *i.e.*, the system of Boolean constraints. This means that the entire tree is found before any constraints are solved. This strategy amounts to using an unbounded amount of parallelism to explore all the multiplicative branches before solving only one, maximal, set of equations.

For (1), eager distribution involves attempting to solve all four leaves at once and noticing that the assignments obtained are all mutually compatible. Hence an eager derivation would have the following form:

$$\dfrac{\dfrac{p, \not p, \not q, \not q \vdash p \qquad \not p, \not p, q, \not q \vdash q}{\Im_1 \vdash p \otimes q} \qquad \dfrac{\not p, p, \not q, \not q \vdash p \qquad \not p, \not p, \not q, q \vdash q}{\Im_2 \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} \qquad \rightsquigarrow \ldots .$$

At each leaf, the resources available on the left-hand side are p, p, q, q. Closing each leaf requires exactly one of these. Noting that the choices at all four leaves are mutually compatible, *i.e.*, each formula is needed in exactly one place, we conclude that this derivation can evaluate to

$$\ldots \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{}{p \vdash p} \qquad \dfrac{}{q \vdash q}}{p, q \vdash p \otimes q} \qquad \dfrac{\dfrac{}{p \vdash p} \qquad \dfrac{}{q \vdash q}}{p, q \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} .$$

For (2), we produce the entire tree, only to find that there is no proof. This results in the tree below.

$$\dfrac{\dfrac{\not p, p, \not q, \not q \vdash p \qquad \not p, \not p, q, \not q \vdash q}{p, p, q, q \vdash p \otimes q} \qquad \dfrac{p, \not p, \not q, \not q \vdash p \qquad \not p, \not p, \not q, q \vdash q}{p, p, q, q \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \,\&\, (p \otimes q)}$$

Note that in each of the & branches, the formulae p, q are in excess.

For (3), we also produce the entire tree,

$$\dfrac{\dfrac{\dfrac{p, \not q \vdash p \qquad \not p, q \vdash q}{p, q \vdash p \otimes q}}{p \otimes q \vdash \Im_1} \qquad \dfrac{\dfrac{p, \not q \vdash p \qquad \not p, q \vdash q}{p, q \vdash p \otimes q}}{p \otimes q \vdash \Im_2}}{(p \otimes q) \parr (p \otimes q) \vdash (p \otimes q), (p \otimes q)} .$$

Again, all the allocations to leaves are compatible, so we have a proof.

Intermediate distribution. An *intermediate* distribution strategy is one in which more than one multiplicative branch is explored at once *up to some specified maximum number of branches.* Such a strategy amounts to a *bounded parallelism* — we have a fixed maximum number of search-agents (*e.g.*, processors), each of which can be allocated to a multiplicative branch.

Returning to (1), if we allow *at most two* multiplicative branches, we first generate the derivation

$$\dfrac{\Im_1 \vdash p \otimes q \qquad \Im_2 \vdash p \otimes q}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} .$$

Then, having reached our limit (two branches), we explore each of the subsequent multiplicative branches, above the two first found, sequentially. The main difference here compared with the lazy strategy is the "distance" which the constraints are propagated. As above, we will find that only p is needed on the leftmost leaf, and as a result of this, we know that it will not be used on any other leaf. In addition, a parallel search has identified that the other occurrence of p is needed on the third leftmost leaf. We close these two leaves, and then switch our attention to the two remaining ones. Here $\rightsquigarrow$ denotes the evaluation consisting of calculating the resources required to close both of the leaves.

$$\rightsquigarrow \dfrac{\dfrac{p, \not p, \not q, \not q \vdash p \quad \Im_3 \vdash q}{\Im_1 \vdash p \otimes q} \qquad \dfrac{\not p, p, \not q, \not q \vdash p \quad \Im_4 \vdash q}{\Im_2 \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} \quad \rightsquigarrow \quad \dfrac{\dfrac{p \vdash p \quad q, \not q \vdash q}{p, q \vdash p \otimes q} \qquad \dfrac{p \vdash p \quad \not q, q \vdash q}{p, q \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} .$$

The strategy above amounts to applying a breadth-first search until the limit (two branches) was reached; then searching in a depth-first manner. A slightly different approach would be to reverse this order once it has been determined that the limit would be exceeded were the breadth-first search to continue. In particular, when it is found that at least three multiplicative branches would be needed, we could then calculate the distribution for the two leftmost leaves in parallel, and then for the two rightmost ones. This means that the choice of the two multiplicative branches to be explored is not to use the first two encountered, but the two "closest" ones, *i.e.*, the ones with the least common ancestor closest to the leaves. Following this latter strategy would result in the following derivation:

$$\ldots \rightsquigarrow \dfrac{\dfrac{p, \not p, \not q, \not q \vdash p \quad \not p, \not p, q, \not q \vdash q}{\Im_1 \vdash p \otimes q} \qquad \Im_2 \vdash p \otimes q}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} \quad \rightsquigarrow \ldots \quad .$$

With $\Im_1$ now determined as p, q, we then solve the remaining two leaves in parallel to get

$$\ldots \rightsquigarrow \dfrac{\dfrac{\overline{p \vdash p} \quad \overline{q \vdash q}}{p, q \vdash p \otimes q} \qquad \dfrac{p, \not q \vdash p \quad \not p, q \vdash q}{p, q \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} \quad \rightsquigarrow \quad \dfrac{\dfrac{\overline{p \vdash p} \quad \overline{q \vdash q}}{p, q \vdash p \otimes q} \qquad \dfrac{\overline{p \vdash p} \quad \overline{q \vdash q}}{p, q \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} .$$

For (2), note that we will first produce two additive branches, each of which then splits into two multiplicative ones. Hence we follow essentially the same process as the lazy one, but we search two multiplicative branches in parallel, rather than sequentially, in order to find that this sequent is unprovable. This gives us the tree

$$\dfrac{\dfrac{\not p, p, \not q, \not q \vdash p \quad \not p, \not p, q, \not q \vdash q}{p, p, q, q \vdash p \otimes q} \qquad p, p, q, q \vdash p \otimes q}{p, p, q, q \vdash (p \otimes q) \,\&\, (p \otimes q)} .$$

At this point, as in the lazy case, we halt with failure, as all the leaves have been closed, but there remain the unconsumed resources p, q: for multisets, $\{p, p, q, q\} \neq \{p, q\}$.

For (3), we proceed as follows:

$$\dfrac{\dfrac{p, q \vdash \Im_1}{p \otimes q \vdash \Im_1} \quad \dfrac{p, q \vdash \Im_2}{p \otimes q \vdash \Im_2}}{(p \otimes q) \parr (p \otimes q) \vdash (p \otimes q), (p \otimes q)} \; .$$

At this point, we note we can solve the distribution of the succedent of the endsequent, but that the next application of a rule on each branch will exceed the limit of 2. Hence we then explore each remaining in a lazy manner,

$$\dfrac{\dfrac{\dfrac{p, \cancel{q} \vdash p \quad \Im_1 \vdash q}{p, q \vdash p \otimes q}}{p \otimes q \vdash p \otimes q} \quad \dfrac{\dfrac{p, \cancel{q} \vdash p \quad \Im_2 \vdash q}{p, q \vdash p \otimes q}}{p \otimes q \vdash p \otimes q}}{(p \otimes q) \parr (p \otimes q) \vdash (p \otimes q), (p \otimes q)} \rightsquigarrow \dfrac{\dfrac{\dfrac{\dfrac{}{p \vdash p} \quad \dfrac{}{q \vdash q}}{p, q \vdash p \otimes q}}{p \otimes q \vdash p \otimes q} \quad \dfrac{\dfrac{\dfrac{}{p \vdash p} \quad \dfrac{}{q \vdash q}}{p, q \vdash p \otimes q}}{p \otimes q \vdash p \otimes q}}{(p \otimes q) \parr (p \otimes q) \vdash (p \otimes q), (p \otimes q)} \; .$$

For the alternative intermediate strategy, we have the same tree up to the point where the limit is reached, and then we explore in turn each remaining subtree, which we are able to do entirely eagerly in each case,

$$\dfrac{\dfrac{\dfrac{p, \cancel{q} \vdash p \quad \cancel{p}, q \vdash q}{p, q \vdash \Im_1}}{p \otimes q \vdash \Im_1} \quad \dfrac{p, q \vdash \Im_2}{p \otimes q \vdash \Im_2}}{(p \otimes q) \parr (p \otimes q) \vdash (p \otimes q), (p \otimes q)} \rightsquigarrow \dfrac{\dfrac{\dfrac{\dfrac{}{p \vdash p} \quad \dfrac{}{q \vdash q}}{p, q \vdash p \otimes q}}{p \otimes q \vdash p \otimes q} \quad \dfrac{\dfrac{p, \cancel{q} \vdash p \quad \cancel{p}, q \vdash q}{p, q \vdash p \otimes q}}{p \otimes q \vdash p \otimes q}}{(p \otimes q) \parr (p \otimes q) \vdash (p \otimes q), (p \otimes q)} \; .$$

2.2 A calculus with constraints

The informal account of resource-distribution strategies we have provided in § 2.1 is intuitively satisfying. However, in order to a facilitate theoretical analyses, or systematic mechanical implementations, of resource-distribution we must have a formal account. We provide a sequent calculus for linear logic in which the non-deterministic splitting of resources at multiplicative reductions is explicit.

Definition 1. An *annotated* formula is a formula F together with a Boolean expression e, denoted as $F[e]$.[5] We denote by $exp(F)$ the Boolean expression associated with the annotated formula F. A sequent consisting entirely of annotated formulae is known as a *resource sequent.*

In general, the state of the knowledge of the distribution of the formulae is characterized by the state of knowledge of the Boolean variables, with the distribution of the formula known iff the corresponding Boolean expression has been assigned a value. Hence in addition to the proof tree, we maintain an assignment of the Boolean variables which appear in the proof tree.

[5] We require the following grammar of Boolean expressions: $e ::= x \mid \overline{x} \mid x.e \mid \overline{x}.e$.

Definition 2. Given a multiset of annotated formulae $\Delta = \{F_1[e_1], \ldots F_n[e_n]\}$ and a total assignment I of the Boolean variables in Δ, we define $\Delta[I] = \{F_1[v_1], \ldots F_n[v_n]\}$, where e_i has the value v_i under I. We denote by $\Delta[I]^1$ the multiset of annotated formulae $F[e]$ in $\Delta[I]$ such that e evaluates to 1 under I.

We will often identify an unannotated formula F with the annotated formula $F[1]$ (for instance, in Proposition 8); it will always be possible to disambiguate such annotations from the context.

Definition 3. Let $V = \{x_1, x_2, \ldots x_n\}$ be a set of Boolean variables. Then we denote by $\overline{V}$ the set of Boolean expressions $\{\overline{x_1}, \overline{x_2}, \ldots \overline{x_n}\}$. We denote by $\{e\}^n$ the multiset which contains n copies of the Boolean expression e.

Definition 4. Let $\Gamma = \{F_1[e_1], F_2[e_2], \ldots F_n[e_n]\}$ be a multiset of annotated formulae, and let $\{x_1, x_2, \ldots, x_n\}$ be a set of Boolean variables not occurring in Γ. Then $\Gamma.\{x_1, x_2, \ldots, x_n\} = \{F_1[e_1.x_1], F_2[e_2.x_2], \ldots F_n[e_n.x_n]\}$.

Definition 5. We define the following *sequent calculus with constraints*:

$$\frac{e_1 = e_2 = 1 \ \ \forall e_3 \in \exp(\Gamma \cup \Delta)(e_3 = 0)}{\Gamma, A[e_1] \vdash A[e_2], \Delta}\text{Axiom}$$

$$\frac{e_1 = 1, e_2 = 0 \ \ \forall e_2 \in \exp(\Gamma \cup \Delta)}{\Gamma, \bot[e_1] \vdash \Delta}\bot\text{L} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}\bot\text{R}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1} \vdash \Delta}\mathbf{1}\text{L} \quad \frac{e = 1}{\Gamma, \mathbf{0}[e] \vdash \Delta}\mathbf{0}\text{L} \qquad \frac{e_1 = 1, e_2 = 0 \ \ \forall e_2 \in \exp(\Gamma \cup \Delta)}{\Gamma \vdash \mathbf{1}[e_1], \Delta}\mathbf{1}\text{R} \qquad \frac{e = 1}{\Gamma \vdash \top[e], \Delta}\top\text{R}$$

$$\frac{\Gamma, F_1[x.e], F_2[\overline{x}.e] \vdash \Delta}{\Gamma, (F_1 \,\&\, F_2)[e] \vdash \Delta}\,\&\,\text{L} \qquad \frac{\Gamma \vdash F_1[e], \Delta \quad \Gamma \vdash F_2[e], \Delta}{\Gamma \vdash (F_1 \,\&\, F_2)[e], \Delta}\,\&\,\text{R}$$

$$\frac{\Gamma, F_1[e] \vdash \Delta \quad \Gamma, F_2[e] \vdash \Delta}{\Gamma, (F_1 \oplus F_2)[e] \vdash \Delta}\oplus\text{L} \qquad \frac{\Gamma \vdash F_1[x.e], F_2[\overline{x}.e], \Delta}{\Gamma \vdash (F_1 \oplus F_2)[e], \Delta}\oplus\text{R}$$

$$\frac{\Gamma.V, F_1[e] \vdash \Delta.W \quad \Gamma.\overline{V}, F_2[e] \vdash \Delta.\overline{W}}{\Gamma, (F_1 \parr F_2)[e] \vdash \Delta}\parr\text{L} \qquad \frac{\Gamma \vdash F_1[e], F_2[e], \Delta}{\Gamma \vdash (F_1 \parr F_2)[e], \Delta}\parr\text{R}$$

$$\frac{\Gamma, F_1[e], F_2[e] \vdash \Delta}{\Gamma, (F_1 \otimes F_2)[e] \vdash \Delta}\otimes\text{L} \qquad \frac{\Gamma.V \vdash F_1[e], \Delta.W \quad \Gamma.\overline{V} \vdash F_2[e], \Delta.\overline{W}}{\Gamma \vdash (F_1 \otimes F_2)[e], \Delta}\otimes\text{R}$$

$$\frac{\Gamma.V \vdash F_1[e], \Delta.W \qquad \Gamma.\overline{V}, F_2[e] \vdash \Delta.\overline{W}}{\Gamma, (F_1 \multimap F_2)[e] \vdash \Delta}\multimap\text{L} \qquad \frac{\Gamma, F_1[e] \vdash F_2[e], \Delta}{\Gamma \vdash (F_1 \multimap F_2)[e], \Delta}\multimap\text{R}$$

$$\frac{\Gamma, \vdash F_1[e], \Delta}{\Gamma, (F_1^{\perp})[e] \vdash \Delta}{-}^{\perp}\text{L} \qquad \frac{\Gamma, F_1[e] \vdash \Delta}{\Gamma \vdash (F_1^{\perp})[e], \Delta}{-}^{\perp}\text{R}$$

$$\frac{\Gamma, (F[y/x])[e] \vdash \Delta}{\Gamma, (\exists x F)[e] \vdash \Delta}\exists\text{L} \qquad \frac{\Gamma, (F[t/x])[e] \vdash \Delta}{\Gamma, (\forall x F)[e] \vdash \Delta}\forall\text{L} \qquad \frac{\Gamma \vdash (F[t/x])[e]\Delta}{\Gamma \vdash (\exists x F)[e], \Delta}\exists\text{R} \qquad \frac{\Gamma \vdash (F[y/x])[e], \Delta}{\Gamma \vdash (\forall x F)[e], \Delta}\forall\text{R}$$

$$\frac{\Gamma, F[e] \vdash \Delta}{\Gamma, (!F)[e] \vdash \Delta}!\text{L} \qquad \frac{!\Gamma, F[e] \vdash ?\Delta}{!\Gamma, (?F)[e] \vdash ?\Delta}?\text{L} \qquad \frac{!\Gamma \vdash F[e], ?\Delta}{!\Gamma \vdash (!F)[e], ?\Delta}!\text{R} \qquad \frac{\Gamma \vdash F[e], \Delta}{\Gamma \vdash (?F)[e], \Delta}?\text{R}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, !F \vdash \Delta}W!\text{L} \qquad \frac{\Gamma, !F[e], !F[e] \vdash \Delta}{\Gamma, !F[e] \vdash \Delta}C!\text{L} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?F, \Delta}W?\text{R} \qquad \frac{\Gamma \vdash ?F[e], ?F[e], \Delta}{\Gamma \vdash ?F[e], \Delta}C?\text{R}$$

where the rules $\otimes$R, $\parr$L and $\multimap$L have the side-condition that V and W are disjoint sets of Boolean variables, none of which occur in Γ, Δ, F_1 or F_2, and the rules $\oplus$R and &L have the side condition that x is a Boolean variable which does not occur in Γ, Δ, F_1 or F_2. In $\exists L$ and $\forall R$, $x \notin \text{FV}(\Gamma, \Delta)$.

Definition 2. Given a multiset of annotated formulae $\Delta = \{F_1[e_1], \dots F_n[e_n]\}$ and a total assignment I of the Boolean variables in Δ, we define $\Delta[I] = \{F_1[v_1], \dots F_n[v_n]\}$, where e_i has the value v_i under I. We denote by $\Delta[I]^1$ the multiset of annotated formulae $F[e]$ in $\Delta[I]$ such that e evaluates to 1 under I.

We will often identify an unannotated formula F with the annotated formula $F[1]$ (for instance, in Proposition 8); it will always be possible to disambiguate such annotations from the context.

Definition 3. Let $V = \{x_1, x_2, \dots x_n\}$ be a set of Boolean variables. Then we denote by $\overline{V}$ the set of Boolean expressions $\{\overline{x_1}, \overline{x_2}, \dots \overline{x_n}\}$. We denote by $\{e\}^n$ the multiset which contains n copies of the Boolean expression e.

Definition 4. Let $\Gamma = \{F_1[e_1], F_2[e_2], \dots F_n[e_n]\}$ be a multiset of annotated formulae, and let $\{x_1, x_2, \dots, x_n\}$ be a set of Boolean variables not occurring in Γ. Then $\Gamma.\{x_1, x_2, \dots, x_n\} = \{F_1[e_1.x_1], F_2[e_2.x_2], \dots F_n[e_n.x_n]\}$.

Definition 5. We define the following *sequent calculus with constraints*:

$$\dfrac{e_1 = e_2 = 1 \ \ \forall e_3 \in \exp(\Gamma \cup \Delta)(e_3 = 0)}{\Gamma, A[e_1] \vdash A[e_2], \Delta}\text{Axiom}$$

$$\dfrac{e_1 = 1, e_2 = 0 \ \ \forall e_2 \in \exp(\Gamma \cup \Delta)}{\Gamma, \bot[e_1] \vdash \Delta}\bot\text{L} \qquad \dfrac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}\bot\text{R}$$

$$\dfrac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1} \vdash \Delta}\mathbf{1}\text{L} \qquad \dfrac{e = 1}{\Gamma, \mathbf{0}[e] \vdash \Delta}\mathbf{0}\text{L} \qquad \dfrac{e_1 = 1, e_2 = 0 \ \ \forall e_2 \in \exp(\Gamma \cup \Delta)}{\Gamma \vdash \mathbf{1}[e_1], \Delta}\mathbf{1}\text{R} \qquad \dfrac{e = 1}{\Gamma \vdash \top[e], \Delta}\top\text{R}$$

$$\dfrac{\Gamma, F_1[x.e], F_2[\overline{x}.e] \vdash \Delta}{\Gamma, (F_1 \,\&\, F_2)[e] \vdash \Delta}\,\&\,\text{L} \qquad \dfrac{\Gamma \vdash F_1[e], \Delta \quad \Gamma \vdash F_2[e], \Delta}{\Gamma \vdash (F_1 \,\&\, F_2)[e], \Delta}\,\&\,\text{R}$$

$$\dfrac{\Gamma, F_1[e] \vdash \Delta \quad \Gamma, F_2[e] \vdash \Delta}{\Gamma, (F_1 \oplus F_2)[e] \vdash \Delta}\oplus\text{L} \qquad \dfrac{\Gamma \vdash F_1[x.e], F_2[\overline{x}.e], \Delta}{\Gamma \vdash (F_1 \oplus F_2)[e], \Delta}\oplus\text{R}$$

$$\dfrac{\Gamma.V, F_1[e] \vdash \Delta.W \quad \Gamma.\overline{V}, F_2[e] \vdash \Delta.\overline{W}}{\Gamma, (F_1 \parr F_2)[e] \vdash \Delta}\parr\text{L} \qquad \dfrac{\Gamma \vdash F_1[e], F_2[e], \Delta}{\Gamma \vdash (F_1 \parr F_2)[e], \Delta}\parr\text{R}$$

$$\dfrac{\Gamma, F_1[e], F_2[e] \vdash \Delta}{\Gamma, (F_1 \otimes F_2)[e] \vdash \Delta}\otimes\text{L} \qquad \dfrac{\Gamma.V \vdash F_1[e], \Delta.W \quad \Gamma.\overline{V} \vdash F_2[e], \Delta.\overline{W}}{\Gamma \vdash (F_1 \otimes F_2)[e], \Delta}\otimes\text{R}$$

$$\dfrac{\Gamma.V \vdash F_1[e], \Delta.W \qquad \Gamma.\overline{V}, F_2[e] \vdash \Delta.\overline{W}}{\Gamma, (F_1 \multimap F_2)[e] \vdash \Delta}\multimap\text{L} \qquad \dfrac{\Gamma, F_1[e] \vdash F_2[e], \Delta}{\Gamma \vdash (F_1 \multimap F_2)[e], \Delta}\multimap\text{R}$$

$$\dfrac{\Gamma, \vdash F_1[e], \Delta}{\Gamma, (F_1^{\bot})[e] \vdash \Delta}-^{\bot}\text{L} \qquad \dfrac{\Gamma, F_1[e] \vdash \Delta}{\Gamma \vdash (F_1^{\bot})[e], \Delta}-^{\bot}\text{R}$$

$$\dfrac{\Gamma, (F[y/x])[e] \vdash \Delta}{\Gamma, (\exists x F)[e] \vdash \Delta}\exists\text{L} \qquad \dfrac{\Gamma, (F[t/x])[e] \vdash \Delta}{\Gamma, (\forall x F)[e] \vdash \Delta}\forall\text{L} \qquad \dfrac{\Gamma \vdash (F[t/x])[e]\Delta}{\Gamma \vdash (\exists x F)[e], \Delta}\exists\text{R} \qquad \dfrac{\Gamma \vdash (F[y/x])[e], \Delta}{\Gamma \vdash (\forall x F)[e], \Delta}\forall\text{R}$$

$$\dfrac{\Gamma, F[e] \vdash \Delta}{\Gamma, (!F)[e] \vdash \Delta}!\text{L} \qquad \dfrac{!\Gamma, F[e] \vdash ?\Delta}{!\Gamma, (?F)[e] \vdash ?\Delta}?\text{L} \qquad \dfrac{!\Gamma \vdash F[e], ?\Delta}{!\Gamma \vdash (!F)[e], ?\Delta}!\text{R} \qquad \dfrac{\Gamma \vdash F[e], \Delta}{\Gamma \vdash (?F)[e], \Delta}?\text{R}$$

$$\dfrac{\Gamma \vdash \Delta}{\Gamma, !F \vdash \Delta}W!\text{L} \qquad \dfrac{\Gamma, !F[e], !F[e] \vdash \Delta}{\Gamma, !F[e] \vdash \Delta}C!\text{L} \qquad \dfrac{\Gamma \vdash \Delta}{\Gamma \vdash ?F, \Delta}W?\text{R} \qquad \dfrac{\Gamma \vdash ?F[e], ?F[e], \Delta}{\Gamma \vdash ?F[e], \Delta}C?\text{R}$$

where the rules $\otimes$R, $\parr$L and $\multimap$L have the side-condition that V and W are disjoint sets of Boolean variables, none of which occur in Γ, Δ, F_1 or F_2, and the rules $\oplus$R and $\&$L have the side condition that x is a Boolean variable which does not occur in Γ, Δ, F_1 or F_2. In $\exists L$ and $\forall R$, $x \notin \text{FV}(\Gamma, \Delta)$.

Definition 6. A *resource-derivation* is a tree regulated by the rules of the resource-calculus in which formula of the endsequent is assigned a distinct Bollean variable, together with a partial assignment of the Boolean variables appearing in the derivation. A resource-derivation is *total* if its assignment of the Boolean variables is total. Otherwise, the resource derivation is *partial.* A resource derivation is *closed* if all of the leaves of the proof tree are axioms *i.e.*, one of the rules Axiom, **1**R, **0**L, $\bot$L, $\top$R. A *resource-proof* is a total, closed resource-derivation in which all the Boolean variables in the endsequent and all principal formulae are assigned the value 1.

For notational simplicity, a lack of annotation in any of the rules of the resource-calculus implies that the constraints currently applicable to the formula are not changed. For example, the &R rule (see Definition 5) does not alter the constraints applicable to the formulae in Γ and Δ.

Definition 7. Let R be a total resource-derivation, with proof tree T and Boolean assignment I. The *linear proof tree corresponding to R* is the proof tree obtained by deleting from T all formulae whose Boolean expression evaluates to 0 under I.

There may be many resource-proofs which have the same corresponding linear proof — for example, the linear proof corresponding to a resource-proof of $\Gamma \vdash \Delta$ will be the same as one corresponding to $\Gamma, F[0] \vdash \Delta$.

Resource-proofs are sound and complete with respect to linear sequent calculus. As usual, soundness consists in showing that our global conditions are strong enough to recover proofs from the locally unsound system.

Proposition 8 (soundness of resource-proofs). *Let* $\Gamma \vdash \Delta$ *be a resource-sequent. If* $\Gamma \vdash \Delta$ *has a resource-proof R with Boolean assignment I, then the linear proof tree corresponding to R is a linear proof of* $\Gamma[I]^1 \vdash \Delta[I]^1$.

Proof. By induction on the structure of resource-proofs. We describe just the case for $\otimes$R, the remainder being readily constructible by the reader.

If the last rule of the resource-proof is $\otimes$R, then we have that $\Delta = (F_1 \otimes F_2)[1], \Delta'$, and $\Gamma.V \vdash F_1, \Delta'.W$ and $\Gamma.\overline{V} \vdash F_2, \Delta'.\overline{W}$ both have resource-proofs for some disjoint sets of Boolean variables V and W. By the hypothesis, we have that there are linear proofs of $(\Gamma.V)[I]^1 \vdash (F_1)[I]^1, (\Delta'.W)[I]^1$ and $(\Gamma.\overline{V})[I]^1 \vdash (F_2)[I]^1, (\Delta'.\overline{W})[I]^1$ (recall that I must be an assignment of *all* Boolean variables in the proof), and so there is a linear proof of $(\Gamma.V)[I]^1, (\Gamma.\overline{V})[I]^1 \vdash (F_1 \otimes F_2)[I]^1, (\Delta'.W)[I]^1, (\Delta'.\overline{W})[I]^1$ which is just $\Gamma[I]^1 \vdash (F_1 \otimes F_2)[I]^1, \Delta'[I]^1$, as required.

Note that resource-proofs *do not* allow us to construct a "proof" of $p \otimes q \vdash p \otimes q$ in which the right-hand tensor is reduced first: such a proof is prohibited by the requirement of the $\otimes$L rule that each of p, q and $p \otimes q$ be assigned the same value.

Completeness says that all proofs of all consequences provable in linear sequent calculus can be found. We require the following simple lemma:

Lemma 9. *If $\Gamma \vdash \Delta$ has a closed resource-derivation, then $\Gamma, F[0] \vdash \Delta$ and $\Gamma \vdash F[0], \Delta$ also have closed resource-derivations.*

Proof. Immediate from the definition of (closed) resource-derivations.

Proposition 10 (completeness of resource-proofs). *If $\Gamma \vdash \Delta$ has a proof Φ in the linear sequent calculus, then there are disjoint sets of Boolean variables V and W such that $\Gamma.V \vdash \Delta.W$ has a resource-proof R and the linear proof tree corresponding to R is Φ.*

Proof. By induction on the structure of proofs in linear sequent calculus. We describe just the case for ⊗R, the remainder being readily constructible by the reader.

If the last rule in Φ is ⊗R, then we have that $\Gamma = \Gamma_1, \Gamma_2$, $\Delta = F_1 \otimes F_2, \Delta_1, \Delta_2$ such that $\Gamma_1 \vdash F_1, \Delta_1$ and $\Gamma_2 \vdash F_2, \Delta_2$ both have proofs in the linear sequent calculus. Hence by the hypothesis there are disjoint sets of Boolean variables V_1, V_2, W_1, W_2 such that $\Gamma_1.V_1 \vdash F_1, \Delta_1.W_1$ and $\Gamma_2.V_2 \vdash F_2, \Delta_2.W_2$ have resource-proofs, (and moreover the linear proofs corresponding to each resource-proof is the appropriate subproof of Φ) and so by Lemma 9, there are closed resource-derivations of $\Gamma_1.V_1, \Gamma_2.\{0\}^n \vdash F_1, \Delta_1.W_1, \Delta_2.\{0\}^n$ and $\Gamma_1.\{0\}^n, \Gamma_2.V_2 \vdash F_2, \Delta_1.\{0\}^n, \Delta_2.W_2$. Hence there are new disjoint sets of Boolean variables (*i.e.*, not occurring anywhere in the above two resource sequents) V and W and a total assignment$I$ of $V \cup W$ such that $(\Gamma_1.V_1, \Gamma_2.V_2).V \vdash F_1, (\Delta_1.W_1, \Delta_2.W_2).W$ and $(\Gamma_1.V_1, \Gamma_2.V_2).\overline{V} \vdash F_2, (\Delta_1.W_1, \Delta_2.W_2).\overline{W}$ have resource-proofs, and so there is a resource-proof of $\Gamma_1.V_1, \Gamma_2.V_2 \vdash (F_1 \otimes F_2)[1], \Delta_1.W_1, \Delta_2.W_2$, *i.e.*, $\Gamma.V' \vdash (F_1 \otimes F_2)[1], \Delta.W'$ for some disjoint sets of Boolean variables V' and W', and clearly the linear proof corresponding to this resource-proof is Φ.

2.3 Strategies via the calculus with constraints

Our intended class of resource-distribution strategies can be described formally via the calculus with constraints, resource-derivations and resource-proofs. Our discussion of the lazy, intermediate and eager strategies will also serve to provide examples of resource-derivations and resource-proofs.

As mentioned above, resource proofs are intended to be independent of a particular strategy, but to contain an explicit specification of the distributive constraints. Hence we consider an *n-strategy* to be one which solves the equations from at most n multiplicative branches at a time. Hence, the lazy strategy is a 1-strategy, intermediate strategies are n-strategies for $2 \leq n \leq k$ for some finite k, and the eager strategy is an ω-strategy.

Lazy distribution. In terms of the calculus introduced above, the lazy strategy solves one multiplicative branch's worth of Boolean constraints at a time, and propagates the solution (as well as any remaining constraints) to the next multiplicative branch. This may be thought of as a pessimistic strategy, in that as only a minimal set of constraints is solved, if the derivation turns out to be unsuccessful, then only a minimal amount of work has been done.

Eager distribution. The eager distribution is an ω-strategy, in that an unbounded number of equations can be solved, and so we wait until all leaves are closed before attempting to solve the set of constraints. Then there is one (large) set of constraints to be solved. This may be thought of as an optimistic strategy, in that if one of the branches leads to failure, then the work done on evaluating all the other branches in parallel has been wasted. In general we do not have sufficient information to solve any of these constraints until a leaf is reached; however, we shall assume that an eager strategy whenever possible solves constraints as soon as they arise (see Example (3) below).[6]

Intermediate distribution. Intermediate strategies are n-strategies, where $n \geq 2$. As discussed above, the precise way in which a proof which involves $n+1$ multiplicative branches may be either an eager search for the first n such branches (proceeding from the root), and then lazy searches from then on (effectively performing n lazy searches in parallel), or to "switch" the eager version to a place further from the root (effectively performing a number of lazy searches, one of which is a n-way eager search).

As in general it is not possible to predict in advance where the leaves in a proof will be found, it would seem intuitively reasonable in a bottom-up system to adopt the policy that the eager behaviour occurs towards the root, and once the bound of n is reached, n multiplicative branches are chosen to be explored in a lazy manner. However, this may result in sub-optimal behaviour, as the "locality" of the constraints is lost (see the example (3) below). The alternative would require extra analysis, as initially search would proceed as above, but once the limit is reached, it is necessary to re-assign the n searchers to work on the sub-branches of a particular multiplicative branch in some appropriate way.

For example, consider a 2-strategy with the sequent $p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)$. It is easy to see that as there are 3 occurrences of $\otimes$ in the formula in the succedent, there will be (at least) 4 multiplicative branches in the ensuing proof. Hence it would be reasonable to use the lazy manner for the first occurrence of $\otimes$, and then solve each generated branch in an eager manner.

Examples. We return to our earlier examples. For (1), resource-proof of this sequent is of the form

$$\dfrac{\dfrac{P_1 \qquad P_2}{p[x_1], p[x_2], q[x_3], q[x_4] \vdash p \otimes q} \qquad \dfrac{P_3 \qquad P_4}{p[\overline{x_1}], p[\overline{x_2}], q[\overline{x_3}], q[\overline{x_4}] \vdash p \otimes q}}{p, p, q, q \vdash (p \otimes q) \otimes (p \otimes q)} ,$$

where the leaves are as follows:

$P_1 : p[x_1.y_1], p[x_2.y_2], q[x_3.y_3], q[x_4.y_4] \vdash p$ $\qquad$ $P_3 : p[\overline{x_1}.z_1], p[\overline{x_2}.z_2], q[\overline{x_3}.z_3], q[\overline{x_4}.z_4] \vdash p$

$P_2 : p[x_1.\overline{y_1}], p[x_2.\overline{y_2}], q[x_3.\overline{y_3}], q[x_4.\overline{y_4}] \vdash q$ $\qquad$ $P_4 : p[\overline{x_1}.\overline{z_1}], p[\overline{x_2}.\overline{z_2}], q[\overline{x_3}.\overline{z_3}], q[\overline{x_4}.\overline{z_4}] \vdash q$

The lazy strategy yields the following sequence of constraints and solutions:

Leaf	**Constraints added**	**Solutions**
P_1	$x_1.y_1 = 1, x_2.y_2 = 0, x_3.y_3 = 0, x_4.y_4 = 0$	$x_1 = 1, y_1 = 1$
P_2	$x_2.\overline{y_2} = 0, x_3.\overline{y_3} = 1, x_4.\overline{y_4} = 0$	$x_3 = 1, y_3 = 0, x_2 = 0, x_4 = 0$
P_3	$z_2 = 1, z_4 = 0$	$z_2 = 1, z_4 = 0$
P_4	$\overline{x_4}.\overline{z_4} = 1$	

[6] It may also be useful to check that the current constraints have a solution (as distinct from actually solving them), as happens in many constraint logic programming languages.

which gives us the overall solution

$$x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, \quad y_1 = 1, y_2 = 0, y_3 = 0, y_4 = 0, \quad z_1 = 0, z_2 = 1, z_3 = 0, z_4 = 0$$

where y_2, y_4, z_1 and z_3 have been arbitrarily assigned the value 0. Note that we can conclude from the equations $x_4.y_4 = 0$ and $x_4.\overline{y_4} = 0$ that x_4 must be 0, and similarly for x_2.

The eager strategy collects the entire set of equations below, and then solves it to produce the same overall solution.

$$\begin{array}{ll} x_1.y_1 = 1, x_2.y_2 = 0, x_3.y_3 = 0, x_4.y_4 = 0 & \overline{x_1}.z_1 = 0, \overline{x_2}.z_2 = 1, \overline{x_3}.z_3 = 0, \overline{x_4}.z_4 = 0 \\ x_1.\overline{y_1} = 0, x_2.\overline{y_2} = 0, x_3.\overline{y_3} = 1, x_4.\overline{y_4} = 0 & \overline{x_1}.\overline{z_1} = 0, \overline{x_2}.\overline{z_2} = 0, \overline{x_3}.\overline{z_3} = 0, \overline{x_4}.\overline{z_4} = 1 \end{array}$$

The first variant of the intermediate strategy first solves the equations for P_1 and P_3 in parallel, and then those for P_2 and P_4:

Leaf	**Constraints added**	**Solutions**
P_1, P_3	$x_1.y_1 = 1, x_2.y_2 = 0, x_3.y_3 = 0, x_4.y_4 = 0$	$x_1 = 1, y_1 = 1$
	$\overline{x_1}.z_1 = 0, \overline{x_2}.z_2 = 1, \overline{x_3}.z_3 = 0, \overline{x_4}.z_4 = 0$	$x_2 = 0, z_2 = 1$
P_2, P_4	$x_2.\overline{y_2} = 0, x_3.\overline{y_3} = 1, x_4.\overline{y_4} = 0$	$x_3 = 1, y_3 = 0$
	$\overline{x_3}.\overline{z_3} = 0, \overline{x_4}.\overline{z_4} = 1$	$x_4 = 0, z_4 = 0$

The second variant of the intermediate strategy first solves the equations for P_1 and P_2 in parallel, and then those for P_3 and P_4:

Leaf	**Constraints added**	**Solutions**
P_1, P_2	$x_1.y_1 = 1, x_2.y_2 = 0, x_3.y_3 = 0, x_4.y_4 = 0$	$x_1 = 1, y_1 = 1$
	$x_1.\overline{y_1} = 0, x_2.\overline{y_2} = 0, x_3.\overline{y_3} = 1, x_4.\overline{y_4} = 0$	$x_3 = 1, y_3 = 0, x_4 = 0, x_2 = 1$
P_3, P_4	$z_2 = 1, z_4 = 0$	$z_2 = 1, z_4 = 0$
	$z_2 = 1, \overline{z_4} = 1$	

Note that solving the equations for P_1 and P_2 in parallel generates more of the solution than solving those for P_1 and P_3 in parallel.

For (2), we note that the lazy strategy finds the leaf $p[x_1], p[x_2], q[x_3], q[x_4] \vdash p[1]$, and finds the solution $x_1 = 1, x_2 = x_3 = x_4 = 0$, and then attempts to close the leaf $p[\overline{x_1}], p[\overline{x_2}], q[\overline{x_3}], q[\overline{x_4}] \vdash q[1]$. In doing so, it then attempts to solve the equations $\overline{x_1} = 0, \overline{x_2} = 0, \overline{x_3} = 1, \overline{x_4} = 0$, which cannot be done as we have that $x_1 = 1, x_2 = x_3 = x_4 = 0$. Hence we find that there is no resource-proof (note that similar behaviour occurs for the other solution $x_1 = 0, x_2 = 1, x_3 = x_4 = 0$ for the first leaf).

The eager strategy is clearly less efficient *in this case*, as it will generate the full set of equations below before finding that there is no solution:

$$\begin{array}{ll} x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0 & y_1 = 1, y_2 = 0, y_3 = 0, y_4 = 0 \\ \overline{x_1} = 0, \overline{x_2} = 0, \overline{x_3} = 2, \overline{x_4} = 0 & \overline{z_1} = 0, \overline{z_2} = 0, \overline{z_3} = 2, \overline{z_4} = 0 \end{array}$$

The first variant of the intermediate strategy will find solutions for the first and third leaves, before finding that these are incompatible for the second and fourth, and so will perform a similar amount of work to the eager one. The other variant will be similar to the lazy one, exploring the first two leaves in parallel before finding that there is no solution.

For (3), the resource-proof is of the form

$$\dfrac{\dfrac{\dfrac{P_1 \qquad P_2}{p[1], q[1] \vdash (p \otimes q)[x_1], (p \otimes q)[x_2]}}{(p \otimes q)[1] \vdash (p \otimes q)[x_1], (p \otimes q)[x_2]} \qquad \dfrac{\dfrac{P_3 \qquad P_4}{p[1], q[1] \vdash (p \otimes q)[\overline{x_1}], (p \otimes q)[\overline{x_2}]}}{(p \otimes q)[1] \vdash (p \otimes q)[\overline{x_1}], (p \otimes q)[\overline{x_2}]}}{(p \otimes q) \mathbin{⅋} (p \otimes q) \vdash (p \otimes q), (p \otimes q)},$$

where the "leaves" are as follows:

$$\begin{array}{ll} P_1\colon p[y_1], q[y_2] \vdash p[x_1], (p \otimes q)[x_2.y_3] & P_3\colon p[z_1], q[z_2] \vdash (p \otimes q)[\overline{x_1}.z_3], p[\overline{x_2}] \\ P_2\colon p[\overline{y_1}], q[\overline{y_2}] \vdash q[x_1], (p \otimes q)[x_2.\overline{y_3}] & P_4\colon p[\overline{z_1}], q[\overline{z_2}] \vdash (p \otimes q)[\overline{x_1}.\overline{z_3}], q[\overline{x_2}]. \end{array}$$

Note that we know from the application of the rules that $x_1 = 1$ and $x_2 = 0$; however we do not explicitly use this information above, as the different strategies will discover this in different ways. For example, the lazy strategy will discover that x_1 must be 1 before it is found that x_2 must be 0.

The lazy strategy yields the following sequence of constraints and solutions:

Leaf	**Constraints added**	**Solutions**
P_1	$x_1 = 1, y_1 = 1, y_2 = 0, x_2.y_3 = 0$	$x_1 = 1, y_1 = 1, y_2 = 0$
P_2	$x_1 = 1, x_2.\overline{y_3} = 0$	$x_2 = 0$
P_3	$\overline{x_2} = 1, z_1 = 1, z_2 = 0$	$z_1 = 1, z_2 = 0$
P_4	$\overline{z_1} = 0, \overline{z_2} = 1$	

The eager strategy will produce the full set of equations, given below.

$$x_1 = 1, x_2 = 0,\ y_1 = 1, y_2 = 0, x_2.y_3 = 0,\ \overline{y_1} = 0, \overline{y_2} = 1, x_2.\overline{y_3} = 0,$$
$$z_1 = 1, z_2 = 0, \overline{x_2}.z_3 = 0,\ \overline{z_1} = 0, \overline{z_2} = 1, \overline{x_1} = 0, \overline{x_2}.\overline{z_3} = 0$$

The intermediate strategies both find that $x_1 = 1$ and $x_2 = 0$ before the leaves are reached. The first version produces the constraints $y_1 = 1, y_2 = 0, z_1 = 1, z_2 = 0$, and then the constraints $\overline{y_1} = 0, \overline{y_2} = 1, \overline{z_1} = 1, \overline{z_2} = 0$, which merely require verification rather than solving.

The second version produces the constraints $y_1 = 1, y_2 = 0, \overline{y_1} = 0, \overline{y_2} = 1$, and then the constraints $z_1 = 1, z_2 = 0, \overline{z_1} = 0, \overline{z_2} = 1$, which clearly give the same solution.

Other strategies. Obviously the calculus is able to describe other, perhaps more *ad hoc*, strategies. For example, consider the rule application

$$\frac{\Gamma.V \vdash F_1, \Delta.W \quad \Gamma.\overline{V} \vdash F_1, \Delta.\overline{W}}{\Gamma \vdash F_1 \otimes F_2, \Delta}\ .$$

If one of the premisses is an axiom, then clearly it would be reasonable to solve the equations for the axiom (if possible), and then propagate the solutions to the other branch.[7] It may also be reasonable to use an adaptive strategy, such as using an eager strategy to provide maximal information about future branches, but then reverting to a lazy strategy when the probability of failure, early detection of which is desired, is higher.

2.4 Selection

We have not explicitly mentioned the *selective non-determinism* in using the above rules, *i.e.*, the choice of formula in the &L and ⊕R rules. Here we use the same mechanism as for the binary multiplicative rules, but rather than labelling the occurrences of each formula with a complementary expression on each branch, we have two copies of the appropriate formula on the same branch, each with a complementary expression. Hence, there will be exactly one formula selected at some point higher up in the proof.

For example, consider the sequent $p, p \,\&\, q \vdash p \otimes q$. Here we get the resource-derivation

$$\frac{\dfrac{p[y_1], p[x.y_2], q[\overline{x}.y_3] \vdash p \quad p[\overline{y_1}], p[x.\overline{y_2}], q[\overline{x}.\overline{y_3}] \vdash q}{p, p[x], q[\overline{x}] \vdash p \otimes q}}{p, p \,\&\, q \vdash p \otimes q}\ \&L,$$

[7] Unlike the lazy strategy, a particular *order* of evaluation is involved here.

which gives the solution $y_1 = 1, x = 0, y_2 = 0, y_3 = 0$. Note that we cannot solve the equations $y_1 = 0, x.y_2 = 1, \overline{x}.y_3 = 0, \overline{y_1} = 1, x.\overline{y_2} = 0, \overline{x}.\overline{y_3} = 1$.

It should be noted that this method of dealing with the binary multiplicative rules is essentially making such rules additive, and recovering the appropriate provability relation by means of the Boolean constraints.

We can reduce the need for explicit branching in the proof tree if we allow a more complex system of Boolean constraints to be used. The system used above is implicitly existentially quantified; we may consider a constraint such as $x_2.x_1 = 1$ as an abbreviation for $\exists x_1, x_2\ (x_2.x_1 = 1)$, and so all we need to do is to find a single solution to the Boolean constraints (although clearly different solutions will generate different proofs). However, if we allow universally quantified Boolean constraints as well, then we may borrow a technique from proof-nets [4, 5], and eliminate the need for explicit additive branching.

For example, consider the provable sequent $p \mathbin{\&} q \vdash p \mathbin{\&} q$. A resource proof of this sequent is given below.

$$\dfrac{\dfrac{p[x], q[\overline{x}] \vdash p[1]}{(p \mathbin{\&} q)[1] \vdash p[1]}\ \&L \qquad \dfrac{p[w], q[\overline{w}] \vdash q[1]}{(p \mathbin{\&} q)[1] \vdash q[1]}\ \&L}{(p \mathbin{\&} q)[1] \vdash (p \mathbin{\&} q)[1]}\ \&R.$$

We find that the system of equations $x = 1, \overline{x} = 0, w = 0, \overline{w} = 1$ has the solution $x = 1, w = 0$, so we have a resource proof. In the "branchless" version, we would have the derivation

$$\dfrac{\dfrac{p[x], q[\overline{x}] \vdash p[y], q[\overline{y}]}{(p \mathbin{\&} q)[1] \vdash p[y], q[\overline{y}]}\ \&L}{(p \mathbin{\&} q)[1] \vdash (p \mathbin{\&} q)[1]}\ \&R$$

and we would then have to solve the constraint $\forall y \in \{0, 1\}\ (\exists x\ (y = x))$, which is clearly satisfiable. Hence the sequent is provable.

3 Applications to linear logic programming

The existing linear logic programming languages may be divided into those which are implemented sequentially (Lygon, LO, Lolli, Forum, LinLog) and those which are intended as concurrent languages (LC, ACL). The former languages all use the lazy strategy to distribute formulae across multiplicative branches, whereas the latter involve a genuinely concurrent implementation of the multiplicatives. Hence we may use a lazy strategy (such as the one used in Lygon) to model state changes, or an eager strategy for producer-consumer problems (or other such multi-threaded computations). Our framework characterizes all of the strategies used in the implementation of these languages.

A strategy embodying a lazy-eager duality would be useful in the area of transaction-processing, where for efficiency reasons it is often desirable to interleave the execution of transactions as much as possible, whilst maintaining the property that, if necessary, it is possible to reconstruct a total ordering of the execution of the transactions with the same overall effect as the interleaved execution (this property is often called *serializability*). In our terms, this corresponds to executing the transactions with the eager strategy (or perhaps with an intermediate one), knowing that a lazy execution of the same transactions is possible. In other words, the serializability property is an immediate result for any transaction processing system expressible in this framework.

Paradigms such as chemical programming (which, in essence, result from a concurrent interpretation of $\parr$) are easily modelled by the use of the eager strategy. Furthermore, it is interesting to note that as transitions between states in this paradigm are often modelled by linear formulae of the form $A_1 \parr \ldots \parr A_n \multimap B_1 \parr \ldots \parr B_m$, together with an initial state $C_n \parr \ldots \parr C_k$, there is generally only one non-trivial multiplicative branch in a proof, and so the lazy, intermediate and eager strategies all coincide. Our framework clearly also provides for a generalization of this paradigm to formulae of the form $(G_1 \multimap D_1) \parr \ldots \parr (G_n \multimap D_n)$, in which the distribution problems are not trivial [10].

An appropriate implementation of resource-proofs would be to use a finite-domain constraint logic programming language in order to provide an appropriate mix of proof-search techniques and Boolean constraint solving methods.

Concerning complexity, we conjecture that it will be possible to exploit the essential restriction of linear logic programming languages to *hereditary Harrop formulae* and the identification of *paths* [10] (a kind of proof-object [11], related to proof-nets [4, 5]) to partition the sets of Boolean equations obtained into smaller, independently solvable collections. Indeed, the techniques presented herein amount to a formal account of the use of paths to describe and execute strategies.

Acknowledgements. We thank Michael Winikoff and the anonymous referees for their comments on this work. Partial support from the British Council, the UK EPSRC and the RMIT Faculty of Applied Science is gratefully acknowledged.

References

1. J.-M. Andreoli. Logic Programming with Focusing Proofs in Linear Logic. *J. Logic Computat.* 2(3), 1992.
2. D. Galmiche and G. Perrier. On proof normalization in Linear Logic, *Theoret. Comp. Sci.* 135, 67-110, 1994.
3. G. Gentzen. Untersuchungen über das logische Schliessen. *Math. Zeit.* 39, 176-210, 405-431, 1934.
4. J.-Y. Girard. Linear Logic. *Theoret. Comp. Sci.* 50, 1-102, 1987.
5. J.-Y. Girard. Proof-nets for Additives. Manuscript, 1994.
6. J. Harland, D. Pym and M. Winikoff. Programming in Lygon: An Overview. Proc. AMAST'96, M. Wirsing and M. Nivat, editors, LNCS 1101, 391-405, July, 1996.
7. J. Hodas and D. Miller. Logic Programming in a Fragment of Intuitionistic Linear Logic. *Inform. and Computat.* 110:2:327-365, 1994.
8. P. Lincoln and N. Shankar. Proof Search in First-order Linear Logic and Other Cut-free Proc. of the IEEE Symposium on *Logic in Computer Science* 281-292, Paris, June, 1994.
9. D. Miller. A multiple-conclusion meta-logic. Proc. of the IEEE Symposium on *Logic in Computer Science* 272–281, Paris, June, 1994.
10. D. Pym and J. Harland. A Uniform Proof-theoretic Investigation of Linear Logic Programming. *J. Logic Computat.* 4:2:175-207, 1994.
11. D. Pym and L. Wallen. Logic Programming via Proof-valued Computations. In: ALPUK92, Proc. 4th U.K. Conference on Logic Programming, K. Broda (editor), 253-263. Springer-Verlag, Workshops in Computing Series, 1992.
12. T. Tammet. Proof Search Strategies in Linear Logic. *J. Automat. Reas.* 12:273-304, 1994.
13. A. Troelstra. Lectures on Linear Logic. CSLI Lecture Notes No. 29, 1992.
14. M. Winikoff and J. Harland. Implementing the Linear Logic Programming Language Lygon. Proc. ILPS'95 66-80, J. Lloyd (ed.), MIT Press, 1995.

This article was processed using the LaTeX macro package with LLNCS style