Sharing Resource-Sensitive Knowledge using Combinator Logics*

Marcelo Finger**
Instituto de Matemática e Estatística
Universidade de São Paulo
Rua do Matão, 1010
05315-970 São Paulo, Brazil
Phone: +55 -11 818 6310
Email: mfinger@ime.usp.br

Wamberto Vasconcelos***
Institut für Informatik
Universität Zürich
Winterthurerstr., 190
CH 8057 Zürich, Switzerland
Phone: +41 -1 635 6757
Email: wvasconcelos@acm.org

Abstract. Research on ontologies has been pursued as a solution to the difficult problem of knowledge sharing. Ontologies consist of a domain description which suits the needs of all systems to be integrated. Any agreed ontology, however, is not the end of the problems involved in knowledge sharing since how we represent knowledge is intimately linked to the inferences we expect to perform with it. Knowledge sharing cannot ignore the similarities and differences between the inference engines participating in the information exchange. We illustrate this issue via a case study on resource-sensitive knowledge-based systems and we show how these can efficiently share their knowledge using combinator logics.
Keywords. Knowledge sharing; resource-sensitive logics.

1 Introduction

One of the benefits of formally representing knowledge lies in its potential to be *shared*. Technologies for computer interconnection, now relatively cheap and widespread, make it possible for knowledge bases and inference engines developed in different locations to interact in order to solve together more complex problems than those they were originally intended for individually [6].

In this paper we investigate the problem of knowledge sharing among *resource-sensitive* systems (also called substructural logic systems [9]). The class of substructural logics, encompassing, for instance, intuitionistic logics [8], relevance logics [2] and linear logics [15], employ in their inferences *structural rules* which take into account the structure of premises in a deduction [9]. Substructural logics differ from each other by virtue of the structural rules allowed in their proofs: the set of structural rules permitted in one logic may be extended or reduced thus giving rise to other logics. A number of real-life problems can be naturally represented and elegantly solved via resource-sensitive inferencing. Useful and computationally efficient reasoning systems have been developed employing substructural logics (for instance, [19] and [25]).

* Work sponsored by the Consortium British Council/CAPES (Brazil), Grant no. 070/98.
** Partially sponsored by CNPq (Brazilian Research Council), Grant no. 300597/95-9.
*** On a Post-Doctoral leave of absence from Departamento de Estatística e Computação, Universidade Estadual do Ceará, Ceará, Brazil, sponsored by CNPq, Grant no. 201340/91-7.

The organisation of this paper is as follows. Initially, we list some of the issues related to knowledge sharing and give a perspective on work carried out. In Section 2 we present *combinator logics* as our chosen formalism for defining a knowledge-sharing framework among systems based in resource-sensitive logics; we also justify our choice and compare it with alternative approaches. In Section 3 we show how combinator logics can be used to foster integration among knowledge-based systems. Finally in Section 4 we comment on our work, draw conclusions and give directions for future research.

1.1 Some Issues on Knowledge Sharing

Research on knowledge sharing mostly concentrates on the mappings between different domain-specific notations while making the assumption that the inference mechanisms of each system are compatible. Sometimes this assumption is explicit: to share knowledge, each system must translate its knowledge into a standard system of inference [17, 21, 23]. At other times the assumption is implicit: a standard knowledge representation language is provided but it is mostly left to users of the notation to choose compatible forms of inference [18, 24].

Without this assumption it would be very difficult to guarantee that the meaning of knowledge expressed in one system is preserved when used by another system. This meaning preservation would, in principle, require us to demonstrate that the models of the world permitted by a system supplying its knowledge include all the models of the world permitted by the system receiving that knowledge. If we allow donors of knowledge also to be recipients then the theoretical constraint becomes even stronger: the models permitted by all systems must coincide precisely. This raises a major practical problem because differences in models of the world often show up in the distinct forms of inference used to derive consequences from the knowledge we represent.

A definitive solution for this problem would require that all knowledge-based systems must share the same models of the world. This solution, however, is not practical, for it is far too restrictive, ruling out many interesting and useful forms of *partial* knowledge sharing, as shown in [7]. Partially shared knowledge involves loss of information but in many cases the loss may be detected, assessed and made tolerable.

In [5] a formal method to share components of different logics is described. That proposal is based on the concept of *institutions* [16] to represent logics and employs special mappings among their components as a means to translate logics and to share their model theory, deductive system, axioms and theorems. However, we would like to achieve knowledge sharing in a more opportunistic form: rather than assuming we have a formal description of each system, our scenario will be a more realistic one if we assume that very little is known of all participating systems. A knowledge-based system (henceforth called "KBS", for short) KBS_1 may want to *try* to use another system KBS_2 by posing it queries and analysing its answers. In this work we describe a computationally efficient approach to perform the analysis of answers.

Knowledge can be shared in many different ways. We describe here some possible scenarios for knowledge sharing. Let there be two knowledge based systems, $KBS_1 = (IE_1, KB_1)$ and $KBS_2 = (IE_2, KB_2)$, where KB_i is the system's respective knowledge base and IE_i its inference engine [14]. Let us suppose that

KBS_1, the *recipient*, poses query Q to KBS_2, the *donor*. KBS_1 must then supply the triple $\langle Q, IE, KB\rangle$, informing the donor system of the knowledge sharing details: KBS_2 is to provide an answer to Q employing KB and IE, that is, whether $KB \vdash_{IE} Q$ holds. We can then have:

- Donor system as a *surrogate* – this situation arises when KBS_1 supplies the triple $\langle Q, IE_1, KB_1\rangle$ to KBS_2 and KBS_2 employs IE_1 on KB_1 in order to solve Q. KBS_2 is simply used as a surrogate computational device emulating KBS_1.
- Donor system as a *partial surrogate* – this happens when KBS_1 supplies $\langle Q, IE_1, KB_2\rangle$ to KBS_2 and KBS_2 employs IE_1 on its own knowledge base KB_2 in order to solve Q.
- Donor system as a *partial oracle* – this situation happens when KBS_1 supplies $\langle Q, IE_2, KB_1\rangle$ to KBS_2 and KBS_2 employs its own inference engine IE_2 on the given knowledge base KB_1 in order to solve Q.
- Donor system as an *oracle* – this happens when KBS_1 supplies $\langle Q, IE_2, KB_2\rangle$ to KBS_2 and KBS_2 employs its own inference engine IE_2 on its own knowledge base KB_2 in order to solve Q.

In the oracle case, the answers provided by the donor system KBS_2 are to be considered as always right and used as such: the inferences carried out within the oracle are not regarded as important. If, on the other hand, the inferences are to be taken into account, that is, the recipient system is not only interested in the final answers but also in how they have been obtained, then we say that KBS_2 is a *surrogate system* [7]. Variants of the scenarios above are possible, such as KBS_1 providing $\langle Q, IE_1 \cup IE_2, KB_2\rangle$ or $\langle Q, IE_1, KB_1 \cup KB_2\rangle$ in which case the recipient system's capabilities are extended. By posing restrictions or extending the KBS's inferential power, a number of interesting and practical situations arise.

1.2 The Problem: Accepting Inferences

Each system has its own inference engine. This poses a difficulty when we are dealing with surrogacy situations, for it may happen that the recipient system is supplied with answers whose deductions would not be permitted were its own inference engine employed. How can this problem be circumvented?

We suggest that the solution is to let the donor system behave as it wants, and to let the recipient decide whether it accepts or not the answers to a given query.

To enable a KBS to reject an answer from a remote system, the latter must provide not only the answer for a given query, but also describe how that inference was achieved. This means that the donor must provide, together with its answer, a description of the inference steps that led to that conclusion. The receiver will inspect that inference and decide whether to accept or reject it.

This poses an extra overhead which may lead to unacceptable inefficiencies. Proofs are normally large objects, usually of orders of magnitude larger than the answer they generate. Procedures to examine a proof and check for properties in it will naturally reflect the size and complexity of the objects involved.

In the following we show how this problem can be avoided by sending a much more compact representation of the "important aspects" of the inferences. We

show that this can be done efficiently for the class of resource-sensitive logics known as *substructural logics*.

2 Substructural and Combinator Logics

2.1 Substructural Logics

Substructural logics are a family of logics which differ from each other by the set of *structural rules* that each logic in the family accepts. These structural rules determine how resources are dealt with by each logic, and therefore the whole family is also known as *resource sensitive logics*.

For the purpose of this work, we will be working with a fragment of the logic defined by the connectives $\otimes$ (multiplicative conjunction), $\rightarrow$ (right implication) and $\leftarrow$ (left implication) (in the absence of commutativity, $\rightarrow$ and $\leftarrow$ are not equivalent). Each logic in the family will obey the following connective rules, depicted as Gentzen sequent rules:

$$\frac{}{\varphi \vdash \varphi}\ (\text{Axiom}) \qquad \frac{\Gamma \vdash \varphi \quad \Delta \vdash \chi}{\Gamma, \Delta \vdash \varphi \otimes \chi}\ (\vdash \otimes) \qquad \frac{\Gamma[\varphi, \psi] \vdash \chi}{\Gamma[\varphi \otimes \psi] \vdash \chi}\ (\otimes \vdash) \qquad \frac{\Gamma, \varphi \vdash \chi}{\Gamma \vdash \varphi \rightarrow \chi}\ (\vdash \rightarrow)$$

$$\frac{\Gamma \vdash \varphi \quad \Delta[\psi] \vdash \chi}{\Delta[\varphi \rightarrow \psi, \Gamma] \vdash \chi}\ (\rightarrow \vdash) \qquad \frac{\varphi, \Gamma \vdash \chi}{\Gamma \vdash \chi \leftarrow \varphi}\ (\vdash \leftarrow) \qquad \frac{\Gamma \vdash \varphi \quad \Delta[\psi] \vdash \chi}{\Delta[\Gamma, \psi \leftarrow \varphi] \vdash \chi}\ (\leftarrow \vdash)$$

Because we are not assuming a priori structural rules, the antecedent of a sequent is a binary tree, with formulae at its leaves and ',' at the internal nodes. Antecedents are by default left-associative, so Φ_1, Φ_2, Φ_3 actually represents $((\Phi_1, \Phi_2), \Phi_3)$. The consequent of a sequent is always a single formula. By $\Gamma[\varphi]$ we mean a specific occurrence of φ in the structure Γ, and a corresponding $\Gamma[\Psi]$ in the lower part of a rule means the substitution of that occurrence of φ by the structure Ψ in Γ. What distinguishes one substructural logic from another are the structural rules that are allowed in its inferences. The most common structural rules are:

Left-associativity	*Right-associativity*	*Commutativity*	*Contraction*	*Thinning*
$\dfrac{\Gamma, (\Phi, \Psi) \vdash \chi}{(\Gamma, \Phi), \Psi \vdash \chi}$	$\dfrac{(\Gamma, \Phi), \Psi \vdash \chi}{\Gamma, (\Phi, \Psi) \vdash \chi}$	$\dfrac{(\Gamma, \Phi), \Psi \vdash \chi}{(\Gamma, \Psi), \Phi \vdash \chi}$	$\dfrac{(\Gamma, \Phi), \Phi \vdash \chi}{\Gamma, \Phi \vdash \chi}$	$\dfrac{\Gamma \vdash \chi}{\Gamma, \Psi \vdash \chi}$

For example, the Lambek Calculus is the logic that accepts only the associativity rules (while the Pure Lambek Calculus accepts none), which, in terms of resources, means that all formulae must be used in a given order; Linear Logic accepts associativity and commutativity, so formulae must all be used, and only once, but in any order; Relevance Logics further accepts contraction, which allows it to reuse formulae in a deduction. Finally, Intuitionistic Logic accepts all structural rules, and therefore accepts all *constructible* theorems.

The hierarchy of substructural logics has Intuitionistic Logic at its top. Classical Logic would be the next step, accepting all structural rules and non-constructive proofs (we can prove $A \vee \neg A$ in classical logic, without proving either A or $\neg A$), but it remains outside the family of resource-sensitive substructural logics.

2.2 Combinators

Dunn and Meyer [10] noted that the structural rules can be represented by *combinators*. Combinators are λ-terms with no free variables [3]. We present below a few examples of combinators (we represent combinators by capital letters; the choice of letters is historical):

$\mathsf{B} \equiv \lambda xyz.x(yz)$	$\mathsf{B}xyz \twoheadrightarrow x(yz)$	$\mathsf{W} \equiv \lambda xy.xyy$	$\mathsf{W}xy \twoheadrightarrow xyy$
$\mathsf{C} \equiv \lambda xyz.xzy$	$\mathsf{C}xyz \twoheadrightarrow xzy$	$\mathsf{S} \equiv \lambda xyz.xz(yz)$	$\mathsf{S}xyz \twoheadrightarrow xz(yz)$
$\mathsf{I} \equiv \lambda x.x$	$\mathsf{I}x \twoheadrightarrow x$	$\mathsf{K} \equiv \lambda xy.x$	$\mathsf{K}xy \twoheadrightarrow x$

The symbol $\twoheadrightarrow$ means "reduces to" and in the traditional λ-calculus it is replaced by $=$. The right hand-side column shows that combinators can be defined without λ-abstraction and, in this sense, they become *proper* combinators dissociated from the λ-calculus, as in their original formulation [22].

There are also *compound combinators* obtained from the *primitive combinators* above using *functional application*: C_1C_2 reads C_1 applied to C_2; application is also left-associative. For example, combinators W and S are interdefinable in the presence of the I, B and C: $\mathsf{W} = \mathsf{CSI}$ and $\mathsf{S} = \mathsf{B}(\mathsf{BW})(\mathsf{BC}(\mathsf{BB}))$ as it can be verified from their definition above, that is, $\mathsf{W}xy = \mathsf{CSI}xy \twoheadrightarrow \mathsf{S}x\mathsf{I}y \twoheadrightarrow xy(\mathsf{I}y) \twoheadrightarrow xyy$ and $\mathsf{S}xyz = \mathsf{B}(\mathsf{BW})(\mathsf{BC}(\mathsf{BB}))xyz \twoheadrightarrow \mathsf{BW}\,((\mathsf{BC}(\mathsf{BB}))x)\,yz \twoheadrightarrow \mathsf{W}\,((\mathsf{BC})(\mathsf{BB})xy)\,z \twoheadrightarrow \mathsf{BC}(\mathsf{BB})xyzz \twoheadrightarrow \mathsf{C}\,((\mathsf{BB})x)\,yzz \twoheadrightarrow \mathsf{BB}xzyz \twoheadrightarrow \mathsf{B}\,(xz)\,yz \twoheadrightarrow xz(yz)$. Indeed, any λ-definable function (and therefore any combinator) can be expressed in terms of the combinators S and K [3]; however, the set of combinators presented above is very convenient to account for the use of the most common structural rules.

In fact, if we read the structural rules bottom-up, we can see that the B combinator accounts for left-associativity, C for commutativity, I for identity, W for contraction, K for thinning (S is also a type of contraction, not very usual in logics).

2.3 Combinator Logics

Dunn and Meyer [10] proposed a *structurally-free logic* (SFL) where the system is free from any structural presupposition (whence its name). All structural operations have to be accounted for via *combinator rules*, and hence another name for such a logic is *Combinator Logic*. In the language of such a logic, the combinators are considered as special atomic formulae. Hence, $p \to (\mathsf{B} \otimes q)$ is a formula of such a language. The connective rules for such a language are exactly those presented before. However, there are no structural rules in SFL. Instead, we have *connective rules*; a generic combinator rule is

$$\frac{\Gamma[\sigma(\Phi_1, \ldots, \Phi_k)] \vdash \chi}{\Gamma[\mathsf{X}, \Phi_1, \ldots, \Phi_k] \vdash \chi}\ (\mathsf{X} \vdash)$$

(where $\mathsf{X}x_1 \ldots x_k \twoheadrightarrow \sigma(x_1, \ldots, x_k)$) for a combinator that, when applied to the lower sequent as shown, generates the upper sequent.

Some instantiations of the generic combinator rule for the combinators presented above are:

Identity	*Left-associativity*	*Thinning*
$\dfrac{\Gamma[\Phi] \vdash \chi}{\Gamma[\mathsf{I}, \Phi] \vdash \chi}\ (\mathsf{I} \vdash)$	$\dfrac{\Gamma[\Phi, (\Psi, \Xi)] \vdash \chi}{\Gamma[\mathsf{B}, \Phi, \Psi, \Xi] \vdash \chi}\ (\mathsf{B} \vdash)$	$\dfrac{\Gamma[\Phi] \vdash \xi}{\Gamma[\mathsf{K}, \Phi, \Psi] \vdash \xi}\ (\mathsf{K} \vdash)$
Commutativity	*Contraction*	
$\dfrac{\Gamma[\Phi, \Xi, \Psi] \vdash \chi}{\Gamma[\mathsf{C}, \Phi, \Psi, \Xi] \vdash \chi}\ (\mathsf{C} \vdash)$	$\dfrac{\Gamma[\Phi, \Psi, \Psi] \vdash \chi}{\Gamma[\mathsf{W}, \Phi, \Psi] \vdash \chi}\ (\mathsf{W} \vdash)$	

Note that above we also show the structural rule associated with each combinator rule.

Combinator rules leave a "trail" of combinators in a proof, and such combinators are evidence of the structural rules needed for the deduction. For example, to show that $\mathsf{B} \vdash (p \to q) \to [(r \to p) \to (r \to q)]$ we perform the following deduction steps:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{q \vdash q \quad p \vdash p}{p \to q, p \vdash q}\ (\to\vdash) \qquad r \vdash r}{p \to q, (r \to p, r) \vdash q}\ (\to\vdash)}{\mathsf{B}, p \to q, r \to p, r \vdash q}\ (\mathsf{B} \vdash)}{\mathsf{B} \vdash (p \to q) \to [(r \to p) \to (r \to q)]}\ (\vdash\to \quad 3\times)$$

This deduction shows that the formula $(p \to q) \to [(r \to p) \to (r \to q)]$ is a theorem of the $\to$-fragment of all logics which permit the structural rule for left-associativity.

2.4 Structurally-Free Theorem Proving

In the context of Combinator Logics, Finger [11] proposed the notion of *Structurally-Free Theorem Proving* (SFTP), which can be defined as follows. Given an antecedent Γ and a consequent χ, find a combinator X such that $\mathsf{X}, \Gamma \vdash \chi$ is deducible in Combinator Logic. Such an activity is a generalisation of traditional theorem proving, because by inspecting the combinators that compose the answer X, it allows us to answer the question: *in which substructural logics is a given sequent deducible.*

Finger [11] noted that there were a few problems with the logic of combinators, namely the fact that there was no combinator rule to deal with *right-associativity*, and that it could not cope with the $\leftarrow$ connective.

To deal with right associativity, a new combinator B^{-1}, was introduced in [12], with its associated combinator rule:

$$\mathsf{B}^{-1}x(yz) \twoheadrightarrow xyz \qquad \dfrac{\Gamma[(\Phi, \Psi), \Xi] \vdash \chi}{\Gamma[\mathsf{B}^{-1}, \Phi, (\Psi, \Xi)] \vdash \chi}\ (\mathsf{B}^{-1} \vdash)$$

Adding the combinator B^{-1} to the usual combinatorial system introduces some consistency problems; these problems have been addressed and solved in [12] by reducing the class of combinators allowed, without diminishing the set of structural rules representable. It was then shown in [11] that for every intuitionistically valid sequent $\Gamma \vdash \chi$ in the $\{\to, \otimes\}$-fragment, SFTP can be solved; that is, a combinator X can be computed such that $\mathsf{X}, \Gamma \vdash \chi$ is deducible in SFL.

To introduce the $\leftarrow$-connective in the logic, it is necessary to remove combinators as atomic formulae and add them as a label, in the fashion of Labelled Deductive Systems [13]. The algorithm of [11] can then be extended to deal with the $\{\to, \leftarrow, \otimes\}$-fragment.

3 Efficient Knowledge Sharing between Substructural Logics

In this work we explore the issues concerning knowledge sharing among a very specific class of knowledge-based systems: those whose underlying logics are substructural. We shall assume that our systems all share the same vocabulary, that is, their knowledge bases are all subsets of a language $\mathcal{P}$. If this assumption does not hold then a translation function among the languages of each system should be provided [5] or, alternatively, the correspondence among specific constructs to be shared [6] ought to be given.

3.1 Knowledge Sharing between Relevance and Linear Logics

Let there be the set of formulae $\{p \leftarrow q \otimes r, q \leftarrow r, r \leftarrow s, s \leftarrow\}$ comprising the knowledge-base of a donor system. It represents a small fragment of simple propositional knowledge. Let us assume further that the inference engine of such system incorporates a relevant logics [2]. When posed a query $\leftarrow p$ then we have the following sequent proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{q \vdash q \quad r \vdash r}{q, r \vdash q \otimes r}\,(\vdash\otimes) \qquad p \vdash p}{p \leftarrow q \otimes r, (q, r) \vdash p}\,(\leftarrow\vdash)}{\mathsf{B}, p \leftarrow q \otimes r, q, r \vdash p}\,(\mathsf{B}\vdash) \qquad r \vdash r}{\mathsf{B}, p \leftarrow q \otimes r, (q \leftarrow r, r), r \vdash p}\,(\leftarrow\vdash)}{\mathsf{BBB}, p \leftarrow q \otimes r, q \leftarrow r, r, r \vdash p}\,(\mathsf{B}\vdash 2\times)}{\mathsf{B}(\mathsf{BW})(\mathsf{BBB}), p \leftarrow q \otimes r, q \leftarrow r, r \vdash p}\,(\mathsf{B}\vdash, \mathsf{W}\vdash, \mathsf{B}\vdash) \qquad s \vdash s}{\mathsf{B}(\mathsf{BW})(\mathsf{BBB}), p \leftarrow q \otimes r, q \leftarrow r, (r \leftarrow s, s) \vdash s}\,(\leftarrow\vdash)}{\mathsf{B}(\mathsf{BB})(\mathsf{B}(\mathsf{BW})(\mathsf{BBB})), p \leftarrow q \otimes r, q \leftarrow r, r \leftarrow s, s \vdash s}\,(\mathsf{B}\vdash)$$

The combinators of the deduction above can be simplified with the *combinator list* notation of [12], of the form $\langle \mathsf{X}_{(i_1)}, \ldots, \mathsf{X}_{(i_n)} \rangle$, hence we have $\mathsf{B}(\mathsf{BB})(\mathsf{B}(\mathsf{BW})(\mathsf{B}\,\mathsf{B}(\mathsf{BB}(\mathsf{I}))))$ and thus $\langle \mathsf{B}_{(3)}, \mathsf{W}_{(3)}, \mathsf{B}_{(2)}, \mathsf{B}_{(2)} \rangle$.

This list notation shows the *real combinatorial content* of the complex combinator term. The list $\langle \mathsf{B}_{(3)}, \mathsf{W}_{(3)}, \mathsf{B}_{(2)}, \mathsf{B}_{(2)} \rangle$ clearly shows that there were three uses of left-associativity (the B's) and one use of contraction (the W). The indices represent the position on the antecedent where these rules were applied: there was a contraction at the third position, and left-associativity was applied twice at the second position and once at the third. An inspection of the proof shows that this was indeed the case. Details on how this works, and how to compute the list combinator associated with a deduction (without actually constructing the whole deduction) can be found in [11, 12].

The output of the query to the KBS above consists of the proof that $\leftarrow p$ holds, the sequent proof and the list of combinators above: $\langle \mathsf{B}_{(3)}, \mathsf{W}_{(3)}, \mathsf{B}_{(2)}, \mathsf{B}_{(2)} \rangle$. Recipient systems are able to efficiently evaluate the answer provided — rather than examining the proof, a linear scan of the list of combinators is enough to

check for properties of the donor system. In our example, since W is found in the output string of combinators, we conclude that the deduction is not linear. Recipient systems which required that proofs be linear could reject the deduction above just by examining the simplified list of combinators.

Such a method for quickly checking the kind of structural rule (i.e. the control of resources) used by a remote inference system easily generalises to any substructural system. All it is required to do is to send, together with the answer to a query, a combinator list associated with the deduction of the answer to the query.

3.2 Knowledge Sharing between Lambek and Propositional Calculi

Another interesting opportunity to foster knowledge sharing between systems arises in the context of natural language processing. Let us suppose the donor system consists of a parser for English with the fragment of phrase structure grammar [1] $\{S \leftarrow NP \otimes VP, NP \leftarrow Art \otimes N, VP \leftarrow V_{be} \otimes Adj\}$. Such grammar can be represented in categorial grammar [20, 4] as $\{S = s, NP = np, VP = np\backslash s, Art = np/n, N = n, V_{be} = (np\backslash s)/(n/n), Adj = n/n\}$. As is usual in the categorial grammar tradition, the slash notation is used instead of the implications, so the slashes above are simply a notational variation ($A/B \equiv B \rightarrow A$ and $B\backslash A \equiv A \leftarrow B$), so that the directional versions of Modus Ponens are $A/B, B \vdash A$ and $B, B\backslash A \vdash A$.

Let us further suppose that the recipient system wants to know if the string "The tree is green" is a correct sentence in English and submits the list of associated types of the components of the string, that is, $(np\backslash s)/(n/n), n/n, (np/n), n$, to find out whether the donor system can infer a sentence s from such an input.

If the donor system is equipped with an inference system for propositional logic, the following proof is possible:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{np \vdash np \quad s \vdash s}{np, np\backslash s \vdash s}\,(\mathsf{C}_{(n)})}{\langle \mathsf{C}_{(1)} \rangle \,:\, np\backslash s, np \vdash s \quad n\backslash n \vdash n\backslash n}\,(\backslash\vdash)}{\langle \mathsf{C}_{(1)} \rangle \,:\, (np\backslash s)/(n/n), n/n, np \vdash s \quad n \vdash n}\,(/\vdash)}{\langle \mathsf{C}_{(1)} \rangle \,:\, (np\backslash s)/(n/n), n/n, (np/n, n) \vdash s}\,(/\vdash)}{\langle \mathsf{B}_{(3)}, \mathsf{C}_{(1)} \rangle \,:\, (np\backslash s)/(n/n), n/n, np/n, n \vdash s}\,(\mathsf{B}_{(n)})$$

However, upon examination of the associated list $\langle \mathsf{B}_{(3)}, \mathsf{C}_{(1)} \rangle$ of combinators, the recipient system may have a restriction that it will not accept proofs in which the order of formulae is changed (that is, there are occurrences of combinator C). This rejection will happen when the recipient system incorporates Lambek Calculus [9].

The order of the formulae deduced in the antecedent is not exactly the order of the formulae generated by "The tree is green". Indeed, the sequent above is actually a proof that the string "is green The tree" is a grammatically correct sentence, which is obviously not true. Although the categorial grammar above can be employed in many different and useful ways, the recipient system only

wants proofs for those sentences in which no changes in the order of components of the sentence is carried out. The recipient system may ask for an alternative proof, in which case it may be supplied with the last sequent of the following proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{np \vdash np \quad s \vdash s}{np, np\backslash s \vdash s}(\backslash\vdash) \quad n \vdash n}{np/n, n, np\backslash s \vdash s}(/\vdash) \quad n/n \vdash n/n}{np/n, n, ((np\backslash s)/(n/n), n/n \vdash s}(/\vdash)}{\langle \mathsf{B}_{(3)} \rangle \ : \ np/n, n, (np\backslash s)/(n/n), n/n \vdash s}(\mathsf{B}_{(n)})$$

Upon examination of the combinators associated with the proof above, the recipient (Lambek Calculus) system will decide to accept it. Note that the order of the formulae of this sequent follows directly the order of the formulae associated to each word in "The tree is green". Rather than checking the sequent proof, a simpler test of linear complexity is performed.

4 Conclusions and Directions of Research

We have proposed an efficient way to foster opportunistic sharing of knowledge among inference systems which incorporate different kinds of substructural logics (*e.g.*, linear logics, relevance logics, Lambek Calculus and intuitionistic logics). Our proposal employs *combinator logics* [10, 3] as a unifying framework for representing substructural logics and structurally-free theorem proving [11, 12] as a means to characterise their inferences.

We have addressed systems for which a deeper form of knowledge-sharing is sought: not only the correspondence among terms and formulae is necessary (that is, an *ontology* [24]), but also the inferences performed. The final result of inferences as well as their intermediate steps should be available for inspection and rejection or acceptance, depending on the restrictions of those systems sharing their knowledge.

Our approach does not require that each system be translated into one single all-encompassing logic. Rather, each system may incorporate its own distinct substructural logic as long as its inferences are performed structurally-free and the set of combinators appearing in the proofs are shared by the systems involved. Ours is an efficient approach because proofs can be examined (and accepted or rejected) just by scanning a string of combinators.

Prototypical versions of structurally-free theorem provers have already been developed. However, at their present stage, a proof is supplied and only then can it be examined. We would like to investigate an "early-fail" approach thus enabling an inference to be stopped as soon as a particular combinator (or one of a list of combinator) appears. In our Lambek Calculus and Propositional Logic case above (Section 3.2) the first proof could have been aborted as soon as the C combinator appeared and only the last proof would have been supplied.

References

1. James Allen. *Natural Language Understanding.* Benjamin-Cummings Publishing Co., 2nd edition, 1994.

2. A. R. Anderson and N. D. Belnap Jr. *Entailment: The Logic of Relevance and Necessity*. Princeton Univ. Press, 1975.
3. H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Elsevier Science Publishers, 1981.
4. Bob Carpenter. *Type-Logica Semantics*. MIT press, 1997.
5. M. Cerioli and J. Meseguer. May I borrow your Logic? (Transporting Logical Structures along Maps). *Theoretical Computer Science*, 173:311–347, 1997.
6. F. S. Correa da Silva, W. W. Vasconcelos, and D. S. Robertson. Cooperation Between Knowledge-Based Systems. In *Proc. IV World Congress on Expert Systems*, pages 819–825, Mexico City, Mexico, 1998.
7. F. S. Correa da Silva, W. W. Vasconcelos, D. S. Robertson, J. Agustí, and A. C. V. Melo. Why Ontologies are not Enough for Knowledge Sharing. In Springer-Verlag, editor, *LNAI, vol. 1611*, pages 520–529, 1999.
8. D. Van Dalen. Intuitionistic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosoph. Log.*, volume III, 1984.
9. K. Došen. A Historical Introduction to Substructural Logics. In P. S. Heister and K. Došen, editors, *Substructural Logics*, pages 1–31. Oxford Univ. Press, 1993.
10. J. M. Dunn and R. K. Meyer. Combinators and Structurally Free Logic. *Logic Journal of the IGPL*, 5(4):505–538, 1997.
11. M. Finger. Towards structurally-free theorem proving. *Logic Journal of the IGPL*, 6(3):425–449, 1998.
12. M. Finger. Structurally-free theorem proving and the learning of structural permissions in categorial grammar. In *Proc. 4th Workshop on Logical Aspects of Comp. Ling. (LACL98)*, 1998.
13. Dov M. Gabbay. *Labelled Deductive Systems*, volume 1 of *Oxford Logic Guides: 33*. Oxford Univ. Press, 1996.
14. J. Giarratano and G. Riley. *Expert Systems: Principles and Programming*. PWS Publ. Co., 3rd. edition, 1999.
15. J. Y. Girard. Linear Logic. *Theor. Comp. Sc.*, 50:1–102, 1987.
16. J. A. Goguen and R. M. Burstall. Institutions: Abstract Model Theory for Specification and Programming. *J. ACM*, 39:95–146, 1992.
17. P. Gray et al. KRAFT – Knowledge Reuse and Fusion/Transformation. http:// www.csd.abdn.ac.uk/ apreece/ Research/ KRAFT/ KRAFTinfo.html.
18. N. Guarino, editor. *Formal Ontology in Information Systems*. IOS Press, 1998.
19. J. S. Hodas and D. Miller. Logic Programming in a Fragment of Intuitionistic Linear Logic. *Inf. & Comput.*, 110(2):327–365, 1994.
20. M. Moortgat. Categorial type logics. In J. Van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 93–178. Elsevier North-Holland/MIT Press, 1997.
21. R. Neches and D. Gunning. The Knowledge Sharing Effort. http://www-ksl.stanford.edu/knowledge-sharing /papers/kse-overview.html.
22. A. Schönfinkel. Über die Bausteine der Mathematischen Logik. In J. van Heijenoort, editor, *From Frege to Gödel*. Harvard Univ. Press, Cambridge, Mass., 1924. Reprinted.
23. V. S. (project director) Subrahmanian. Hermes – a Heterogeneous Reasoning and Mediator System. http:// www.cs.umd.edu/ projects/ hermes/index.html.
24. M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. *Knowl. Eng. Review*, 11(2):93–136, 1996.
25. M. Winikoff and J. Harland. Implementation and Development Issues for the Linear Logic Programming Language Lygon. In *Proc. 8^{th} Australasian Computer Science Conf.*, pages 562–573, Adelaide, Australia, February 1995.