PARSING AND DERIVATIONAL EQUIVALENCE*

Mark Hepple and Glyn Morrill

Centre for Cognitive Science, University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW Scotland

Abstract

It is a tacit assumption of much linguistic inquiry that all distinct derivations of a string should assign distinct meanings. But despite the tidiness of such derivational uniqueness, there seems to be no a priori reason to assume that a grammar must have this property. If a grammar exhibits derivational equivalence, whereby distinct derivations of a string assign the same meanings, naive exhaustive search for all derivations will be redundant, and quite possibly intractable. In this paper we show how notions of derivation-reduction and normal form can be used to avoid unnecessary work while parsing with grammars exhibiting derivational equivalence. With grammar regarded as analogous to logic, derivations are proofs; what we are advocating is proof-reduction, and normal form proof; the invocation of these logical techniques adds a further paragraph to the story of parsing-as-deduction.

Introduction

The phenomenon of derivational equivalence is most evident in work on generalised categorial grammars, where it has been referred to as 'spurious ambiguity'. It has been argued that the capacity to assign left-branching, and therefore incrementally interpretable, analyses makes these grammars of particular psychological interest. We will illustrate our methodology by reference to generalised categorial grammars using a combinatory logic (as opposed to say, lambda-calculus) semantics. In particular we consider combinatory (categorial) grammars with rules and generalised rules of the kind of Steedman (1987), and with metarules (Morrill, 1988).

Although the problem of derivational equivalence is most apparent in generalised categorial grammars, the problem is likely to recur in many grammars characterising a full complement of constructions. For example, suppose that a grammar is capable of characterising right extraposition of an object's adjunct to clause-final position. Then sentences such as *John met a man yesterday who swims* will be generated. But it is probable that the same grammar will assign *John met a man who swims* a right extraposition derivation in which the relative clause happens to occupy its normal position in the string; the normal and right extraposition derivations generate the same strings with the same meanings, so there is derivational equivalence. Note that a single equivalence of this kind in a grammar undermines a methodological assumption of derivational uniqueness.

Combinatory Logic and Combinatory Grammar

Combinatory logic (CL; Curry and Feys, 1958; Curry, Hindley and Seldin, 1972; Hindley and Seldin, 1986) refers to systems which are applicative, like the lambda-calculi, but which formalise functional abstraction through a small number of basic 'combinators', rather than through a variable-binding operator like λ. We will define a typed combinatory logic. Assume a set of basic types, say *e* and *t*. Then the set of types is defined as follows:

(1) a. If A is a basic type then A is a type

b. If A and B are types then A→B is a type

A convention of right-associativity will be used for types, so that e.g. (e→t)→(e→t) may be writ-

*We thank Mike Reape for criticism and suggestions in relation to this material, and Inge Bethke and Henk Zeevat for reading a late draft. All errors are our own. The work was carried out by the alphabetically first author under ESRC Postgraduate Award C00428722003 and by the second under ESRC Postgraduate Award C00428522008 and an SERC Postdoctoral Fellowship in IT.

ten (e→t)→e→t. There is a set of constants (say, *John'*, *walks'*, ...), and a mapping from the set of constants into the set of types. In addition there are the combinators in (2); their lambda-analogues are shown in parentheses.

(2)

$\mathbf{I}_{A\to A}$	$(\lambda x[x])$
$\mathbf{B}_{(B\to C)\to(A\to B)\to A\to C}$	$(\lambda x\lambda y\lambda z[x(yz)])$
$\mathbf{C}_{(A\to B\to C)\to B\to A\to C}$	$(\lambda x\lambda y\lambda z[(xz)y])$
$\mathbf{W}_{(A\to A\to B)\to A\to B}$	$(\lambda x\lambda y[(xy)y])$

The set of CL-terms is defined thus:

(3) a. If M is a constant or combinator of type A then M is a CL-term of type A

b. If M is a CL-term of type B→A and N is a CL-term of type B then (MN) is a CL-term of type A.

The interpretation of a term built by (3b) is given by the functional application of the interpretation of the left-hand sub-term to that of the right-hand one. We will assume a convention of left-association for application. Some examples of CL-terms are as follows, where the types are written below each component term:

(4) a.

walks'	John'
e→t	e
t	

b.

	C		I
((e→t)→e→t)→e→(e→t)→t		(e→t)→e→t	
e→(e→t)→t			

c.

	B	probably'	walks'
(t→t)→(e→t)→e→t		t→t	e→t
(e→t)→e→t			
e→t			

Other basic combinators can be used in a CL, for example **S**, which corresponds to $\lambda x\lambda y\lambda z[(xz)(yz)]$. Our CL definition is (extensionally) equivalent to the λ**I**-calculus, i.e. the lambda-calculus without vacuous abstraction (terms of the form λxM where x does not occur in M). There is a combinator **K** ($\lambda x\lambda y[x]$) which would introduce vacuous abstraction, and the CL with **S** and **K** is (extensionally) equivalent to the λ**K**-calculus, i.e. the full lambda-calculus.

A combinatory grammar (CG) can be defined in a largely analogous manner. Assume a set of basic categories, say S, NP, ... Then the set of categories is defined as follows:

(5) a. If X is a basic category then X is a category

b. If X and Y are categories then X/Y and X\Y are categories

A convention of left-associativity will be used for categories, so that e.g. (S\NP)\(S\NP) may be written S\NP\(S\NP). There is a set of words, and a lexical association of words with categories. There is a set of rules with combinators, minimally:

(6) a. Forward Application (>)

f: X/Y + Y ⇒ X (where f x y = x y)

b. Backward Application (<)

b: Y + X\Y ⇒ X (where b y x = x y)

The set of CG-terms is defined thus:

(7) a. If M is word of category A then M is a CG-term of category A

b. If $X_1+\cdots+X_n \Rightarrow X_0$ is a rule with combinator ϕ, and $S_1, \ldots, S_n$ are CG-terms of category $X_1, \ldots, X_n$, then $[\phi\ S_1 \ldots S_n]$ is a CG-term of category X_0.

The interpretation of a term built by (7b) is given by the functional application of the combinator to the sub-term interpretations in left-to-right order.

A verb phrase containing an auxiliary can be derived as in (8) (throughout, VP abbreviates S\NP). The meaning assigned is given by (9a), which is equal to (9b).

(8)

will	see	John
VP/VP	VP/NP	NP
	VP	
VP		

(9) a. (f will' (f see' John'))

b. will' (see' John')

Suppose the grammar is augmented with a rule of functional composition (10), as is claimed to be appropriate for analysis of extraction and coordination (Ades and Steedman, 1982; Steedman, 1985). Then for example, the right hand conjunct in (11a) can be analysed as shown in (11b).

(10) Forward Composition (>B)

$$B: X/Y + Y/Z \Rightarrow X/Z \quad (\text{where } B\ x\ y\ z = x\ (y\ z))$$

(11) a. Mary [phoned and will see] John

b.
$$\frac{\underline{\text{will}} \quad \underline{\text{see}}}{\dfrac{\text{VP/VP} \quad \text{VP/NP}}{\text{VP/NP}}\,{>}\mathbf{B}}$$

Forward Application of (11b) to *John* will assign meaning (12) which is again equal to (9b), and this is appropriate because *will see John* is unambiguous.

(12) (f (B will' see') John')

However the grammar now exhibits derivational equivalence, with different derivations assigning the same meaning. In general a sequence $A_1/A_2+A_2/A_3+A_3/A_4+\cdots+A_n$ can be analysed as A_1 with the same meaning by combining any pair of adjacent elements at each step. Thus there are a number of equivalent derivations equal to the number of n-leaf binary trees; this is given by the Catalan series, which is such that Catalan(n) $> 2^{n-2}$. As well as it being inefficient to search through derivations which are equivalent, the exponential figure signifies computational intractability.

Several suggestions have been made in relation to this problem. Pareschi and Steedman (1987) describe what they call a 'lazy chart parser' intended to yield only one of each set of equivalent analyses by adopting a reduce-first parsing strategy, and invoking a special recovery procedure to avoid the backtracking that this strategy would otherwise necessitate. But Hepple (1987) shows that their algorithm is incomplete.

Wittenburg (1987) presents an approach in which a combinatory grammar is compiled into one not exhibiting derivational equivalence. Such compilation seeks to avoid the problem of parsing with a grammar exhibiting derivational equivalence by arranging that the grammar used on-line does not have this property. The concern here however is management of parsing when the grammar used on-line *does* have the problematic property.

Karttunen (1986) suggests a strategy in which every potential new edge is tested against the chart to see whether an existing analysis spanning the same region is equivalent. If one is found, the new analysis is discarded. However, because this check requires comparison with every edge spanning the relevant region, checking time increases with the number of such edges.

The solution we offer is one in which there is a notion of normal form derivation, and a set of contraction rules which reduce derivations to their normal forms, normal form derivations being those to which no contraction rule can apply. The contraction rules might be used in a number of ways (e.g. to transform one derivation into another, rather than recompute from the start, cf. Pareschi and Steedman). The possibility emphasised here is one in which we ensure that a processing step does not create a non-normal form derivation. Any such derivation is dispensable, assuming exhaustive search: the normal form derivation to which it is equivalent, and which won't be excluded, will yield the same result. Thus the equivalence check can be to make sure that each derivation computed is a normal form, e.g. by checking that no step creates a form to which a contraction rule can apply. Unlike Karttunen's subsumption check this test does not become slower with the size of a chart. The test to see whether a derivation is normal form involves nothing but the derivation itself and the invariant definition of normal form.

The next section gives a general outline of reduction and normal forms. This is followed by an illustration in relation to typed combinatory logic where we emphasise that the reduction constitutes a proof-reduction. We then describe how the notions can be applied to combinatory grammar to handle the problem of parsing and derivational equivalence, and we again note that if derivations are regarded as proofs, the method is an instantiation of proof-reduction.

Reduction and Normal Form

It is a common state of affairs for some terms of a language to be equivalent in that for the intended semantics, their interpretations are the same in all models. In such a circumstance it can be useful to elect normal forms which act as unique representatives of their equivalence class. For example, if terms can be transformed into normal forms, equivalence between terms can be equated with identity of normal forms.[1]

The usual way of defining normal forms is by

[1]For our purposes 'identity' can mean exact syntactic identity, and this simplifies discussion somewhat; in a system with bound variables such as the lambda-calculus, identity would mean identity up to renaming of bound variables.

defining a relation $\triangleright$ ('contracts-to') of CONTRACTION between equivalent terms; a term X is said to be in NORMAL FORM if and only if there is no term Y such that $X \triangleright Y$. The contraction relation generates a reduction relation $\geq$ ('reduces-to') and an equality relation $=$ ('equals') between terms as follows:

(13) a. If $X \triangleright Y$ then $X \geq Y$

b. $X \geq X$

c. If $X \geq Y$ and $Y \geq Z$ then $X \geq Z$

(14) a. If $X \triangleright Y$ then $X = Y$

b. $X = X$

c. If $X = Y$ and $Y = Z$ then $X = Z$

d. If $X = Y$ then $Y = X$

The equality relation is sound with respect to a semantic equivalence relation $\equiv$ if $X = Y$ implies $X \equiv Y$, and complete if $X \equiv Y$ implies $X = Y$. It is a sufficient condition for soundness that the contraction relation is valid. Y is a normal form of X if and only if Y is a normal form and $X \geq Y$. A sequence $X_0 \triangleright X_1 \triangleright \cdots \triangleright X_n$ is called a REDUCTION (of X_0 to X_n).

We see from (14) that if there is a T such that $P \geq T$ and $Q \geq T$, then $P = Q$ $(= T)$. In particular, if X and Y have the same normal form, then $X = Y$.

Suppose the relations of reduction and equality generated by the contraction relation have the following property:

(15) Church-Rosser (C-R): If $P = Q$ then there is a T such that $P \geq T$ and $Q \geq T$.

There follow as corollaries that if P and Q are distinct normal forms then $P \neq Q$, and that any normal form of a term is unique.[2] If two terms X and Y have distinct normal forms P and Q, then $X = P$ and $Y = Q$, but $P \neq Q$, so $X \neq Y$.

[2]Suppose P and Q are distinct normal forms and that $P = Q$. Because normal forms only reduce to themselves and P and Q are distinct, there is no term to which P and Q can both reduce. But C-R tells us that if $P = Q$, then there *is* a term to which they can both reduce. And suppose that a term X has distinct normal forms P and Q; then $X = P$, $X = Q$, and $P = Q$. But by the first corollary, for distinct normal forms P and Q, $P \neq Q$.

We have established that if two terms have the same normal form then they are equal and (given C-R) that if they have different normal forms then they are not equal, and that normal forms are unique. Suppose we also have the following property:

(16) Strong Normalisation (SN): Every reduction is finite.

This has the corollary (normalisation) that every term has a normal form. A sufficient condition to demonstrate SN would be to find a metric which assigns to each term a finite non-negative integer score, and to show that each application of a contraction decrements the score by a non-zero integral amount. It follows that any reduction of a term must be finite. Given both C-R and SN, equality is decidable: we can reduce any terms to their normal forms in a finite number of steps, and compare for identity.

Normal Form and Proof-Reduction in Combinatory Logic

In the CL case, note for example the following equivalence (omitting types for the moment):

(17) **B** probably′ walks′ John′ $\equiv$ probably′ (walks′ John′)

We may have the following contraction rules:

(18) a. **I M** $\triangleright$ **M**

b. **B M N P** $\triangleright$ **M (N P)**

c. **C M N P** $\triangleright$ **M P N**

d. **W M N** $\triangleright$ **M N N**

These state that any term containing an occurrence of the form on the left can be transformed to one in which the occurrence is replaced by the form on the right. A form on the left is called a REDEX, the form on the right, its CONTRACTUM. To see the validity of the contraction relation defined (and the soundness of the consequent equality), note that the functional interpretations of a redex and a contractum are the same, and that by compositionality, the interpretation of a term is unchanged by substitution of a subterm for an occurrence of a subterm with the same interpretation. An example of reduction of a term to its normal form is as follows:

(19) C I John′ (B probably′ walks′) ▷
I (B probably′ walks′) John′ ▷
B probably′ walks′ John′ ▷
probably′ (walks′ John′)

Returning to emphasise types, observe that they can be regarded as formulae of implicational logic. In fact the type schemes of the basic combinators in (2), together with a modus ponens rule corresponding to the application in (3b), provide an axiomatisation of relevant implication (see Morrill and Carpenter, 1987, for discussion in relation to grammar):

(20) a. $A \rightarrow A$
$(B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C$
$(A \rightarrow B \rightarrow C) \rightarrow (B \rightarrow A \rightarrow C)$
$(A \rightarrow A \rightarrow B) \rightarrow A \rightarrow B$

b. $\dfrac{B \rightarrow A \quad B}{A}$

Consider the typed CL-terms in (4). For each of these, the tree of type formulae is a proof in implicational relevance logic. Corresponding to the term-reduction and normal form in (19), there is proof-reduction and a normal form for a proof over the language of types (see e.g. Hindley and Seldin, 1986). There can be proof-contraction rules such as the following:

(21)
$$\dfrac{\dfrac{\dfrac{\dfrac{B}{(B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C} \quad \dfrac{N}{B \rightarrow C} \quad \dfrac{M}{A \rightarrow B}}{(A \rightarrow B) \rightarrow A \rightarrow C} \quad \dfrac{P}{A}}{A \rightarrow C}}{C}$$

$$\triangleright \quad \dfrac{\dfrac{N}{B \rightarrow C} \quad \dfrac{\dfrac{M}{A \rightarrow B} \quad \dfrac{P}{A}}{B}}{C}$$

Proof-reduction originated with Prawitz (1965) and is now a standard technique in logic. The suggestion of this paper is that if parse trees labelled with categories can be regarded as proofs over the language of categories, then the problem of parsing and derivational equivalence can be treated on the pattern of proof-reduction.

Before proceeding to the grammar cases, a couple of remarks are in order. The equivalence addressed by the reductions above is not strong (extensional), but what is called weak equivalence. For example the following pairs (whose types have been omitted) are distinct weak normal forms, but are extensionally equivalent:

(22) a. **B (B probably′necessarily′) walks′**

b. **B probably′(B necessarily′walks′)**

(23) a. **B I walks′**

b. **walks′**

Strong equivalence and reduction is far more complex than weak equivalence and reduction, but unfortunately it is the former which is appropriate for the grammars. Later examples will thus differ in this respect from the one above. A second difference is that in the example above, combinators are axioms, and there is a single rule of application. In the grammar cases combinators are rules. Finally, grammar derivations have both a phonological interpretation (dependent on the order of the words), and a semantic interpretation. Since no derivations are equivalent if they produce a different sequence of words, derivation reduction must always preserve word order.

Normal Form and Proof-Reduction in Combinatory Grammar

Consider a combinatory grammar containing the application rules, Forward Composition, and also Subject Type-Raising (24); the latter two enable association of a subject with an incomplete verb phrase; this is required in (25), as shown in (26).

(24) Subject Type-Raising (>T)

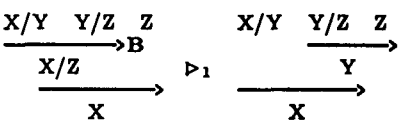
T: NP ⇒ S/(S\NP) (where T y x = x y)

(25) a. [John likes and Mary loves] opera

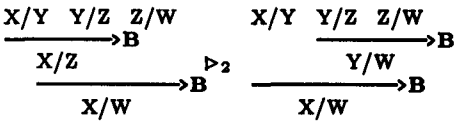
b. the man who John likes

(26)
$$\dfrac{\dfrac{\dfrac{\text{John}}{\text{NP}} \quad \dfrac{\text{likes}}{\text{S\NP/NP}}}{\text{S/(S\NP)}}\,{>}\mathbf{T}}{\text{S/NP}}\,{>}\mathbf{B}$$

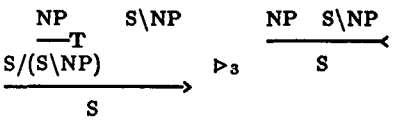
This grammar will allow many equivalent derivations, but consider the following contraction rules:

(27) a. 

(f (B x y) z) = (f x (f y z))

b. 

(B (B x y) z) = (B x (B y z))

c. 

(f (T x) y) = (b x y)

Each contraction rule states that a derivation containing an occurrence of the redex can be transformed into an equivalent one in which the occurrence is replaced by the contractum. To see that the rules are valid, note that in each contraction rule constituent order is preserved, and that the determination of the root meaning in terms of the daughter meanings is (extensionally) equivalent under the functional interpretation of the combinators.

Observe by analogy with combinatory logic that a derivation can be regarded as a proof over the language of categories, and that the derivation-reduction defined above is a proof-reduction. So far as we are aware, the relations of reduction and equality generated observe the C-R corollaries that distinct normal forms are non-equal, and that normal forms are unique. We provide the following reasoning to the effect that SN holds.

Assign to each derivation a score, depending on its binary and unary branching tree structure as follows:

(28) a. An elementary tree has score 1

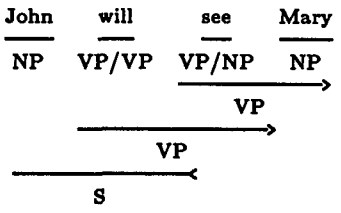
b. If a left subtree has score x and a right subtree has score y, the binary-branching tree formed from them has score $2x + y$

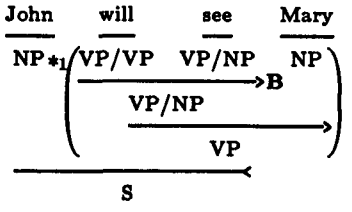
c. If a subtree has score x then a unary-branching tree formed from it has score $2x$

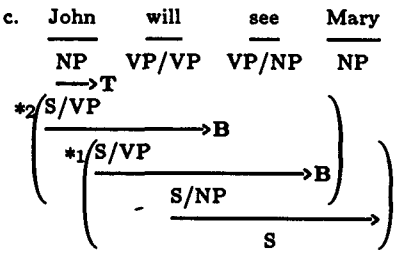
All derivations will have a finite score of at least 1. Consider the scores for the redex and contractum in each of the above. Let x, y, and z be the scores for the subtrees dominated by the leaves in left-to-right order. For $\triangleright_1$, the score of the redex is $2(2x+y)+z$ and that of its contractum is $2x+(2y+z)$: a decrement of $2x$, and this is always non-zero because all scores are at least 1. The case of $\triangleright_2$ is the same. In $\triangleright_3$ the score of the redex is $2(2x) + y$, that of the contractum $2x+y$: also a proper decrement. So all reductions are finite, and there is the corollary that all derivations have normal forms.

Since all derivations have normal forms, we can safely limit attention in parsing to normal form derivations: for all the derivations excluded, there is an equivalent normal form which is not excluded. If not all derivations had normal forms, limitation to normal forms might lose those derivations in the grammar which do not have normal forms. The strategy to avoid unnecessary work can be to discontinue any derivation that contains a redex. The test is neutral as to whether the parsing algorithm is, e.g. top-down or bottom-up.

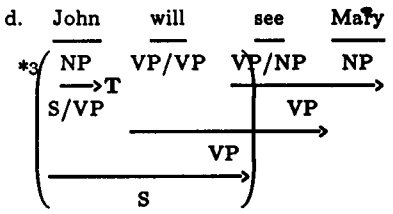
The seven derivations of *John will see Mary* in the grammar are shown below. Each occurrence of a redex is marked with a correspondingly labelled asterisk. It will be seen that of the seven logical possibilities, only one is now licensed:

(29) a. 

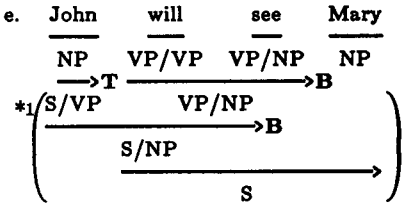
b. 

c. 

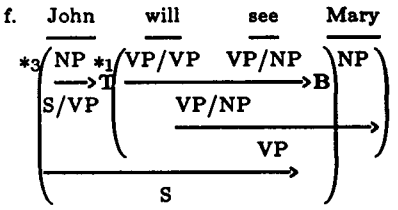
d.



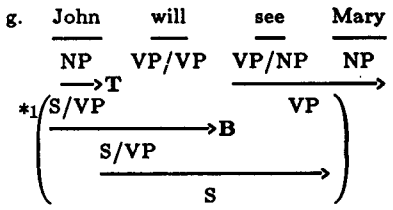
e.



f.

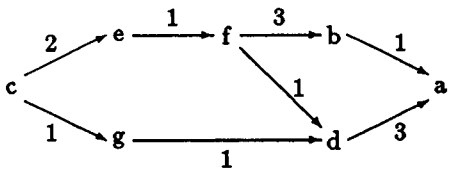


g.



The derivations are related by the contraction relation as follows:

(30)



Consider now the combinatory grammar obtained by replacing Forward Composition by the Generalised Forward Composition rule (31a), whose semantics $\mathbf{B}^n$ is recursively defined in terms of $\mathbf{B}$ as shown in (31b).

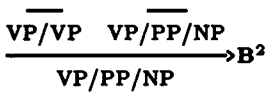
(31) a. Generalised Forward Composition (>$\mathbf{B}^n$):

$$\mathbf{B}^n: X/Y + Y/Z_1\cdots/Z_n \Rightarrow X/Z_1\cdots/Z_n$$

b. $\mathbf{B}^1 = \mathbf{B}; \quad \mathbf{B}^{n+1} = \mathbf{BBB}^n, \quad n \geq 1$

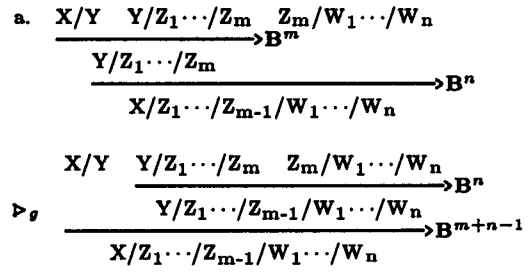
This rule allows for combinations such as the following:

(32)



We may accompany the adoption of this rule with replacement of the contraction rule (27b) by the following generalised version:

(33) a.



b. $(\mathbf{B}^n\ (\mathbf{B}^m\ x\ y)\ z) = (\mathbf{B}^{(m+n-1)}\ x\ (\mathbf{B}^n\ y\ z))$
for $n \geq 1;\ m \geq 1$

It will be seen that (33a) has (27b) as the specia case $n = 1$, $m = 1$. Furthermore, if we admit combinator $\mathbf{B}^0$ which is equivalent to the combi nator $\mathbf{f}$, and use this as the semantics for Forwar Application, we can extend the generalised contrac tion rule (33) to have (27a) as a special case als (by allowing the values for m and n to be such tha $n \geq 0$; $m \geq 1$). It will be seen that again, ever contraction results in a proper decrement of th score assigned, so that SN holds.

In Morrill (1988) it is argued at length that eve rules like generalised forward composition are no adequate to characterise the full range of extrac tion and coordination phenomena, and that deepe generalisations need to be expressed. In particula a system is advocated in which more complex rule are derived from the basic rules of application b the use of metarules, like that in (34); these are sim ilar to those of Gazdar (1981), but with slash inte preted as the categorial operator (see also Geac 1972, p485; Moortgat, 1987, p18).

(34) Right Abstraction

$$\phi: X + Y \Rightarrow V \implies R\phi: X + Y/Z \Rightarrow V/Z$$

(where $(\mathbf{R}\ g\ x\ y)\ z = g\ x\ (y\ z)$)

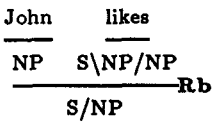
Note for instance that applying Right Abstractio to Forward Application yields Steedman's Forwar Composition primitive, and that successive appl cation yields higher order compositions:

(35) a. **Rf**: X/Y + Y/Z ⇒ X/Z

b. **R(Rf)**: X/Y + Y/Z/W ⇒ X/Z/W

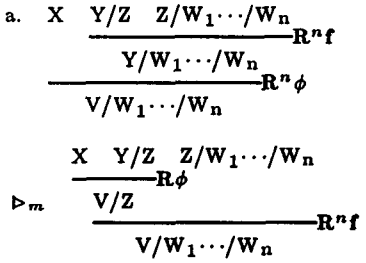
Applying Right Abstraction to Backward Application yields a combinator capable of assembling a subject and incomplete verb phrase, without first type-raising the subject:

(36) a. **Rb**: Y + X\Y/Z ⇒ X/Z

b.


(Note that for this approach, the labelling for a rule used in a derivation is precisely the combinator that forms the semantics for that rule.)

Consider a grammar with just the application rules and Right Abstraction. Let $\mathbf{R}^n\phi$ be $\mathbf{R}(\ldots\mathbf{R}(\phi)\ldots)$ with $n \geq 0$ occurrences of **R**. Instead of the contraction rules earlier we may have:

(37) a.


$\triangleright_m$

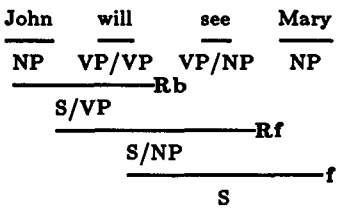
b. $(\mathbf{R}^n\phi\ x\ (\mathbf{R}^n f\ y\ z)) = (\mathbf{R}^n f\ (\mathbf{R}\phi\ x\ y)\ z)$

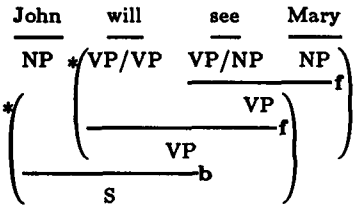
Suppose we now assign scores as follows:

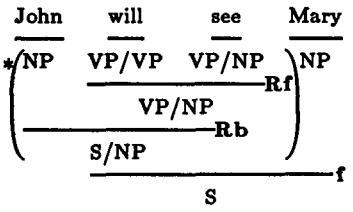
(38) a. An elementary tree has score 1

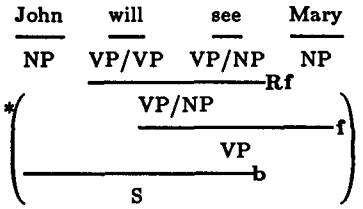
b. If a left subtree has score x and a right subtree has score y, the binary-branching tree formed from them has score $x + 2y$

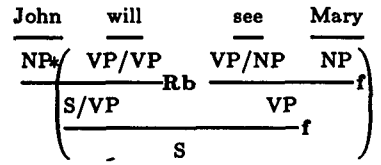
The score of a redex will be $x+2(y+2z)$ and that of its contractum $(x + 2y) + 2z$: a proper decrement, so SN holds and all derivations have normal forms as before. For the sentence *John will see Mary*, the grammar allows the set of derivations shown in (39).

(39) a.


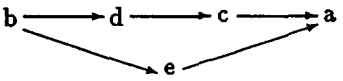
b.


c.


d.


e.


As before, we can see that only one derivation, (39b), contains no redexes, and it is thus the only admissible normal form derivation. The derivations are related by the contraction relation as follows:

(40)


Conclusion

We have offered a solution to the problem of parsing and derivational equivalence by introducing a notion of normal-form derivation. A definition of redex can be used to avoid computing non-normal form derivations. Computing only normal form derivations is safe provided every non-normal form derivation has a normal form equivalent. By

demonstrating strong normalisation for the examples given, we have shown that every derivation does have a normal form, and that consequently parsing with this method is complete in the sense that at least one member of each equivalence class is computed. In addition, it would be desirable to show that the Church-Rosser property holds, to guarantee that each equivalence class has a unique normal form. This would ensure that parsing with this method is optimal in the sense that for each equivalence class, only one derivation is computed.

References

Ades, A. and Steedman, M. J. 1982. On the Order of Words. *Linguistics and Philosophy*, 4: 517-558.

Curry, H. B. and Feys, R. 1958. *Combinatory logic*, Volume I. North Holland, Amsterdam.

Curry, H. B., Hindley, J. R. and Seldin, J. P. 1972. *Combinatory logic*, Volume II. North Holland, Amsterdam.

Gazdar, G. 1981. Unbounded dependencies and coordinate structure. *Linguistic Inquiry*, 12: 155-184.

Geach, P. T. 1972. A program for syntax. In Davidson, D. and Harman, G. (eds.) *Semantics of Natural Language*. Dordrecht: D. Reidel.

Hindley, J. R. and Seldin, J. P. 1986. *Introduction to combinators and λ-calculus*. Cambridge University Press, Cambridge.

Hepple, M. 1987. Methods for Parsing Combinatory Grammars and the Spurious Ambiguity Problem. Masters Thesis, Centre for Cognitive Science, University of Edinburgh.

Karttunen, L. 1986. Radical Lexicalism. Report No. CSLI-86-68, Center for the Study of Language and Information, December, 1986. Paper presented at the Conference on Alternative Conceptions of Phrase Structure, July 1986, New York.

Moortgat, M. 1987. Lambek Categorial Grammar and the Autonomy Thesis. INL Working Papers No. 87-03, Instituut voor Nederlandse Lexicologie, Leiden, April, 1987.

Morrill, G. 1988. Extraction and Coordination in Phrase Structure Grammar and Categorial Grammar. PhD Thesis, Centre for Cognitive Science, University of Edinburgh.

Morrill, G. and Carpenter, B. 1987. Compositionality, Implicational Logics, and Theories of Grammar. Research Paper No. EUCCS/RP-11, Centre for Cognitive Science, University of Edinburgh, Edinburgh, June, 1987. To appear in *Linguistics and Philosophy*.

Pareschi, R. and Steedman, M. J. 1987. A Lazy Way to Chart-Parse with Extended Categorial Grammars. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, Ca., 6-9 July, 1987.

Prawitz, D. 1965. *Natural Deduction: A Proof-Theoretical Study*. Almqvist and Wiksell, Uppsala.

Steedman, M. 1985. Dependency and Coordination in the Grammar of Dutch and English. *Language*, 61: 523-568.

Steedman, M. 1987. Combinatory Grammars and Parasitic Gaps. *Natural Language and Linguistic Theory*, 5: 403-439.

Wittenburg, K. 1987. Predictive Combinators: a Method for Efficient Processing of Combinatory Categorial Grammar. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, Ca., 6-9 July, 1987.