EDWARD P. STABLER, JR.

AVOID THE PEDESTRIAN'S PARADOX

1. THE PARADOX

Mark Steedman (1989) has suggested that each of the following assumptions about human language understanding is appealing, but that they lead to a paradox when taken together:

1. *Incremental comprehension:* Human natural language understanding is serial and incremental: the words of a sentence are interpreted rapidly as they are heard or read.

2. *Right-branching syntactic structures:* English, like other SVO and SOV languages, has predominantly right-branching syntactic structures.

3. *Strong competence hypothesis:* The principles of the competence grammar are directly used by the human language processor in constructing a syntactic structure and interpreting it.

The problem is that the first complete, interpretable phrase containing the first word of a sentence may contain many words, and yet there is abundant introspective and experimental support for the view that an interpretation of the first words is available almost immediately. Steedman concludes that one of the three assumptions must be abandoned. Similar arguments have played a prominent role in psycholinguistic disputes before (see section 4 below), but never have they been so crisply stated.

Since the empirical support for the first assumption is so overwhelming, the other two assumptions become suspect. The strong competence hypothesis is often rejected, but this makes the prospects for adequate performance and acquisition models seem rather bleak. If the human parser does not use the linguists' grammar, then the psychologist is left with the problem of saying what rules the human parser *does* use and how *those* rules are acquired.[1] Given the complexity of our phonological, syntactic, and semantic abilities, this is a formidable task. Another less

R. C. Berwick et al. (eds.),
Principle-Based Parsing: Computation and Psycholinguistics, 199–237.

common response to the paradox, recently championed by Steedman and others, has been to adopt a left-branching syntax, even for languages like English. This option is unappealing, though, until we have disarmed the constituency arguments that support right-branching accounts.

So we seem to be faced with the the most challenging kind of paradox: we do not want to reject any of the assumptions of the argument. In his famous essay on paradoxes, Quine (1961, p. 5) takes particular delight in puzzles like this, and points out the only alternative to rejecting an assumption: "some tacit and trusted pattern of reasoning must be made explicit and henceforward avoided or revised". This is the response I want to urge: *the assumptions are not inconsistent.* A tacit and eminently discardable assumption plays an essential role in the argument that there is a paradox here. Once this tacit assumption is revealed, we see that assumptions (1)–(3) are not only logically compatible, but there is not even any tension among them that should worry us. Furthermore, once this is granted, we see that the strong competence hypothesis has been rejected too quickly by many psychologists and philosophers. Many of the arguments against the strong competence hypothesis ultimately rest on essentially the same implausible assumption that generates Steedman's paradox.

2. REJECTING THE PEDESTRIAN'S ASSUMPTION

The suggested problem is that in right-branching languages, words are not generally rightmost in the interpretable phrases containing them, and yet interpretation develops almost continuously, word-by-word or even more quickly than that (Marslen-Wilson, 1987; 1989; Marslen-Wilson and Tyler, 1980; etc.). In left-branching constructions like that in figure 1, this problem would not arise because each word completes a constituent.

The problem is that, on standard syntactic accounts, languages are not restricted to structures of this form. Even in predominantly left-branching languages, some right-branching constructions occur, and yet there is no psychological support for the idea that a right-branching construction disrupts incremental interpretation. Steedman suggests that this is a problem for any performance model in which the rules of the grammar correspond to processing steps. The reasoning here is presumably that if the grammar has rules defining the interpretation of syntactic constituents, the parser cannot deliver syntactic constituents

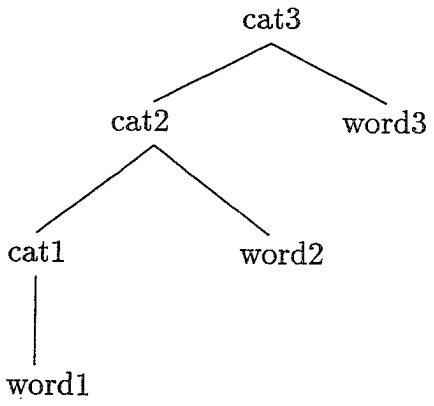


Figure 1: A left-branching constituent structure. Each word completes a constituent in this kind of phrase structure, making word-by-word interpretation straightforward.

for interpretation on a word-by-word basis, because many words do not complete (nonlexical) constituents. A different performance model that defines some special kind of interpretable 'partial constituents' would not be a strong competence model since it would need these special representations and rules defined over them.

The fallacy arises in this last step. Once we suppose that understanding involves the two steps of identifying a syntactic constituent and interpreting it, it is a mistake to think that we must finish the first step before beginning the second. Let's call this the pedestrian's assumption. Notice that this perfectly reasonable idea about walking is not a reasonable idea about, say, preparing and serving a dinner. It is true that the food must be prepared before it is served, but only the most inexperienced chef would *finish* the preparation before beginning the serving. The salad can go in the bowl before the steak is done cooking. We still perform both steps, preparing and serving, literally and completely, and completing the second depends on having completed the first. Furthermore, in a clear sense, we do not need to be engaged, at any given instant, in doing two things at once. Some sort of parallel processing is a possibility for expert and ambidextrous chefs, but the essential point is that each step consists in performing a number of subtasks, and so we can perform all and only these subtasks using a strategy that intermingles them. Once this is allowed as a possibility, it

offers another way out of the paradox. Not only is this a *possible* way to avoid the paradox, but this way out is demanded by the evidence.

Computer scientists are fond of noting that we can define the alphabetization of a list as a permutation or reshuffling of the original list that respects the alphabetic ordering, but it would be silly to use this as a recipe for alphabetizing lists. For example, the definition might suggest that we could alphabetize a list with the following procedure:

(1) Reshuffle the items in the list,

(2) Check to see if the new list is alphabetic. If it is, we are done; if not, go to step 1.

To help us remember just how inefficient this is, Knuth (1973) recommends that we commit to memory the fact that a 10 element list has about $3\frac{1}{2}$ million permutations. As every filing clerk knows, all of the reasonable alphabetizing procedures, in effect, interleave the reshuffling step with the ordering checks. Computational linguists have similar points to keep in mind. Church and Patil (1982) have argued that because of the attachment ambiguity in English, an English sentence with 10 prepositional phrases and relative clauses, a sentence like

I put the bouquet of flowers that you gave me for Mother's Day in the vase that you gave me for my birthday on the chest of drawers that you gave me for Armistice Day.

can have 4,862 different syntactic trees (even if it has no other kind of syntactic ambiguity). And Langendoen (1987) has argued that a sentence with 10 coordinate expressions can have 103,049 different structures. He found the following example in his corpus:

Combine grapefruit with bananas, strawberries and bananas, bananas and melon balls, raspberries or strawberries and melon balls, seedless white grapes and melon balls, or pineapple cubes with orange slices.

Local ambiguities can be even more numerous than global ambiguities like these. So if these observations are close to right, the *parse first, then interpret* strategy is infeasible, and would still be so even if we had no reason to believe the first of the three assumptions of the purported paradox. That is, the mere fact that you can successfully manage a filing task or a comprehension task tells us something about how you must be

doing it: you cannot be using a hopelessly pedestrian strategy in either case.[2]

No one seriously proposes pedestrian approaches to language understanding according to which *sentences* are fully parsed and then interpreted. So what could make the three assumptions with which we began seem paradoxical? It is just the idea that, although we do not need to have complete sentence structures before interpretation begins, we do at least need complete syntactic constituents, such as phrases. But why keep this assumption? The structures of phrases are complex (*i.e.*, composed of a number of elements), as are the structures of sentences, so we can perfectly well assume that the constituent parts of phrase structures become available for interpretation as soon as syntactic analysis formulates them. Furthermore, since the interpretation of a syntactic constituent is definitionally and procedurally complex, typically involving the interpretation of subconstituents, these subtasks can be intermingled with the parsing subtasks, *without making any modification at all in our definition of what each task involves.* Similarly for later stages of processing.

I will call a language processing system *pedestrian* just in case semantic processing of syntactic structures begins only when the input syntactic constituents are complete, and assessment of reference and truth values does not begin until completed constituents of semantic form are available. What Steedman's paradox really shows is that no one should seriously propose pedestrian approaches to language understanding, according to which *phrasal constituents* of *any* category must be fully parsed before their interpretation starts.[3] Once the nonpedestrian alternatives are noted, Steedman's three assumptions can all be embraced at once.

3. A SIMPLE NONPEDESTRIAN PARSER

The idea of nonpedestrian parsing is not new, but we do not need to look at large, complex natural language understanding systems to illustrate the approach. A little example suffices to show that nonpedestrian parsing is a very natural option, requiring no special rules about partial trees or any such thing. I will present in complete detail a little example of a nonpedestrian parser that respects the strong competence hypothesis (with respect to a little context-free grammar) and provides incremental interpretation (and even incremental referential assessment) on a word-

by-word basis. It is clear that the same basic ideas apply to applications of arbitrarily complex grammars to arbitrary linguistic tasks. (Anyone willing to accept the existence of simple examples of this sort can skip to section 4 without losing the main thread of the argument.)

In this example, we represent syntactic, semantic, and interpretive rules as directly as possible in standard first-order logic, adding *nothing* special for our nonpedestrian use of those rules. Since the rules are represented as directly as possible, and since they are used by a sound and complete deductive strategy, there can hardly be any worry about adherence to the strong competence hypothesis in this case. We will explain the nonpedestrian proof strategy (and a complete implementation of it is provided in appendix A). Since the grammar is represented and used in the most direct way, there would also be no obstacle to the use of an acquisition theory for these grammars, if one were available (as we hope there will be in the natural language case). The key advantage of using a transparent *logical* representation is roughly that we have a well-developed proof theory that can be used to guarantee that various proof techniques are sound and complete, as we will see below. We could of course consider models that differ from the one presented here along many dimensions: interactive vs. noninteractive, top-down vs. bottom-up, parallel vs. serial, but these distinctions are irrelevant to defusing the purported paradox. It happens though, that our example is interactive, top-down, and serial. These choices were made for simplicity of exposition, and none of them is essential to the argument.

The syntax of our example is trivial but definitely right-branching, our semantic rules apply to the specified syntactic constituents, and our assessment of reference and truth is, in turn, based on constituents of the semantic representation. Nevertheless, the formulation of logical form and the assessment of reference and truth can be done in a very elegant way on a word-by-word basis, using simple deductive reasoning. This by itself would defuse the paradox, but the example is so simple and natural that we see that there is truly no tension at all among the three assumptions that Steedman has drawn to our attention. The nonpedestrian strategy is nothing more than a very simple proof technique: we use a natural principle to decide the order in which steps of the proof are to be taken.

3.1. *Syntax*

In this notation, predicates and function symbols begin with lower case letters, while variables begin with upper case letters, the sentences are all universally closed, and we write material conditionals of the form $(p \wedge q) \rightarrow r$ as $r \leftarrow q \wedge r$. We will represent a simple rewrite rule like the following:

$$ip \rightarrow np\ vp$$

with the following sentence of first-order logic:

$$ip(L0, L) \leftarrow np(L0, L1) \wedge vp(L1, L).$$

The two arguments of each grammatical predicate represent the string to which the predicate applies; it is the string that remains when a list L, given as the second argument, is removed from the end of the list $L0$ given as the first argument. Using the notation $L0 - L$ ($L0$ minus L) to mean the string that results from removing list L from the end of list $L0$, our logical sentence can be read, 'for all $L0$ and L, the string $L0 - L$ is an inflectional phrase *(i.e.*, a sentence or IP) if the string $L0 - L1$ is a noun phrase and the string $L1 - L$ is a verb phrase'. With this interpretation of grammatical predicates, it is no surprise that lexical rewrite rules have a slightly different appearance. Using the notation $[the|L]$ to represent the string that begins with 'the' and ends with the list L, it is clear that $[the|L] - L$ is simply the word 'the', and so the rewrite rule,

$$d \rightarrow the$$

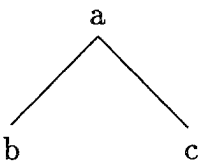
is represented by the logical sentence

$$d([the|L], L).$$

In other words, 'the' is a determiner, no matter what string L follows it. This logical representation of the grammar not only expresses just what the grammar does, but there is an isomorphism between derivations of strings from the grammar and deductions of results about strings from the logical representation of the grammar.[4]

Since we are interested in parsing, rather than merely recognizing or generating strings, a slight elaboration of these logical sentences is required, again along the lines suggested by Pereira and Warren (1980).

We simply add an extra argument to each grammatical predicate to hold the derivation tree that is, in effect, dominated by that predicate. We represent trees like,



with the term a/[b,c]. The '/' represents the 'immediately dominates' relation which holds between a node and the ordered list of subtrees of which it is the parent. Then the two rules shown above are elaborated as follows:

$$ip(L0, L, ip/[NP, VP]) \leftarrow np(L0, L1, NP) \wedge vp(L1, L, VP). \quad (S1)$$
$$d([the|L], L, d/[the]). \quad (S2)$$

Sentence (S1) now says, for example, that (for all $L0, L, NP, VP$) $L0 - L$ is a sentence with structure $ip/[NP, VP]$ if $L0 - L1$ is a noun phrase with structure NP and $L1 - L$ is a verb phrase with structure VP. To complete our simple syntax we add just 9 more rules:

$$np(L0, L, np/[D, N]) \leftarrow d(L0, L1, D) \wedge n(L1, L, N). \quad (S3)$$
$$np(L0, L, np/[D, N, CP]) \leftarrow d(L0, L1, D) \wedge n(L1, L2, N) \wedge cp(L2, L, CP). \quad (S4)$$
$$cp(L0, L, cp/[C, IP]) \leftarrow c(L0, L1, C) \wedge ip(L1, L, IP). \quad (S5)$$
$$vp(L0, L, vp/[V, A]) \leftarrow v(L0, L1, V) \wedge a(L1, L, A). \quad (S6)$$
$$n([joke|L], L, n/[joke]). \quad (S7)$$
$$n([argument|L], L, n/[argument]). \quad (S8)$$
$$c([that|L], L, c/[that]). \quad (S9)$$
$$v([is|L], L, v/[is]). \quad (S10)$$
$$a([funny|L], L, a/[funny]). \quad (S11)$$

The language generated by this grammar includes noun phrases with arbitrarily deep center embeddings:

the joke is funny
the argument is funny
the joke that the argument is funny is funny

the argument that the joke that the argument is funny is funny is funny

...

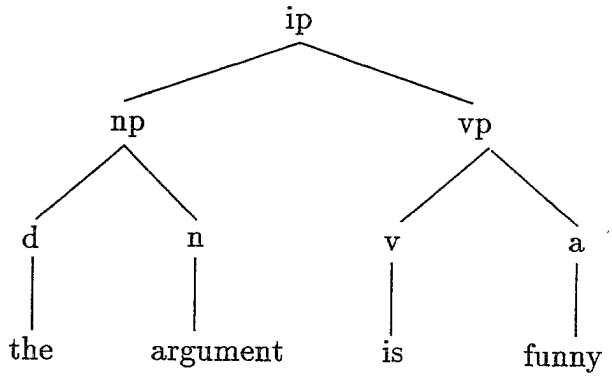
Using standard resolution proof techniques, it is a straightforward matter to use this theory in a parser, obtaining the parse trees from the substitutions used in the proofs that the strings are in the language. For example, a (pure) Prolog proof (which is a perfectly sound logical deduction) suffices to prove that there is a parse tree for the string *the joke is funny*:[5]

$$
\begin{aligned}
& \leftarrow ip([the, joke, is, funny], [\,], Tree) \\
\textit{by } (S1): \; & \leftarrow np([the, joke, is, funny], L1, NP) \wedge vp(L1, [\,], VP) \\
\textit{by } (S3): \; & \leftarrow d([the, joke, is, funny], L2, D) \wedge n(L2, L1, N) \wedge \\
& \quad vp(L1, [\,], VP) \\
\textit{by } (S2): \; & \leftarrow n([joke, is, funny], L1, N) \wedge vp(L1, [\,], VP) \\
\textit{by } (S7): \; & \leftarrow vp([is, funny], [\,], VP) \\
\textit{by } (S6): \; & \leftarrow v([is, funny], L1, V) \wedge a(L1, [\,], A) \\
\textit{by } (S10): \; & \leftarrow a([funny], [\,], A) \\
\textit{by } (S11): \; & \square \quad Q.E.D.
\end{aligned}
$$

We can collect from this proof the instance of the variable $Tree$ that was shown to be the parse of the string:

$$ip/[np/[d/[the], n/[argument]], vp/[v/[is], a/[funny]]].$$

The graphical depiction of this tree is much more readable:



It is a simple matter to demonstrate that a similar proof is available for all and only the strings in the language generated by the rewrite grammar that corresponds to our axioms in the obvious way (Stabler, 1991 forthcoming). Note that the first complete noun phrases can be arbitrarily long, and so the language has the appropriate right-branching structure.

3.2. *Logical Form*

The logical form component for our trivial context-free grammar can be similarly trivial. We can adopt a naive approach to definite descriptions like 'the joke' and treat them as functions. Thus, 'the joke' is a joke we will call $joke(0)$. Intuitively, it does not matter what this joke is about, so let's simply call the subject of the joke '0'. A simple predication, saying that this joke is funny, is sometimes represented as $funny(joke(0))$ or $\lambda X funny(X) joke(0)$ or $joke(0) \in funny$ or $funny \in joke(0)$, but we will use the following simple notation to signify that the object $joke(0)$ satisfies the predicate $funny$:

$$sat(joke(0), funny).$$

Then the joke that the joke is funny can be represented as the function,

$$joke(sat(joke(0), funny)).$$

In other words, this is the joke about $sat(joke(0), funny)$ which is, in turn, the joke's being funny. To map the syntactic structures into these logical forms the following rules do the trick:[6]

$$i\_ip([ip/[NP|VP]], sat(Subj, Pred)) \leftarrow i\_np([NP], Subj) \wedge i\_vp(VP, Pred). \quad (I1)$$

$$i\_np([np/[D|N1]], I\_N) \leftarrow i\_d([D], the) \wedge i\_n(N1, I\_N). \quad (I2)$$

$$i\_cp([cp/[C|IP]], I\_IP) \leftarrow i\_c([C], that) \wedge i\_ip(IP, I\_IP). \quad (I3)$$

$$i\_cp([\,], 0). \quad (I4)$$

$$i\_vp([vp/[V|A]], I\_VP) \leftarrow i\_v([V], is) \wedge i\_a(A, I\_VP). \quad (I5)$$

$$i\_d([d/[the]], the). \quad (I6)$$

$$i\_n([n/[joke]|CP], joke(Subj)) \leftarrow i\_cp(CP, Subj). \quad (I7)$$

$$i\_n([n/[argument]|CP], argument(Subj)) \leftarrow i\_cp(CP, Subj). \quad (I8)$$

$$i\_c([c/[that]], that). \quad (I9)$$

$$i\_v([v/[is]], is). \quad (I10)$$

$$i\_a([a/[funny]], funny). \quad (I11)$$

Rule (I1) says, for example, that the interpretation of an IP structure is $sat(Subj, Pred)$ where $Subj$ is the interpretation of the subject NP and $Pred$ is the interpretation of the main VP. Notice that the role of the interpretation rules for d, c, and v is really insignificant, since we are restricting ourselves to just one determiner, complementizer, and verb. This simplifies the example enormously. According to this theory, the tree displayed above has the logical form,

$$sat(argument(0), funny)),$$

and the following simple Prolog proof demonstrates this fact:

$$\begin{aligned}
& \leftarrow i_ip([ip/[np/[d/[the], n/[argument]], vp/[v/[is], \\
& \qquad a/[funny]]]], sat(argument(0), funny))) \\
by\ (I1): \quad & \leftarrow i_np([np/[d/[the], n/[argument]]], argument(0)) \wedge \\
& \qquad i_vp([vp/[v/[is], a/[funny]]]], funny) \\
by\ (I2): \quad & \leftarrow i_d([d/[the]], the), i_n([n/[argument]], argument(0)) \wedge \\
& \qquad i_vp([vp/[v/[is], a/[funny]]]], funny) \\
by\ (I6): \quad & \leftarrow i_n([n/[argument]], argument(0)) \wedge \\
& \qquad i_vp([vp/[v/[is], a/[funny]]]], funny) \\
by\ (I7): \quad & \leftarrow i_cp([\,], 0) \wedge i_vp([vp/[v/[is], a/[funny]]]], funny) \\
by\ (I4): \quad & \leftarrow i_vp([vp/[v/[is], a/[funny]]]], funny) \\
by\ (I5): \quad & \leftarrow i_v([v/[is]], is) \wedge i_a([a/[funny]], funny) \\
by\ (I10): \quad & \leftarrow i_a([a/[funny]], funny) \\
by\ (I11): \quad & \square\ Q.E.D.
\end{aligned}$$

3.3. *The World Model*

Given a logical form, a hearer may want to assess it with respect to knowledge of the world. What do the terms of the logical form denote, and are the predications true? It is a bit artificial to do this with such a simple language, but it may help illustrate the point that the system can not only parse and interpret on a word-by-word basis, but can also begin assessing the denotations of terms and the truth of predications. Steedman suggests that not only is interpretation incremental, but so is 'reference', in the sense that a hearer can begin making judgments about what in the world the interlocutor is talking about almost as soon as the words are uttered.

So let's assume that the joke that is relevant to this example is a strange logician's joke, a joke about an argument about itself—the sort of

joke that Raymond Smullyan might make. To be more precise, suppose that all of the jokes that can be mentioned in this language are the same. The joke, the joke about the argument, the joke about the argument about the joke, all of these are the same joke. Every noun phrase in our language denotes this same joke, and at LF, the denotation of joke(X) is this same joke for all X. And let's suppose that this joke is, in fact, funny.

In order to get some contrasting cases, let's make rather different assumptions about the arguments. Let's suppose that no two different noun phrases beginning with 'the argument...' refer to the same argument. Let 'the argument' denote a serious argument, an argument about arms control or something, and it is not at all funny. And let 'the argument that the argument is funny' be a different argument, but one that is similarly not funny. But let 'the argument that the joke is funny' be a funny argument about a joke. Notice that all of the complex argument noun phrases in our language are either of the form:

$[_{np}$*the argument that ... the argument is funny*$]$,

or of the form:

$[_{np}$*the argument that ... the joke is funny*$]$.

Let's suppose that all of the arguments denoted by noun phrases of the first form are funny, but none of the arguments denoted by noun phrases of the latter form are funny. If we call the deepest sentence (*ip*) in an noun phrase the 'kernel', then the rule can be intuitively expressed as follows: argument noun phrases with a joke in their kernel all denote funny arguments, argument noun phrases with an argument in their kernel all denote arguments that are not funny, and the argument noun phrase with no kernel denotes an argument that is not funny.

The following rules assign truth values to logical forms according to the scheme just described:

$$truth(sat(joke(X), funny), true). \qquad (M1)$$

$$truth(sat(argument(A), funny), Value) \leftarrow argument\_kernel(A, Value). \qquad (M2)$$

$$argument\_kernel(0, false). \qquad (M3)$$

$$argument\_kernel(joke(A), Value) \leftarrow joke\_kernel(A, Value). \qquad (M4)$$

$$argument\_kernel(argument(A), Value) \qquad (M5)$$
$$\leftarrow argument\_kernel(A, Value).$$
$$argument\_kernel(sat(A, X), Value) \qquad (M6)$$
$$\leftarrow argument\_kernel(A, Value).$$
$$joke\_kernel(0, true). \qquad (M7)$$
$$joke\_kernel(joke(A), Value) \leftarrow joke\_kernel(A, Value). \qquad (M8)$$
$$joke\_kernel(argument(A), Value) \qquad (M9)$$
$$\leftarrow argument\_kernel(A, Value).$$
$$joke\_kernel(sat(A, X), Value) \qquad (M10)$$
$$\leftarrow joke\_kernel(A, Value).$$

According to the interpretation described above, the logical form

$$sat(argument(0), funny))$$

expresses a false proposition, and (M1)–(M10) accordingly entail

$$truth(sat(argument(0), funny), false).$$

We can demonstrate this entailment with the following simple proof:

$$\leftarrow truth(sat(argument(0), funny), false)$$
$$by\ (M2) : \leftarrow argument\_kernel(0, false)$$
$$by\ (M3) : \square\ Q.E.D.$$

On the other hand,

$$sat(joke(sat(argument(0), funny)), funny)$$

expresses a true proposition, and (M1)–(M10) accordingly entail

$$truth(sat(joke(sat(argument(0), funny)), funny), true).$$

With this definition of a world model, we can let our system assess the truth value of the sentences of our little language.

3.4. *The Proof Strategy*

We can now try to prove from our theory that

$$\exists Tree, LF, Value\ ip([the, joke, is, funny], [\,], Tree) \wedge$$
$$i\_ip([Tree], LF) \wedge truth(LF, Value).$$

This can be done with a simple proof strategy that proves each conjunct in order, completing the proof of the first conjunct before starting on the proof of the second conjunct, and completing the second before starting the third. In effect, this simply amounts to doing the three proofs shown in the previous three sections one after the other, and this is essentially what Prolog does. However, we need not proceed in this way. At each step we are faced with proving a conjunction $p_1 \wedge p_2 \wedge \ldots p_n$, and, as even beginners in logic would expect, at each point in the proof, one can work on any one of the conjuncts. What is more remarkable is that the particular choice about the order in which steps are taken will not have any substantial effect on the length of the proof, even though it can have a dramatic influence on the difficulty of *finding* a proof, for a wide class of proof systems.[7] Computer scientists have been interested in this result because it allows us to choose a selection rule that is optimally efficient. Especially given the observations we made about alphabetizing lists, it is no surprise that optimal selection strategies are often nonpedestrian (Naish, 1987). It is easy to describe a nonpedestrian strategy for our simple parser.

In the notation we have used to represent our proofs, each line can be regarded as a conjunction of atomic predications, each atom of which must be eliminated by a resolution step. In all the proofs shown above, each step involves eliminating the leftmost atom. If we proceed in the same way given

$$\leftarrow ip([the, joke, is, funny], [\,], Tree) \wedge \\ i\_ip([Tree], LF) \wedge truth(LF, Value),$$

we end up with a pedestrian system. It is easy to define an alternative strategy, though, based on the idea that we should use the LF and world model predicates as soon as they receive the needed information from the parse predicates. Let's be more precise. Call all the predicates that occur in (M1)–(M10) the *model* predicates; call all the predicates that occur in (I1)–(I11) the *LF* predicates; and call all the predicates in (S1)–(S11) the *syntax predicates*. In the extremely modular theory we have presented, there is no overlap among these sets.[8] In the first line of our proof, then, we have one model predicate, one LF predicate, and one syntax predicate:

$$\leftarrow ip([the, joke, is, funny], [], Tree) \wedge \\ i\_ip([Tree], LF) \wedge truth(LF, Value).$$

It would be a mistake to begin the proof with the model or LF predications here because the arguments of these predicates do not yet contain any information about the input string. As the syntax predications are resolved upon, these arguments will become instantiated to nonvariable terms, and then they can be used. So, at each step of the proof, let's use the following rule:

> If there is one, choose the first model predication whose leftmost argument τ is not a variable and either $\tau = sat(t_1, t_2)$ where t_1 is a nonvariable term, or else $\tau \neq sat(t_1, t_2)$. Otherwise, if there is one, choose the first LF predication whose first argument is a (list containing a) tree structure whose leftmost leaf is not a variable. Otherwise choose the first predication.[9]

We are guaranteed that this selection rule gives us a sound and complete resolution system (since *every* selection rule does). The result in this case is a nonpedestrian language processing system: it develops its interpretation incrementally, beginning before phrases are complete, and yet it obviously respects the strong competence assumption. A brief consideration of a couple of examples illustrates the nonpedestrian online performance of this system.

3.5. *Runtime Analyses*

Consider the example,

> *the joke is funny.*

We can attempt to find a parse tree $Tree$, a logical form LF and a truth value $Value$ for this string by beginning a proof with the following line, in which we have underscored the predication that our selection rule tells us to choose:

$$\leftarrow \underline{ip([the, joke, is, funny], [], Tree)} \wedge i\_ip([Tree], LF) \wedge truth(LF, Value).$$

The first argument of the model predication is a variable, so we cannot work on it. The first argument of the LF predication is also a variable, so we cannot use it either. Subjecting each step to such consideration yields the 17-step proof shown in appendix B. The point to note, which should be obvious even without consulting the appendix, is

that we interpret each word as soon as it is placed into the parse tree. In the case of *the joke is funny*, this means that the system, in effect, figures out what you are talking about as soon as it sees the noun *joke*, and since there is only one predicate, the system immediately and justifiably assumes that the sentence is true at this point, on condition that the remainder of the string satisfies the syntactic and LF constraints. (In the appendix it is easy to see that the truth assessment is begun and completed in the 8th step, before the LF analysis of the initial noun phrase is complete, and before the syntactic analysis of the verb phrase has even begun.)

The same proof strategy applied to the example,

> *the argument that the joke that the argument is funny is funny is funny*

yields an interestingly different proof. In particular, although the LF and referential analysis is still done on a word-by-word basis, we cannot know what the denotation of the initial noun phrase is until we have seen all of it. Accordingly, although our model is constructing its representation of the denotation of the initial noun phrase on a word-by-word basis, that representation is not complete until the whole noun phrase has been completely analyzed at the syntactic and LF levels, and consequently the truth assessment for the sentence is delayed until the whole noun phrase is processed. Since we have presented our processing model in full detail, the reader can easily explore its behavior on a range of other examples. Interpretation is incremental in all of them. In some cases, the proof system can make incorrect guesses about syntactic structure and interpretation (and undo them by backtracking, or abandon them if they are found in parallel).

4. INCREMENTAL INTERPRETATION: COMPLEX LANGUAGES AND BOTTOM-UP PARSERS

Stepping back from the details, how is the incremental parsing achieved in the example of the previous section? One way of looking at the matter is this. A simple proof strategy finds a parse tree in standard top-down left-to-right order, and builds the corresponding parts of the LF form in the same order. For example, the string *the argument is funny* has the parse tree shown in figure 2, and the LF representation can similarly be represented as a kind of tree, as in figure 3.[10]

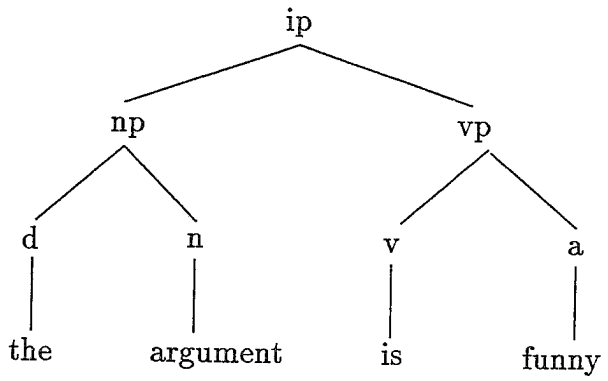


Figure 2: The parse tree for the sentence *the argument is funny*.

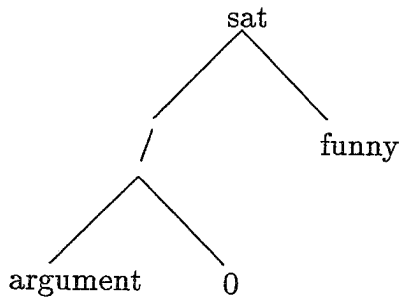


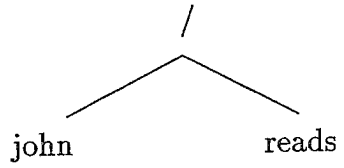
Figure 3: The corresponding LF for the sentence *the argument is funny*.

The property of these representations that was exploited was simply that as the syntactic tree was built from left-to-right on a word by word basis, we were able to build the LF 'tree' from left-to-right. This observation might raise some worries, though, about whether an approach of this kind could be extended to more complex languages, and about whether it could be extended to bottom-up parsing strategies. Let's very briefly consider each of these in turn. The point is not to propose any particular theory of human language processing, but simply to show that a wide variety of processing strategies are compatible with incremental interpretation.

4.1. *Extending the Semantics (Slightly)*

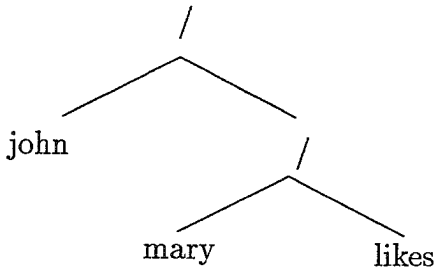
A full syntax and semantics for natural languages is fortunately beyond the scope of this brief note, but one might think that even in very simple fragments of English, the strategy used in our little example will not work. Let's consider just one simple problem which might occur to a semanticist from the generalized quantifier tradition. Suppose that we regard NPs as sets of properties, or, equivalently as functions from properties to truth values. Then a simple sentence like *john reads* might be represented as $john(reads)$, that is, the value of the function corresponding to *john* applied to the property corresponding to *reads*. It is clear that we could begin to construct the LF form $john(reads)$ as we constructed the parse tree for the sentence *john reads*. Similarly, we could construct the form $some(man)(reads)$ as we built the parse tree for *some man reads*, indicating that *some* maps the property *man* into a function which maps the property *reads* to a truth value. However, in most semantic theories that use LF representations like these, the functors are not always assumed to precede their arguments in the syntax. For example, a nominal PP might be interpreted as a function that maps the property corresponding the the head noun into another property. Then the meaning of *some man with some hat reads* might be represented as $some(with(some(hat))(man))(reads)$. Or if we regard direct objects as functions that map binary relations into properties then *john likes mary* might be represented as $john(mary(likes))$ (cf. Keenan, 1989; Keenan and Faltz, 1985).

These last two logical forms present a problem: they cannot be incrementally specified from left-to-right as the parse tree is constructed because the parse tree sometimes specifies arguments before it specifies the corresponding functors. The problem is perhaps clearer if we regard the LF forms as trees. There is no problem with incrementally building the LF form of *john reads* as the parse is built:



As soon as the first leaf of the parse tree is specified, the first leaf of this LF tree can be specified. With the sentence *john likes mary*, though, the

relation between the parse tree and the LF form is more complicated:



The problem is that, if we build our tree from left-to-right, we will be unable to place the property name corresponding to the verb into this LF until the direct object is fully parsed, so we do not obtain incremental interpretation.

Once stated, though, it is easy to see that this problem is just an artifact of an inessential aspect of the LF notation. The inessential aspect is simply the convention that functors are specified before their arguments. The problem does not arise if we use a slightly different notation. Suppose we adopt a bit of notation from categorial grammar, and use the forward slash to indicate the application of a functor to its argument (as already done in the LF trees displayed above),

$$functor/argument$$

and a backward slash to indicate the provision of an argument to a functor,

$$argument\backslash functor.$$

Then the LF of *john likes mary* is $john/(likes\backslash mary)$, and the LF of *some man with some hat reads* is

$$(some/(man\backslash(with/(some/hat))))/reads.$$

These LF trees have the property that their n'th leaf can be specified as soon as the n'th leaf of the parse tree is specified.

4.2. *Incremental Interpretation with Bottom-Up Parsing*

The incremental deductive processor described in section 3 is top-down. Abney (1989, p. 130) has proposed that although top-down parsing can

allow incremental interpretation, this is not possible with bottom-up parsing:

> ...LR parsers cannot model the incrementality of human parsing. Namely, in right-branching structures, LR parsers build no structure until all the input has been read...By contrast, people clearly build structure before the end of the sentence and pass it incrementally to the semantic processor.

The idea here is similar to the one in the previous section. If we use a bottom-up LR parser on the string *the joke is funny*, structure is built by 'reduce' operations, and these precede from left-to-right and bottom-up. That is, the parser first discovers that d dominates *the*, then that n dominates *joke*, then that np dominates the previously formed d, n, and so on. Abney's suggestion is that this makes incremental interpretation of the sentence impossible, since the parser does not even begin to build the tree for ip until the whole string has been read.

After the previous section, it is easy to see that this problem is similarly an artifact of inessential formal details. The problem depends on the assumption that the only syntactic structure that can be interpreted is one that has the root ip, but this assumption is mistaken. Notice that our notation for trees in previous sections is top-down oriented in the sense that the root is specified first:

$$ip/[np/[d/[the], n/[joke]], vp/[v/[is], a/[funny]]],$$

but there is no need to specify trees in this form. We can just as well specify the very same trees with their components in the order that they are found by an LR parser:

$$[d/[the], n/[joke], np/[D, N], v/[is], a/[funny], vp/[V, A], \\ ip/[NP, VP]].$$

Under the obvious interpretation, this latter representation defines exactly the same tree as the former, and so it serves perfectly well as the output of a syntactic processor.

Notice that the LF theory of section 3.2 specifies interpretations for every syntactic constituent, but it uses a top-down oriented notation for both the syntactic constituents and for the LF forms. Given the new notation for our syntactic structures, it is natural to use a bottom-up notation for our LF representations as well. Instead of using the form

$sat(joke(0), funny)$

let's use

$[joke, 0, INP/ICP, funny, sat(INP, IVP)]$.

These LF representations have the desired relation to the LR syntactic representations: the order of elements in LF is exactly the same as the order of the corresponding elements in the syntactic representation. Consequently, it is clear that a proof strategy that does truth assessment and LF interpretation as soon as the requisite parts of the syntactic form are specified will provide incremental interpretation. The theories of syntax, LF, and the world model remain completely modular, and the syntax remains right-branching. The only change is the use of a new tree notation in both the syntax and the LF, and the use of a different parsing strategy.[11] In sum, LR parsing is perfectly compatible with incremental interpretation

5. PREVIOUS REJECTIONS OF THE PEDESTRIAN ASSUMPTION

Various versions of the pedestrian hypothesis have been embraced and rejected before this response to Steedman (1989) and Abney (1989). In response to Tyler and Marslen-Wilson (1977) and Tyler (1980), Berwick and Weinberg (1983, pp. 7–8) make the point very clearly:

> The logic of Tyler and Marslen-Wilson's argument must assume that a grammar adhering to the autonomy of syntax thesis is compatible with one and only one derivational model (the model of noninteracting components usually associated with the *logical* organization of a [transformational grammar]) ... We will show that even under [the strong competence hypothesis], the autonomy thesis says nothing about the flow of information between components. Rather, it spells out the types of information that can form the representations of each component.[12]

Although the possibility of beginning the interpretation of partially built phrases was not explicitly considered by Berwick and Weinberg, the basic idea in this passage is essentially the same.[13] As noted above, once we grant that nothing blocks the interpretation of subsentential

constituents, the same reasoning shows that nothing blocks the interpretation of subphrasal constituents, even in models that conform to the strong competence hypothesis. Under the strong competence hypothesis, linguistic theory places substantial constraints on performance models all right, but it still leaves the whole range of purely procedural issues open. A logical formulation of grammar makes this point obvious: the steps of parsing and interpretation can be interleaved and structured in any way.

To use processing data to support claims about the grammar, psychologists are faced with the prospect of sorting through the large range of possible processing assumptions. As J.D. Fodor (1988; 1991 forthcoming) has pointed out, this makes good arguments from on-line performance to the grammar rather hard to come by, and in fact she fingers the nonpedestrian parsing possibilities with modular grammars as one of the main obstacles:

> Two distinct components of the grammar might be integrated before or during parsing, so that the absence of a processing distinction does not entail the absence of a distinction in the grammar. (Fodor, 1988, p. 127)

This appears to be our rejection of the pedestrian assumption in yet another guise. However, although Fodor rejects the pedestrian's argument from distinctness in the grammar to distinctness in processing, she accepts the converse argument:

> ...evidence for nonsynchronous application during sentence parsing of two (or more) different kinds of linguistic information constitutes indirect evidence for modularity within the mental grammar. If two kinds of information are integrated in the knowledge base, it is difficult to see how they could be systematically separated by the parser and applied at different times to an input sentence. Hence we can reasonably infer a grammatical distinction from a processing distinction. (Fodor, 1988, p. 127)

Fodor (1988) quickly qualifies this claim,[14] but let's briefly consider why some qualification is necessary here. In the first place, there is some vagueness here about what counts as a distinction of the relevant sort. In the little example of the previous section, there were three autonomous modules that had no predicate in common: these were called

the syntax, the semantics and the world model. Does this organization rule out processing models which apply different parts of one module at different times? Certainly not: we can and do apply information about the immediate constituents of noun phrases at one point, delaying other syntactic information such as the constituency of verb phrases, but these two parts of the syntax are in the same module and, in a clear sense, cannot be separated. That is, noun phrases can include verb phrases, and conversely, and so our account of the syntax of one necessarily involves the syntax of the other. This *linguistic argument* for a modularity claim is very strong, but does not begin to suggest that all syntactic information must be applied at once. This point applies quite generally: whenever one module, one rule or one principle, tells you two things, in a sense that allows those two things to be checked at different times, we can design a processing system that does check those two things at different times. The point can be put more abstractly: if we represent ϕ and ψ with a single representation γ, we can still apply ϕ at a different time from ψ in a processing task if we can (easily) compute both ϕ and ψ from γ. Consequently, it seems that processing distinctions do not imply grammatical distinctions in any interesting sense.

Fodor (1988, 1991 forthcoming) considers the following argument (among others). Suppose that we discovered evidence that subcategorization constraints associated with lexical items are not enforced as early as possible in human sentence processing. (For present purposes, the question of whether there really is such evidence is not relevant; we are interested in the significance that evidence of this kind would have.) It is suggested that this would be relatively difficult to reconcile with Government Binding (GB) theories in which phrase structure rules have been eliminated because of their redundancy with lexical subcategorization and other parts of the grammar. Apparently the idea is that, in the absence of phrase structure rules, a transparent GB parser would use lexical subcategorization to guide tree construction in parsing, and consequently it is hard to make sense of delaying this information. Fodor suggests that the delay of subcategorization information might be more easily handled in realizations of a certain kind of GPSG grammar, one in which phrase structure rules are separate from lexical subcategorization information.[15]

We can now see that this is a pedestrian assumption. There is no reason at all to suppose that a parser should use all lexical information at once: it is perfectly natural to suppose that certain key structural

information (the principal syntactic category of the words specified in the lexicon, together with $\overline{X}$ restrictions, etc.) is used first, leaving subcategorization information (also specified in the lexicon) and other less central information to later. Not only is a nonpedestrian processing story possible in this case, but it is very natural and straightforward. The reason for assuming that various kinds of information are represented in the lexicon is that the information is lexical-item-specific. From a linguistic perspective, it is natural to group item-specific information together in the lexicon, even if it rather diverse. From a computational perspective, it is also natural to suppose that the lexicon is accessed (or perhaps, activated) once, to provide a variety of item-specific information. Evidence that subcategorization information is not applied immediately does not disconfirm these accounts.

This is not to say that psychological evidence about what information is used when could never have any bearing at all on questions about what grammar is being used. It is perhaps possible (though it seems very unlikely to happen) that we will find two quite different grammars such that each has strong and roughly equal empirical support, but where only one of them represents some particular aspect of the language in a distinct module with distinct principles. In this case, if psychological evidence showed that information about this aspect of the language has a distinctive role in processing, then this might provide some weak support for the grammar that segregates the information. The evidence would be weak because there could well be an independently motivated nonpedestrian processing story to explain the psychological evidence for the nonmodular account. And the situation seems unlikely to arise in any case because it is hard to imagine that there would not be more direct linguistic arguments to address such a basic difference between two such grammars.

The psychological evidence about when in a processing task each type of information is used tells us just that. This by itself is something we want to know! How is the immense wealth of information in a grammar and lexicon brought to bear in linguistic tasks? The psychological studies mentioned, and many others, show that this problem can be addressed even before the linguists have settled all disputes about what information, exactly, is in the grammar and lexicon, and in what form. There is some consensus among linguists about what information must be there, and that there is lots of it, even though none of the prominent grammars comes very close to getting everything right for every

language (or even for a single language, even at the most superficial, 'descriptive' levels of analysis). Even if significant headway is made on what information is used when, I think it is a mistake to expect that this will tell us very much about how that information must be represented. Although there are certainly inappropriate representations for computational problems, typically the range of suitable representations is quite wide.[16]

However, as discussed above, the question of how and when each piece of information is deployed can have a dramatic effect on performance. This is again just another formulation of our rejection of the pedestrian assumption. To sort a longish list, we really must start using our ordering information while we are still using the information about what constitutes a permutation. In our little deductive parser, or in any similar deductive reasoning, the best strategies reduce the needed search by (roughly) selecting subproblems with relatively small search spaces at each step. From any reasonable perspective, the study of the order in which subproblems are tackled in human language processing is of more direct and immediate interest than such questions as whether the some propositions ϕ and ψ are represented by a single representation in a single part of the grammar or with distinct representations possibly in distinct parts of the grammar.

6. CONCLUSION

Natural processing models can provide incremental left-to-right analysis and interpretation of left-branching structures. The more general and important point is that even in performance models that have no other information about a language than the grammar provides, even in models that assign structure strictly according to deductive reasoning from the grammar, there is a huge range of processing options. There are many possible ways to bring the grammatical constraints to bear in any particular linguistic task. Getting reaction time studies to bear on the grammar is consequently quite difficult in our present state of ignorance about language processing. For example, if psychologists discover that humans initially assign incorrect structures to certain locally ambiguous constructions, there are other options to consider than the common idea that extra-grammatical heuristic principles are being used to assign structure. The evidence might instead reflect on the way that the space of grammatical structures is being explored on the basis of the local,

incomplete information.[17]

However, far from removing the interest of results about what is used when, the rejection of the Steedman's paradox comes with the insight that the order of application of linguistic information, though not determined by the grammar, is fundamental to finding a feasible performance model. The other fundamental requirement for a descriptively adequate model is, of course, that we get a correct account of what the linguistic information is, a correct grammar, but reaction time studies are not the most direct approach to this problem.

APPENDIX A: A PROLOG IMPLEMENTATION OF THE NONPEDESTRIAN PARSER

To get a complete Prolog implementation of the nonpedestrian language processor described above, most Prologs will require only a minor change in the notation of (S1)–(S11), (I1)–(I11), and (M1)–(M10). Simply change $\leftarrow$ to $:-$, and change $\wedge$ to a comma. These axioms then can be used by the *demonstrate* predicate given here. For further discussion of proof predicates like *demonstrate* see Pereira and Shieber (1987) and Stabler (1990).

```
%%% the non-pedestrian metainterpreter: demonstrate(Goal)
% For example, run
% ?- demonstrate(s([the,joke,is,funny],[],Tree))
% All demonstrate proofs use the following rule to select the next step:
% If the first model literal has a first arg with a nonvar leftmost leaf,
%         select it; otherwise
% If the first lf literal has a first arg with a nonvar leftmost leaf,
%         select it;
% otherwise select the first literal

demonstrate([]) :- !. % nothing left to prove - finished!
demonstrate(Goal) :- % otherwise select next literal L from Goal...
% nl,itemize(Goal),nl, % for tracing only
( select(Position,Goal,[L],Rem),model(L),leaved(L) -> true
; select(Position,Goal,[L],Rem),lf(L),leaved(L) -> true
; Position=s(0), select(Position,Goal,[L],Rem)
),
% write('Next literal: '),write(Position),write('.'),nl,
% for tracing
clause(L,Body),
and_to_list(Body,NewLiterals),
select(Position,NewGoal,NewLiterals,Rem),
```

```
demonstrate(NewGoal).

% specify the lf and model predicates:
lf(i_ip(_,_)). lf(i_np(_,_)). lf(i_d(_,_)). lf(i_n(_,_)).
lf(i_cp(_,_)). lf(i_vp(_,_)).
lf(i_v(_,_)). lf(i_a(_,_)).   lf(i_c(_,_)).

model(truth(_,_)). model(argument_kernel(_,_)).
                        model(joke_kernel(_,_)).

% check if leftmost leaf of term is not a variable
leaved(Arg) :- atomic(Arg), !.
leaved(_/L) :- !, leaved(L). % to check left branch of trees
leaved(joke(_)) :- !. % a special leaf for model preds
leaved(argument(_)) :- !. % a special leaf for model preds
leaved(Term) :- arg(1,Term,Arg),nonvar(Arg),leaved(Arg).
                                        % check leftmost arg of other
                                        % terms

% to avoid arithmetic probs, we use s(0)=1, s(s(0))=2, s(s(s(0)))=3,...
select(s(0),Result,Insert,Remainder) :- append(Insert,Remainder,Result).
select(s(Position),[H|Result],Insert,[H|T])
                        :- select(Position,Result,Insert,T).

append([],L,L).
append([H|L1],L2,[H|L3]) :- append(L1,L2,L3).

% convert a conjunction of goals to a list of goals
and_to_list((L,Ls),[L|Rest]) :- !, and_to_list(Ls,Rest).
and_to_list(true,[]) :- !.
and_to_list(L,[L]).

% itemize is just for tracing display
itemize([],_).
itemize([H|T],N) :- M is N+1,write(M),tab(3),write(H),nl,itemize(T,M).
```

APPENDIX B. A NONPEDESTRIAN PROOF

Using the selection rule of appendix A, a rule that always prefers model predicates to LF predicates to syntactic predicates, but can only evaluate predicates whose leftmost argument is sufficiently instantiated, we get the following *incremental, word-by-word* proof that *the joke is funny* has a parse tree IP, a logical form LF and a truth value $Value$:

$\leftarrow \underline{ip([the, joke, is, funny], [\,], IP)}\wedge$
$i\_ip([IP], LF)\wedge$
$truth(LF, Value).$

$by(S1):$

$\leftarrow \underline{np([the, joke, is, funny], S1, NP)}\wedge$
$vp(S1, [\,], VP)\wedge$
$i\_ip([ip/[NP, VP]], LF)\wedge$
$truth(LF, Value).$

$by(S3):$

$\leftarrow \underline{d([the, joke, is, funny], S2, D)}\wedge$
$n(S2, S1, N)\wedge$
$vp(S1, [\,], VP)\wedge$
$i\_ip([ip/[np/[D, N], VP]], LF)\wedge$
$truth(LF, Value).$

$by(S2):$

$\leftarrow n([joke, is, funny], S1, N)\wedge$
$vp(S1, [\,], VP)\wedge$
$\underline{i\_ip([ip/[np/[d/[the], N], VP]], LF)}\wedge$
$truth(LF, Value).$

$by(I1):$

$\leftarrow n([joke, is, funny], S1, N)\wedge$
$vp(S1, [\,], VP)\wedge$
$\underline{i\_np([np/[d/[the], N]], Subj)}\wedge$
$i\_vp([VP], Pred)\wedge$
$truth(sat(Subj, Pred), Value).$

by $(I2)$:

$$\leftarrow n([joke, is, funny], S1, N)\wedge$$
$$vp(S1, [\,], VP)\wedge$$
$$\underline{i\_d([d/[the]], the)\wedge}$$
$$i\_n([N], Subj)\wedge$$
$$i\_vp([VP], Pred)\wedge$$
$$truth(sat(Subj, Pred), Value).$$

by $(I6)$:

$$\leftarrow \underline{n([joke, is, funny], S1, N)\wedge}$$
$$vp(S1, [\,], VP)\wedge$$
$$i\_n([N], Subj)\wedge$$
$$i\_vp([VP], Pred)\wedge$$
$$truth(sat(Subj, Pred), Value).$$

by $(S7)$:

$$\leftarrow vp([is, funny], [\,], VP)\wedge$$
$$\underline{i\_n([n/[joke]], Subj)\wedge}$$
$$i\_vp([VP], Pred)\wedge$$
$$truth(sat(Subj, Pred), Value).$$

by $(I7)$:

$$\leftarrow vp([is, funny], [\,], VP)\wedge$$
$$i\_cp([\,], X)\wedge$$
$$i\_vp([VP], Pred)\wedge$$
$$\underline{truth(sat(joke(X), Pred), Value).}$$

by $(M1)$:

$$\leftarrow vp([is, funny], [\,], VP)\wedge$$
$$\underline{i\_cp([\,], X)\wedge}$$
$$i\_vp([VP], funny).$$

by $(I4)$:

$\leftarrow \underline{vp([is, funny], [\,], VP)}\wedge$
$i\_vp([VP], funny).$

by $(S6)$:

$\leftarrow \underline{v([is, funny], S3, V)}\wedge$
$a(S3, [\,], A)\wedge$
$i\_vp([vp/[V, A]], funny).$

by $(S10)$:

$\leftarrow a([funny], [\,], A)\wedge$
$\underline{i\_vp([vp/[v/[is], A]], funny)}.$

by$(I5)$:

$\leftarrow a([funny], [\,], A)\wedge$
$\underline{i\_v([v/[is]], is)}\wedge$
$i\_a([A], funny).$

by$(I10)$:

$\leftarrow \underline{a([funny], [\,], A)}\wedge$
$i\_a([A], funny).$

by $(S11)$:

$\leftarrow \underline{i\_a([a/[funny]], funny)}.$

by $(I11)$:

$\square$ *Q.E.D.*

$IP = ip/[np/[d/[the], n/[joke]], vp/[v/[is], a/[funny]]]$
$LF = sat(joke(0), funny)$
$Value = true$

APPENDIX C. AN INCREMENTAL LR PARSER

In section 3 a logical representation of the following grammar was used:

(1) ip → np vp

(2) np → d n

(3) np → d n cp

(4) cp → c ip

(5) vp → v a

(6) d → the

(7) n → joke

(8) n → argument

(9) c → that

(10) v → is

(11) a → funny

There is a mechanical method for constructing the LR table for such a grammar. As usual, $ is the end-of-string symbol. (See, *e.g.*, Aho and Ullman (1977) for a description of these tables and their use.) With the simple grammar above, we get the following table:

state	$	argument	funny	is	joke	that	the
0							sh5
1							re9
2				re2		sh1	
3		sh10			sh9		
4							sh5
5		re6			re6		
6				sh11			
7				re4			
8				re3			
9				re7		re7	
10				re8		re8	
11			re10				
12	re11			re11			
13			sh12				
14	re5			re5			
15	re1			re1			
16	acc						

State	a	c	cp	d	ip	n	np	v	vp
0				3	16		6		
1									
2			4	8					
3						2			
4				3	7		6		
5									
6								13	15
7									
8									
9									
10									
11									
12									
13	14								
14									
15									
16									

We always begin in state 0, and then take the action specified for our current state and the next input symbol. For example, beginning in state 0 with *the* as the next word of the input, we see that the prescribe action is *sh*5, *i.e.*, shift the next word of the input into the stack and go into state 5. In state 5 with *joke* as the next word of the input, the prescribed action is *re*6, reduce using production 6. This reduce action pops *the* off the stack, taking us back to state 0. So now we are in state

0 and about to put push a *d* onto the stack. Using the far right of the table, we see that from state 0, on creation of a *d*, we go to state 3. We continue in this way until we get to the action *acc*, accepting the string, or else we get to an error condition in which no action is prescribed.

The steps in parsing and incrementally interpreting *the joke is funny* with this LR table are as on the following page.

stack: 0
input: the joke is funny
action: sh5
tree:
lf:
truth:

stack: 5 the 0
input: joke is funny
action: re6
tree:
lf:
truth:

stack: 3 d 0
input: joke is funny
action: sh9
tree: d/[the]
lf:
truth:

stack: 9 joke 3 d 0
input: is funny
action: re7
tree: d/[the]
lf:
truth:

stack: 2 n 3 d 0
input: is funny
action: re2
tree: d/[the] n/[joke]
lf: joke
truth:

stack: 6 np 0
input: is funny
action: sh11
tree: d/[the] n/[joke] np/[A,B]
lf: joke 0 A/B
truth: true

stack: 11 is 6 np 0
input: funny
action: re10
tree: d/[the] n/[joke] np/[A,B]
lf: joke 0 A/B
truth: true

stack: 13 v 6 np 0
input: funny
action: sh12
tree: d/[the] n/[joke] np/[A,B] v/[is]
lf: joke 0 A/B
truth: true

stack: 12 funny 13 v 6 np 0
input: $
action: re11
tree: d/[the] n/[joke] np/[A,B] v/[is]
lf: joke 0 A/B
truth: true

stack: 14 a 13 v 6 np 0
input: $
action: re5
tree: d/[the] n/[joke] np/[A,B] v/[is]
a/[funny]
lf: joke 0 A/B funny
truth: true

stack: 15 vp 6 np 0
input: $
action: re1
tree: d/[the] n/[joke] np/[A,B] v/[is]
a/[funny] vp/[C,D]
lf: joke 0 A/B funny
truth: true

stack: 16 ip 0
input: $
action: acc
tree: d/[the] n/[joke] np/[A,B] v/[is]
a/[funny] vp/[C,D] ip/[E,F]
lf: joke 0 A/B funny sat(C,D)
truth: true

ACKNOWLEDGEMENTS

The basic idea of this chapter is more carefully developed and applied to natural language processing in Stabler (1991 forthcoming). I am grateful to Robert Berwick, Janet Dean Fodor, Ed Keenan, and Mark Johnson for helpful discussions of this material.

NOTES

[1] It is not perfectly clear what counts as the direct use of a grammar by a processor, but this vagueness will not obscure the main point of this discussion. I will describe some examples that respect the strong competence hypothesis, but I think that my examples will all be uncontroversial. They directly use explicit and complete representations of the grammar, with no other structure-defining principles. Cf. Matthews (1988), Stabler (1984), Berwick and Weinberg (1984), Stabler (1983), Bresnan and Kaplan (1982) for discussions about what strong competence hypothesis we need, exactly, to obtain the desirable compatibility with the competence and learning theory.

[2] Obviously, the conclusion that you are using a nonpedestrian technique does not follow if you are not doing the task. In particular, the conclusion can be drawn about the human parser only on the assumption that humans do, in fact, resolve all the structural ambiguities in such cases, computing a single, definite syntactic structure.

[3] It is puzzling that Steedman does not notice this point, that once we allow interpretation to begin before the sentence is complete (as is done in Winograd (1972) and Bobrow and Webber (1980), cited by Steedman), we can just as well allow interpretation to begin before any phrase is complete.

[4] This isomorphism between context-free derivations and SLD-resolution refutations was implicit in VanderBrug and Minker (1975), Pereira and Warren (1980) and following work, but has been made completely explicit in Stabler (1991 forthcoming).

[5] Strictly speaking, this, and the other proofs shown below, are refutations, proofs by contradiction. Furthermore, all the proofs use the single inference rule: SLD resolution. See Lloyd (1987) for a presentation of this proof technique. It is sound and complete for the theories considered here, since they are all 'Horn theories'. That is, they have a special syntactic property which allows the use of especially simple proof methods. Stabler (1991 forthcoming) points out that most linguistic theories are not Horn, and consequently more powerful proof methods must be used.

[6] Although my strategy here is, I hope, intuitively clear, a logician will notice some tricky business, tricky but perfectly sound and natural. In the LF language, I am using *sat* as both a predicate and as a function. There are no other LF predicates, and the other LF functions and constants are $joke, argument, 0$ and $funny$. However, the LF language is itself an object language relative to the language of the following axioms. That is, the following axioms use terms identical to the expressions of the LF language to denote expressions of the LF language, in order to define the proper relation between syntactic trees and LF expressions.

[7] The completeness of particular selection functions in linear resolution and similar methods was established independently by several researchers, including Reiter

(1971); Kowalski and Kuehner (1971); and Loveland (1972). Completeness for *arbitrary* selection functions was established by Reiter and by Kowalski and Kuehner. The results on the lengths of the shortest proofs in these systems is explored by Kowalski and Kuehner. The result most directly relevant to Prolog is elegantly presented in a recent text by Lloyd (1987) as the 'independence of the computation rule' for Horn clause SLD-resolution. The origin of the result for SLD resolution is not discussed by Lloyd, but Kowalski (1979, p. 71) credits Edinburgh University technical memos by F.M. Brown (1973) and R. Hill (1974).

[8] Note that the modularity I mention here is not a modularity in processing, an 'informational encapsulation' of the sort that is often discussed in the psycholinguistic literature. I do not like calling it a 'representational modularity' either, because it is a property of the language rather than a property of our representation of the language. That is, the language itself is such that we can easily define its syntactic properties without reference to LF, and LF without reference to syntax. One can imagine languages (perhaps English) in which it is difficult or impossible to completely disentangle the specifications of different kinds of properties.

[9] A complete Prolog implementation of a proof predicate that uses this proof strategy can be defined with only a few lines. One is presented in appendix A.

[10] We use the forward slash / as a node label in this tree simply to indicate that *argument*(0) is to be constructed from the two pieces *argument* and 0. This use is different from our previous use of the slash in terms denoting trees. The motivation for this notation will become clear in the following paragraphs.

[11] For readers who want to see a few more of the computational details, see appendix C.

[12] I have replaced Berwick and Weinberg's abbreviation 'TG' by '[transformational grammar]', and their "such Type Transparency assumptions" by our term "[the strong competence hypothesis]".

[13] Berwick and Weinberg have been criticized for their claim that their particular parsing model is a transparent realization of a transformational grammar (Stabler, 1984; J.D. Fodor, 1985). But even if these critiques are sound (and their soundness is controversial: Berwick and Weinberg, 1985a; 1985b), the main point of the quoted passage remains untouched.

[14] Fodor (1988, p. 131) says "Other implementations are certainly imaginable... It does seem justified, as an initial step, to assume the most straightforward implementation of each type of grammar, and on that basis I think it can be agreed that evidence of nonsynchronous application of ... two kinds of information would be easier for [a theory with corresponding modules] to accommodate than for [a theory that does not separated the two kinds of information]." Similar remarks are made in Fodor (1989). My points are, first, that pedestrian implementations are *rarely* the most straightforward implementations, if they are feasible at all, and second, that better evidence for or against modularizing a particular aspect of the language is usually available.

[15] Fodor (1988, 1991, forthcoming); Clifton and Frazier (1986); Frazier, Clifton, and Randall (1983); and many others use essentially similar arguments in in application to many other points. Evidence that control information is delayed in parsing (were there any) is supposed to favor theories that segregate principles of control from

phrase structure and movement relations, as both GB and GPSG do. Evidence that movement relations (and such things as subjacency enforcement) are handled differently from other phrase structure constraints is supposed to favor theories like GB over theories like GPSG in which movement relations are handled with special features.

[16] Computer scientists sometimes express this by saying that basic complexity results remain invariant under any of the 'standard encodings' of a problem. A standard encoding does not have excessive amounts of redundancy or 'padding'; it does not keep most frequently needed information in a structure from which it can be derived only by extensive computation; etc.

[17] In our example, a deductive system could perfectly well explore the possibility that an NP has the structure D N before exploring the possibility that the structure is D N CP. This kind of behavior is perfectly familiar from the simplest context free parsing strategies, and does not reflect the use of any extragrammatical structure-defining principles.

REFERENCES

Abney, S.: 1989, 'A Computational Model of Human Parsing', *Journal of Psycholinguistic Research* **18**, 129–144.

Aho, A. and J. Ullman: 1977, *Principles of Compiler Design*, Addison-Wesley, Menlo Park, California.

Berwick, R. and A. Weinberg: 1983, 'The Role of Grammars in Models of Language Use', *Cognition* **13**, 1–62.

Berwick, R. and A. Weinberg: 1984, *The Grammatical Basis of Linguistic Performance: Language Use and Acquisition*, MIT Press, Cambridge, Massachusetts.

Berwick, R. and A. Weinberg: 1985a, 'Deterministic Parsing and Linguistic Explanation', *Language and Cognitive Processes* **1**, 109–134.

Berwick, R. and A. Weinberg: 1985b, 'The Psychological Relevance of Transformational Grammar: A Reply to Stabler', *Cognition* **19**, 193–204.

Bobrow, R. and B. Webber: 1980, 'Knowledge Representation for Syntactic/Semantic Processing', *Proceedings of the First Annual Conference on Artificial Intelligence*, Morgan-Kaufmann Publishers, Los Altos, California, pp. 316–323.

Bresnan, J. and R. Kaplan: 1982, 'Introduction: Grammars as Mental Representations of Language', in J. Bresnan (ed.), *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, Massachusetts, pp. xvii–lii.

Church, K. and R. Patil: 1982, 'Coping with Syntactic Ambiguity', *American Journal of Computational Linguistics* **8**, 139–149.

Clifton, C. and L. Frazier: 1986, 'The Use of Syntactic Information in Filling Gaps', *Journal of Psycholinguistic Research* **15**, 209–224.

Fodor, J.D.: 1988, 'On Modularity in Syntactic Processing', *Journal of Psycholinguistic Research* **17**, 125–168.

Fodor, J.D.: 1991 forthcoming, 'Sentence Processing and Mental Grammar', in P. Sells, S. Shieber, and T. Wasow (eds.), *Foundational Issues in Natural Language Processing*, MIT Press, Cambridge, Massachusetts.

Frazier, L., C. Clifton, and J. Randall: 1983, 'Filling Gaps: Decision Principles and Structure in Sentence Comprehension', *Cognition* **13**, 187–222.

Keenan, E.: 1989, 'Semantic Case Theory', In R. Bartsch, J. van Bentham, and R. van Emde-Boas (eds.), *Semantics and Contextual Expression*, Dordrecht, Foris, Groningen-Amsterdam Studies in Semantics (GRASS) Volume 11, pp. 33–57.

Keenan, E. and L. Faltz: 1985, *Boolean Semantics for Natural Language*, Reidel, Boston, Massachusetts.

Knuth, D.: 1973, *The Art of Computer Programming, Volume 1: Fundamental Algorithms*, Addison-Wesley, Reading, Massachusetts.

Kowalski, R.: 1979, *Logic for Problem Solving*, North-Holland, New York.

Kowalski, R. and D. Kuehner: 1971, 'Linear Resolution with Selection Functions', *Artificial Intelligence* **2**, 227–260.

Langendoen, T., A. Koslow, and C. Neff: 1988, 'The Syntax and Semantics of Coordinate Compounding in English', unpublished manuscript, City University of New York, New York.

Lloyd, J.: 1987, *Foundations of Logic Programming, Second Edition* Springer-Verlag, New York.

Loveland, D.: 1972, 'A Unifying View of Some Linear Herbrand Procedures', *Journal of the Association for Computing Machinery* **19**, 366–384.

Marslen-Wilson, W. and L.K. Tyler: 1980, 'The Temporal Structure of Spoken Language Understanding', *Cognition* **8**, 1–21.

Marslen-Wilson, W.: 1987, 'Functional Parallelism and Spoken Word Recognition', *Cognition* **25**, 71–102.

Marslen-Wilson, W.: 1989, 'Access and Integration: Projecting Sound onto Meaning', in W. Marslen-Wilson (ed.), *Lexical Representation and Process*, MIT Press, Cambridge, Massachusetts, pp. 3–24.

Marslen-Wilson, W.: 1989, (ed.), *Lexical Representation and Process*, MIT Press, Cambridge, Massachusetts.

Matthews, R.: 1988, 'Psychological Reality of Grammars', Unpublished paper presented at the conference on *The Chomskyan Turn*, Tel-Aviv and Jerusalem, April, 1988.

Naish, L.: 1986, *Negation and Control in Prolog*, Springer-Verlag, New York.

Pereira, F. and D.H.D. Warren: 1980, 'Definite Clause Grammars for Natural Language Analysis', *Artificial Intelligence* **13**, 231–278.

Quine, W.V.: 1961, 'The Ways of Paradox', in W.V. Quine, *The Ways of Paradox and other Essays*, Harvard University Press, Cambridge, Massachusetts, pp. 1–18.

Reiter, R.: 1971, 'Two Results on Ordering for Resolution With Merging and Linear Format', *Journal of the Association for Computing Machinery* **18**, 630–646.

Stabler, E.P. Jr.: 1983, 'How Are Grammars Represented?', *Behavioral and Brain Sciences* **6**, 391–421.

Stabler, E.P. Jr.: 1984, 'Berwick and Weinberg on Linguistics and Cognitive Psychology', *Cognition* **17**, 155–179.

Stabler, E.P. Jr.: 1990, 'Representing Knowledge with Theories about Theories', *Journal of Logic Programming* **9**, 105–138.

Stabler, E.P. Jr.: 1991 forthcoming, *The Logical Approach to Syntax: Foundations, Specifications, and Implementations of Theories of Government and Binding*, MIT Press, Cambridge, Massachusetts.

Steedman, M.: 1989, 'Grammar, Interpretation and Processing From the Lexicon', in W. Marslen-Wilson (ed.), *Lexical Representation and Process*, MIT Press, Cambridge, Massachusetts, pp. 463–504.

Tyler, L.: 1980, *Serial and Interactive Parallel Theories of Syntactic Processing*, Occasional Paper No. 8, Cambridge, Massachusetts, MIT Center for Cognitive Science.

Tyler, L. and W. Marslen-Wilson: 1977, 'The On-line Effects of Semantic Context on Syntactic Processing', *Journal of Verbal Learning and Verbal Behavior* **16**, 683–692.

VanderBrug, G. and J. Minker: 1975, 'State-space, Problem-reduction, and Theorem-proving—Some Relationships', *Communications of the Association for Computing Machinery* **18**, 107–115.

Winograd, T.: 1972, *Understanding Natural Language*, Academic Press, New York.

Department of Linguistics,
University of California at Los Angeles,
Los Angeles, California 90024, U.S.A.