# Chapter 2
# Grammars and Models of Language Use

A key point of contact between experimental psycholinguistics and the theory of generative grammar has been the natural link between theories of knowledge representation and theories of knowledge processing, the grammatical basis of linguistic performance. Although the link has always seemed natural enough, it has proved difficult to forge. On the one hand, the rules and representations of generative grammar serve as a description of knowledge representation, a system independently justified by the role it plays in guaranteeing language learnability. In contrast, very little is known about the machinery actually governing sentence processing.

As a starting point, it was hoped that the theory of grammar could fill in some of the details about how this machinery worked. It is easy to see the logic of this. If we could show that the independently justified grammar plus some processing theory could predict external behavior, then we would have some support for the proposed processing model, and additional evidence for the proposed grammar. Of course, since processing evidence is not veridical, its power to evaluate grammatical theories can only be as strong as the confidence we have in it. If the processing theory and grammar predict only a partial and unnatural class of linguistic behaviors, then support for the processing theory must be correspondingly weak. Similarly, failure to predict behavioral data does not automatically foretell a grammar's doom; this is so only if we can show that there is no processing theory–grammar pair that can produce the right external behavior. There is then one big hitch. A given theory of grammar might be compatible with a variety of parsing models. Unfortunately, the more flexible the grammatical system with respect to implementation, the weaker its ability to constrain what the parser looks like.

We might also ask if the constraint could work the other way. Logically, the answer is yes. We ought to be able to recruit sentence processing results to tell us something about what the grammar should look like. If we had some independently justified parsing model, we could reject grammars that were incompatible with it. In practice though, because very little is known about the details of the syntactic parser, confidence in constraining the choice of grammatical theory via this route must be correspondingly weak. If the parsing theory has no independent motivation, we can always change it to suit the grammatical format.

Even so, some have argued that experimental results are the most important, perhaps the only, road to psychological justification of a theory of grammar. On this view, informant judgments and other evidence standardly used to justify linguistic analyses do not bear on the question of "psychological reality"

> I am assuming...that use and understanding of a language can profitably be described at one level...in computational terms. If we wish to claim...that the language user actually performs these computations...then each step must take some time. Hence, a crucial test of adequacy for such a model is its ability to make correct real-time predictions....real-time tests provide the most versatile and potentially the most finely calibrated measure of complexity; hence, they also constitute the acid test for claims of psychological reality for grammars. (Wasow 1978:83–84)

It is a rare linguist indeed who would say that only informant judgments determine the psychological adequacy of grammars. The real question is whether this evidence has been very illuminating up to the present.[1] Chomsky, for one, certainly claims that experimental results may bear on our theories about a speaker's linguistic representations:

> In the real world of actual research on language, it would be fair to say, I think, that principles based on evidence derived from informant judgments have proved to be deeper and more revealing than those based on evidence derived from experiments on processing and the like, although the future may be different in this regard. If we accept—as I do—...[the] contention that the rules of grammar enter into the processing mechanisms, then evidence concerning production, recognition, recall, and language use in general can be expected (in principle) to have bearing on the investigation of rules of grammar, on what is sometimes called "grammatical competence" or "knowledge of language." (1980b:200–201)

Chomsky insists then that informant judgments are at least legitimate

as evidence for a psychological theory of language.[2] He imagines a situation where a linguist has just finished providing an argument based on informant judgments for a particular mental construct. The linguist is then asked to provide real evidence (that is, evidence bearing on psychological reality) for such a construct. Chomsky points out that in this situation:

We observe what people say and do, how they react and respond, often in situations contrived so that this behavior will provide some evidence (we hope) concerning the operative mechanisms. We then try, as best we can, to devise a theory of some depth and significance with regard to these mechanisms, testing our theory by its success in providing explanations for selected phenomena. Challenged to show that the constructions postulated in that theory have "psychological reality," we can do no more than repeat the evidence and the proposed explanations that involve these constructions. Or...we can search for more conclusive evidence, always aware that in empirical inquiry we can, at best, support a theory against substantive alternatives and empirical challenge; we cannot prove it to be true. (Ibid.:191)

This "more conclusive evidence" could, of course, be reaction time data or further informant judgments. On this view a theory that could handle both informant judgments and online processing data is preferred to one that merely handles judgment facts. This point has often been misconstrued. For example, Bresnan and Kaplan (1982:xxi), citing a fragment of the last passage, conclude that Chomsky has explicitly denied the possible relevance of experimental results for linguistic theory when in fact he has explicitly affirmed it: "On Chomsky's view, then, a grammar is psychologically real if it contributes to the explanation of linguistic judgments and the other verbal behavior studied by linguists, and nothing more need be said."

In a related argument, Bresnan and Kaplan claim that Chomsky's notion of psychological reality forces us to accept a phonological derivation of *baritone* and *grieve* from the same underlying form:

This however, is a much weaker conception of psychological reality than we would like.... For example, the English words *baritone* and *grieve* derive from an Indo-European root *gwer*–'heavy'.... Now it is possible to construct a formal system of morphophonemic rules and abstract representations for these historical relations. By such rules one can formally derive English words from their Indo-European roots. Thus the labiovelar *gw* is the source of both the initial *b* in *baritone* and the initial *g* in *grieve*.... Would such a formal rule system be psychologically real? With Chomsky's conception of psychological reality, we could answer affirmatively. (Ibid.:xxi)

Is this true though? Assume that native speakers come equipped with a universal grammar that allows them to shape the phonological stimulus. Does this stimulus include information about the historical derivation of the words of their native language? Only if native speakers are equipped with a racial memory for the derivational history of their language. This is perhaps bizarre enough, but things are worse. Any person can become a native speaker of any language, via transfer at birth. But then speakers would have to be innately endowed with the derivational histories of all the world's languages, which seems absurd.

The apparent paradox dissolves if we realize that sometimes common historical derivation is reflected in the synchronic rule system of a person's native language, as a residue of things past. Latinate words have predictably different stress patterns from words of Germanic origin (Chomsky and Halle 1968). Chomsky and Halle argue that these words must be marked as belonging to two different classes. It is a mere historical accident—one that is not to be captured by generative phonology—that these classes correspond to different historical sources (that we, as outside observers, label with convenient, perhaps misleading, names). The same logic applies to the example that Bresnan and Kaplan cite. Chomsky has always claimed that informant judgments are relevant only insofar as they contribute to our understanding of how speakers learn their native language. Insofar as the notion of racial memory plays no role in this enterprise, commonality of historical source plays no role in determining an underlying representation of a phonological form (though it may appear to do so, as an historical by-product).[3]

We have two questions here. Can we use the theory of grammar to constrain the choice of possible parsers and vice-versa? How can we use the theory of grammar to constrain parsers? Logically, of course, the more evidence bearing on a theory the better. Practically speaking though, we must remember it might turn out that two systems can be made compatible in a large variety of ways. The more parsing machines that can be made compatible with a grammar and with behavioral results, the less initial constraining power that the grammar has for the theory of parsing. In particular this holds for the case where any single theory of parsing has weak or inconclusive independent justification.

Nonetheless, the simplest answer to the second question is that competence and performance are connected as directly as possible. This answer was tried first. Miller and Chomsky (1963) identified

rules of the grammar with computational operations of the parser in a one-to-one fashion. This identification led to specific behavioral predictions, collapsing grammatical with processing complexity; the more transformations needed to derive a sentence by the grammar, the more computational steps needed to parse a sentence. Again, this simple first attempt was the natural one. If it had been correct, we would have learned a lot about the parsing device; namely, that it was a serial machine that actively computed the inverse of transformational rules on-the-fly. We would also have fresh confirmation, from an entirely different source, for transformational grammar.

> The psychological plausibility of a transformational model of the language user would be strengthened, of course, if it could be shown that our performance on tasks requiring an appreciation of the structure of transformed sentences is some function of the nature, number, and complexity of the grammatical transformations involved. (1963:481)

Miller and Chomsky's original (1963) suggestion is really that grammars be realized more or less directly as parsing algorithms. We might take this as a methodological principle. In this case we impose the condition that the logical organization of rules and structures incorporated in a grammar be mirrored rather exactly in the organization of the parsing mechanism. We will call this *type transparency*.

The intuitive appeal of the type transparency condition is easy to understand. The demand for a direct relationship between the theoretical objects of grammar and those of parsing would seem to allow experiments that tap into actual online processing to bear equally directly on the choice of both grammars and parsers for natural language. The claim that we now know enough about the grammar-parser relationship for experimental data to bear on the choice of grammars has been expressed, for example, by Bresnan: "But the grammatical realization problem can clarify and delimit the grammatical characterization problem. We can narrow the class of possible theoretical solutions by subjecting them to experimental psychological investigation as well as to linguistic investigation" (1978:59).

But, as with all experimental paradigms, reaction time experiments do not have any *a priori* methodological status. We do not start out knowing that this paradigm will reveal deep properties about language use. We must find out that this is so.[4] Type transparency helps guarantee that the reaction time paradigm has the requisite constraining power, but we obviously need independent arguments that

this principle holds. We might try to provide a methodological argument for type transparency. That is, we might claim that in principle the rules and representation of a grammar should be token-token identical to the corresponding components of a parsing algorithm. This argument seems unreasonable. As Chomsky has repeatedly noted, the grammar describes only what knowledge a speaker/hearer has of language; it does not prescribe any one particular parsing algorithm for how that knowledge is put to use. Even though one should expect a natural connection between what you know and how you put that knowledge to use, the connection may be rather indirect:

it is important to distinguish between the function and the properties of the perceptual model PM and the competence model G that it incorporates.... Although we may describe the grammar G as a system of processes and rules that apply in a certain order to relate sound and meaning, we are not entitled to take this as a description of the successive acts of a performance model. (1968:117)

Even if we cannot make the direct mapping a logical necessity, we might ask if there are other arguments favoring such an approach. Bresnan and Kaplan (1982) try to make such an argument from simplicity and uniformity:

If it is uncontroversial that stored knowledge structures underlie all forms of verbal behavior, the question arises of how these different components of linguistic knowledge are related. To reject the competence hypothesis [the Bresnan and Kaplan version of type transparency rcb/asw] is to adopt the theoretical alternative that a different body of knowledge of one's language is required for every type of verbal behavior....it is the weakest hypothesis that one could entertain since it postulates multiple stores of linguistic knowledge that have no necessary connection. In contrast, the competence hypothesis postulates an *isomorphic* [our emphasis rcb/asw] relationship between the different knowledge components and is thus the strongest and simplest hypothesis that one could adopt. (1982:xix)

This position seems perfectly reasonable. It assumes that grammars are mentally represented and justified by their ability to describe a native speaker's acquisition of language. Beyond this, grammars do double duty in that they contribute to a theory of why speakers find sentences acceptable or ungrammatical (assuming that speakers use the mentally represented grammar to make such judgments). If we postulated another system of mental representation to handle language processing we would be advocating an odd redundancy. One system of knowledge would be for learning and another would be for parsing.

Even so, the uniformity argument does not guarantee isomorphism between the linguist's representations and those that people use for online processing. After all, the linguist does not typically ponder the issues of computational implementation that must be faced squarely by online processors, be they people or machines. Still Bresnan and Kaplan are right in the weak sense that we certainly want reasons for claiming that one system of linguistic knowledge is not sufficient to govern all types of verbal behavior. Any difference between the knowledge representation used in one domain and that used in some other should be justified. In principle, we would like to show that the modifications to the linguist's representational format are motivated functionally from below or above—from below by the language using mechanism (sentence producer or comprehension device) and from above by the peculiarities of the subtask that the machinery and grammar must perform. These conditions may force us to entertain less direct mappings between the grammar and parser. Starting from the strongest position, we could require an isomorphism between rules and operations of the grammar and the corresponding rules and operations of the parser. In its most literal interpretation this means that if a grammar derives a sentence using four transformations, then the parsing mechanism must take four operations to analyze the sentence.[5]

Plainly, far weaker conditions are still compatible with the requirement of "direct realization." For instance, we might insist that the parser merely preserve distinctions made in the grammar (allow a homomorphic mapping); then the parser would be free to make additional distinctions. But the spirit of even this weakened condition still requires that more complex derivations in the grammar map over into more complex parsing operations in an order preserving way; a derivation that takes five steps in the grammar should take, say, seven or eight steps for the parser.

We could weaken the condition on homomorphism still further, as is done in Bresnan 1978. The mapping that Bresnan has in mind is weaker than token transparency between rules of the grammar–parser pair. Rather, the distinctions between types of grammar and parser rules is preserved. Once this type–type relationship is acknowledged we can admit many positions intermediate between, on the one hand, absolute type transparency, and on the other hand, a weak form of association between grammar and parser where the grammar specifies only the extension of the function that the parser computes. We will

outline some of these alternatives later in this chapter. Both the type transparency and the Chomskyan approaches take the grammar as at least specifying the function to be computed by a parsing algorithm. That is, the grammar spells out which sentences input to the parser are to be considered members of a given language and provides structural descriptions for these sentences.[6]

On either view, the speaker/hearer's knowledge of language guides the use of language. At the broader theoretical level, theories about the system of linguistic knowledge (the grammar) guide the construction of theories of parsing. Both positions constrain the theory of parsing to choose among algorithms that are capable of computing the function specified by the grammar.

Weakening the relationship between grammars and parsers still further, we might demand that partial descriptions of such levels of representation as deep structure or surface structure must be preserved either isomorphically or homomorphically by the parser while allowing computational operations mapping between these levels to vary freely. This is effectively the position that Fodor, Bever, and Garrett (1974) maintained.

Under any of these interpretations we could still say that the parser operates so as to "interpret or generate sentences of L in the manner of G," to use Chomsky's phrase (1963:399).

In the rest of this chapter we shall sort out all these proposals. We begin by examining two approaches that have made crucial use of the type transparency condition. First we consider the derivational theory of complexity (DTC), which assumes the type transparency condition. Next we take up the extended lexical theory as outlined in Bresnan 1978. Both approaches presume that transformational rules are only weakly realizable.[7] Assuming that type transparency acts as an *a priori* methodological principle, sanctioning only direct mappings between grammars and parsers, it follows that transformational grammar is undermined as a description of linguistic competence.

Bresnan's criticism of transformational grammar opens by examining the failure of the direct explanatory connection between grammar and parser proposed by Miller and Chomsky. The experiments that forced a retreat are reported in Fodor, Bever, and Garrett 1974. Bresnan observes that the failure of the DTC has convinced many psychologists that "no model of language use that incorporates a transformational grammar, or indeed any grammar, is reasonable" (1978:2).

In fact, Fodor, Bever, and Garrett (1974) argued for retaining a transformationally based knowledge representation for learning while adopting "heuristic" or nongrammatically based representations to handle parsing facts. Bresnan adopts Fodor, Bever, and Garrett's conclusions and argues that since lexical-functional grammar permits an isomorphism between grammar and parser rules it is to be preferred to a transformational grammar. We will scrutinize this position in some detail below and judge it unfounded for the following reasons:

It is not clear that the reported psycholinguistic results are in fact problematic for current transformational theories.

The Bresnan system can handle the relevant psycholinguistic results only if it is allowed a homomorphic mapping between grammar and parser. This homomorphism can in fact be duplicated by a transformationally based parser.

In short, type transparency only favors lexical-functional grammar when it is combined with a particular view of human computational capacities. We shall show that we can provide a model for the type transparent realization of a transformational grammar simply by embedding the grammar in an alternative parsing system. By an alternative parsing system we mean an alternative measure of computational complexity that can be embedded in a machine. We outline one such proposal that employs the parser of Marcus 1980. Type transparency does not guarantee that psycholinguistic results can choose between competing grammars. Both transformational grammar and extended lexical grammar can meet the demand of type transparency, while retaining compatibility with the relevant psycholinguistic evidence.

More broadly, we conclude that the evaluation of psycholinguistic experiments is perhaps more complicated than has previously been thought. The proper evaluation of competing parsing procedures only makes sense if we supply two things: (1) the procedures to be compared written in a uniform language (an algorithmic language) and (2) an underlying theory of computational complexity, that is, a machine specification (its architecture plus explicit costs for each primitive operation of the machine), and how the procedures specified in the algorithmic language "execute" on that machine. In using psycholinguistic experiments to choose between grammars it is not sufficient to present one parser (incorporating some grammar) that can perform a certain task. Rather, we must justify at least in a preliminary way both the grammar and the theory of human computational capacity

underlying the parser. The computational moral is even more pointed in the particular case at hand. In order to use psycholinguistic evidence to show that one grammar is more highly valued than another we must have an independently plausible theory of computational capacity that yields the correct predictions for the experimental data most naturally when coupled with that particular theory of grammar. We will see that none of this has been shown. Current parsing evidence is neutral with respect to the choice between candidate grammars for natural languages.[8]

Arguments against transformational grammar are also flawed in their assumption that adequate grammars must meet the type transparency condition. We claim that even if a transformational grammar could not meet this condition this should not be construed as a decisive argument against the grammar. As we shall show, it is unwise to grant *a priori* methodological preference to theories that comport with type transparency.

The last section of this chapter deals more directly with the notion of type transparency as a theoretical principle. In this section we outline how a theory that does not assume type transparency may still express the relationship between a theory of knowledge of language and a theory of language use. We present a formal way of stating this relaxation of transparency via a device that has some currency in the study of parsers for programming languages, the notion of a *covering grammar*.

## The Derivational Theory of Complexity

We turn first to theories assuming type transparency. The classic view of a direct relationship between theories of grammar and parsing is the derivational theory of complexity, or DTC. At the core of the DTC is a simple set of theses about human sentence processing. The first is about what representation is constructed during a parse, the second about how that representation is constructed, and the third about the complexity of the computation itself. These are the three necessary components of any complete computational model that links grammar to external behavior: a representation, an algorithm, and a complexity metric.

What representation is computed:
The so-called Standard Theory (outlined in *Aspects of the Theory of Syntax*) was taken as the optimal theory of grammar. By type transparency, this means that sentences must be analyzed by a direct processing analogue of the Standard Theory.

How that representation is computed:
The sentence parser recovers both the deep and surface structure representations of the input string of words. The surface structure is built up by consulting the phrase structure rules of the grammar and matching them against the input string. The deep structure is derived from the surface structure by applying "inverse transformations," if any; otherwise, the deep structure is just "read off" the corresponding surface structure.

The measure of computational complexity:
Every grammatical operation has a unit time cost. That is, in order to be counted as an "active" component of the computation, each grammatical operation must take a unit of time to compute. A parser containing a grammar that maps between deep and surface structure by applying inverse transformations would thus assign a unit cost to each one. Because each inverse operation counts in the final complexity tally, we see that the model is implicitly serial. Two operations take two units of time. Thus the total cost of constructing the deep and surface structures is simply the sum of the total number of rules involved in the derivation of the sentence. Note that surface behavioral complexity reflects processing time, rather than processing space as measured by memory load.

So much for the theory. What about the experimental results testing it? Under the assumptions of the DTC a passive sentence would cost one more time unit than an active sentence because there would be an extra operation, the passive transformation, involved in the mapping between deep and surface structure for passive sentences.[9]

This hypothesis was investigated experimentally, and early work (experiments by McMahon 1963, Gough 1965, and Savin and Perchonock 1965) apparently supported it. However, later investigation evidently disconfirmed the DTC. Slobin 1966 and Walker et al. 1968 dealt the final blows. All experiments found no correlation between sentence processing time and length of transformational derivation. Fodor, Bever, and Garrett (1974:369–370) concluded that "the grammar is probably not concretely realized in a perceptual model."

There are four obvious ways out of this dilemma:
*Question the theory of grammar.* Either transformational grammar as a whole is wrong or the individual transformations contributing to results disconfirming the DTC are wrongly formulated.
*Question the behavioral results.* The experiments disconfirming the DTC are irrelevant to the theory of online sentence comprehension.
*Question the notion of direct realization.* The direct embedding of a transformational grammar into the online sentence processor is wrong.
*Question the complexity measure.* The direct association between unit time costs and sequential (serial) inverse transformational operations is wrong.

Each of these four "escape hatches" has been tried; we will survey the first three before turning to a detailed discussion of the last one, the question of alternative complexity measures.

In exploring the first alternative we consider a set of experiments that suggest minor modifications of the grammar (the Standard Theory) assumed by the DTC but otherwise leave the DTC unscathed. The theory of grammar underlying the Standard Theory is maintained but independent evidence is adduced to show that details about the rule format are incorrect or that there are distinctions among rules that were ignored when the early experiments were performed.[10]

The basic problem with the grammatical theory underlying the DTC is that the only formal operations it allowed were transformations. Subsequent work has shown that this restriction yields an unnatural class of transformations. It lumps together rules that have very different grammatical properties.[11]

As a case in point, Fodor and Garrett (1967) performed an experiment contrasting sentences whose noun phrases were modified by a series of prenominal adjectives with sentences containing only bare noun phrases. They assumed a theory that derived prenominal adjectives from relative clauses by a reduction and preposing operation called *whiz* deletion.[12] Under this analysis a phrase like (1a) derives from (1b) by first deleting the *wh* and verb sequence and then preposing the adjective to prenominal position:

(1) a. The red book
    b. The book which is red

Assuming that *whiz* deletion is a transformation, constructions with prenominal adjectives should take more total time to parse than those containing basic noun phrases. Fodor and Garrett found though that

prenominal modification produced no complexity effect: "The sentences with adjectives exhibited no tendency to inhibit subjects' accuracy on the paraphrase task" (Fodor, Bever, and Garrett 1974:325). Their results indicate that prenominal adjectives are not transformationally derived.

Williams (1975) independently reached the same conclusion. He provides a host of purely syntactic arguments against the *whiz* deletion analysis, offering an alternative theory where prenominal adjectives are simply generated in place. He then shows how this analysis fits with the kind of transformational grammar proposed by Chomsky 1970.[13] Assuming Williams's analysis, we may predict the complexity results of Fodor and Garrett while staying entirely within the framework of the Standard Theory.

Another purported counterexample to the DTC is reported in Watt 1970. Watt presupposes a theory advanced by many linguists in the 1960s whereby "short passives" (such as *John was hit*) are derived from "long passives" (*John was hit by Fred*) via deletion of the agentive *by* phrase. Watt (following Fodor and Garrett 1967) claims that short passives take no longer to parse than their longer counterparts, a problem for the DTC on the assumption that the deletion operation involved in generating short passives adds time complexity to the analysis of these sentences. But current linguistic theory generates short passives directly; no deletion of a *by* phrase is required.[14]

The next set of experiments seemingly impugning the direct embedding of a transformational grammar in a processing model deal with so-called particle and adverbial movements. In unpublished experiments (reported in Fodor, Bever, and Garrett 1974 and Mehler 1963) the following sentences were compared:

(2) a. John phoned up the girl.
b. John phoned the girl up.

Bever and Mehler (Bever 1968) compared sentences like (3a) and (3b):

(3) a. Slowly the operator looked the number up.
b. The operator slowly looked the number up.

At one time both of the (b) examples were supposedly derived transformationally from the corresponding (a) examples. (2b) was derived from (2a) by particle movement (Emonds 1976), and (3b) from (3a) by adverb preposing (Emonds 1976 and Keyser 1968). Given the assumptions of the DTC we would predict that the (b) examples would

be more complex than the (a) examples. No such complexity effects appeared.

The problem is that these rules do not look like ordinary transformations. In (2) and (3) above there is no reason to suppose that the (a) sentences represent canonical thematic structures and (b) derived thematic structures. In some cases relating a sentential adverb back to a postverbal position would even give the wrong semantic interpretation. There is no semantic reason to say that any of the adverb or particle positions are canonical deep structure positions. Neither the particle nor the adverb receives its semantic interpretation by being related to another position. Rather, the semantics for these elements read directly off the surface structure. Further, these rules do not apply successive cyclically. They cannot move a constituent out of the clause from which it originates, again unlike NP movement rules. Contrast, for example, (4b) and (4d). (The "[e]" denotes an empty position):

(4) a. I said that the operator quickly dialed the number.
b. I said quickly that the operator [e] dialed the number.
c. [e] seems [e] to have eaten the yogurt.
d. The yogurt seems [e] to be eaten [e].

Although (4b) is grammatical, it cannot be derived from (4a) because in (4a) the adverb modifies the verb *to dial* while in (4b) it modifies *said* and has no relation at all to the lower verb. It seems reasonable to assume then that the adverbs in these sentences simply originate in the clauses where they are found in surface structure. Adverb movement is only clause internal. In contrast, (4d) is derived from (4c) by the iterated application of NP movement.

In the case of particle movement, the particle cannot even be moved from the verb phrase from which it originated. Compare:

(5) a. I called the operator up yesterday.
b. *I called the operator yesterday up.

(5b) is ungrammatical because *yesterday* is a sentence adverb. Its derivation requires movement of the particle *up* out of the verb phrase and into a slot directly under the S. This is impossible if we assume that the domain of particle movement is strictly "local" or internal to the verb phrase (Emonds 1976).

These properties contrast sharply with those of the so-called passive rule and with the properties of noun phrase and *wh* movement in

general. Consider the derivation of (6a) from (6b) via step (6c):

(6) a. John was believed to have been seen by Bill.
b. [e] believed [e] to have been seen John by Bill.
c. [e] was believed John to have been seen by Bill.

Starting with deep structure (6b), we apply the passive rule to get (6c); the passive rule applies once more to the next higher sentence to derive (6a).

In contrast to the particle and adverb movement cases, it is clear that we must relate *John* back to its deep structure position because the proper interpretation of this sentence requires that *John* be construed as the logical object of *be seen*.

Because rules like particle movement and adverb placement have properties that are so different from passive, it has been proposed that these rules be classified as stylistic (Chomsky and Lasnik 1977 and Dresher and Hornstein 1979). They are cordoned off from the class of "core" transformations.

Under such a proposal we could still cling fast to the DTC by claiming that transformations but not stylistic rules are actually computed during sentence processing and therefore add to its complexity. Because particle movement and adverb placement do not influence semantics, we could perhaps claim that "detransforming" operations need not apply in these cases; the particles or adverbs are simply left in place, as they appear in the input string. This would accord with the DTC hypothesis that the sentence processor need only recover structure relevant to semantic interpretation.[15]

To summarize: The important point is that even the Standard Theory provides the distinctions needed to make the DTC compatible with the cited psycholinguistic complexity results.

Turning now to the actual behavioral data, the most devastating psycholinguistic evidence against the DTC comes from Slobin's experiment (1966). Slobin presented subjects with pictures of action scenes that were described by either passive or active sentences. Subjects were asked to verify whether the supplied sentences truly described the corresponding pictures. Given the DTC, pictures described with passive sentences should be associated with longer verification times. Presumably, the task forced subjects to retrieve a representation of the deep structure level, a structure presumably one unit harder to construct in the passive than in the active case.

Unfortunately for the DTC though, this expected difference showed

up only in the verification of pictures described by reversible passive sentences, that is, sentences such as *John was loved by Mary*, where either argument *John* or *Mary* may be reasonably interpreted as the subject or direct object of the sentence. These sentences contrast with nonreversible passives such as *The cookies were smelled by John*. Here only the animate *John* can be interpreted as the subject of the sentence because *cookies* are incapable of smelling. Because both reversible and nonreversible passives have the same syntactic structure, any account of the complexity difference between these two sentence types cannot be syntactic. Some other component of the grammar or processing system must account for the difference; Slobin suggested a hypothesized semantic component. Unfortunately, these results disconfirm the DTC particularly when one compares reversible passive sentences against active sentences. According to the DTC both reversible and nonreversible passives should take longer to compute than their active counterparts because both involve the same number of transformations from deep to surface structure.

Following the familiar pattern of debate in the psycholinguistic arena, researchers have recently questioned the validity of Slobin's results. Forster (1976) has noted that "using other experimental techniques (the RSVP presentation, Forster and Olbrei 1973), reaction times for passives were found to be significantly *longer* than those for actives when subjects were asked to decide whether sentences were 'intelligible and grammatical'" (1976 quoted in Bresnan 1978:49). If Forster is right then the correct parsing theory for English should predict that passives are more complex than corresponding active sentences. These results confirm the DTC. Forster and Olbrei's experiments showed no reversibility effects once sentences were controlled for their semantic plausibility.

Results aside, Forster has also argued that the RSVP technique is a better probe into online sentence behavior: "The problem with this [Slobin's rcb/asw] experiment is that it used a picture verification technique.... Unfortunately, this technique appears to have little to do with on-line sentence processing" (1979:47). We agree with Forster's criticisms. Because we are interested in the conceptual issue of Slobin-type results as a barrier to transformationally based models, for purposes of discussion we are going to ignore them. The issue raised by the Slobin results will surely emerge anyway if the computational models of psycholinguistic theories are not identical to those people

actually require. Even if it is false, the Slobin data can help us to prepare for this eventual day of reckoning.[16]

Fodor, Bever, and Garrett (1974) decided that the way to accommodate results like Slobin's was to revise the way that grammar and parser are related. In their model online sentence comprehension does not normally make use of a transformational component. Parsing functions previously attributed to transformations are in fact performed by "heuristic strategies" (1974:356ff.). The purpose of the heuristic strategies is to reduce or eliminate the amount of online computation involved in sentence comprehension.[17]

Instead of repairing transformational grammar piecemeal, we could do away with it entirely. This is the position taken in Bresnan 1978. The remainder of this section shows just why this position is unnecessary. First we discuss the assumption that extra computation must necessarily be associated with added time complexity in experimental tasks. This assumption also underpins Bresnan's 1978 critique of transformational grammar and guides the design of a computational model associated with her alternative theory of grammar. Next we show why this assumption need not hold. We present a model allowing simultaneous computation (relaxing DTC thesis 3 above) while directly realizing a transformational grammar in the parsing mechanism.

### The Extended Lexical Theory

We have seen that Slobin's refutation of DTC led to his rejection of transformational grammar as a "realized" component of sentence processing. Bresnan 1978 has also apparently taken these results to mean that the DTC has been effectively refuted. But rather than exploring alternative computational organizations, Bresnan has opted for modifying the grammar so that it is compatible with the parsing organization.

The Bresnan 1978 approach differs from transformational grammar in essentially two ways.[18] First, Bresnan claims that no noun phrase movement transformations[19] are part of the grammar or a model of sentence processing. Rather, these transformational rules are reformulated as rules of lexical-functional interpretation. Bresnan 1978 presents one way for these rules to be embedded in a parser, as precomputed templates rather than "active" computations. This allows Bresnan to embed the modified grammar (extended lexical grammar, or ELG) in a parsing model organized along the lines of DTC

assumption 3 above. She claims that the compatibility of this particular grammar-parser pair with results like Slobin's provides a strong reason for preferring the ELG theory of grammar to a transformational one.

In order to understand exactly how these claims about grammar interact with those about parsing, it will be necessary first to outline the kind of grammar that Bresnan envisions and then to sketch one way of realizing it in a parsing model.

The main difference between transformational grammar and extended lexical grammar is the method by which these theories relate the thematic argument structure of predicates to surface syntactic structure. Consider the following three sentences:

(7) John sang *the Messiah.*
(8) What did John sing?
(9) *The Messiah* was sung by John.

Here, *the Messiah* serves as the patient of the verb *sing.* Transformational grammar encodes this via a representational level of deep structure. In all three sentences *the Messiah* will be in the direct object position at this level of representation.[20] Transformations take care of mapping deep structure to surface syntactic structure.

The lexical-functional approach eliminates the deep structure level and the transformations for all the noun phrase movement cases, deriving thematic argument structure directly from the surface structure representation of a sentence.[21] This is done by "defining a set of lexical-functional structures that provide a direct mapping from the logical structure of a verb into its various syntactic contexts" (Bresnan 1978:23).

Let us provide a clarifying example. In English, positions in the phrase structure tree are associated with certain functional roles, telling us whether a noun phrase should be the subject, direct object, and so forth. Noun phrases in English receive functional interpretations as indicated: an NP directly dominated by S is a subject; an NP directly dominated by VP is an object; and an NP directly dominated by a PP is a prepositional object. Following Bresnan 1978, we abbreviate these NPs as $NP_1$, $NP_2$, and $NP_P$.

Returning now to sentence (7), the lexical theory enters *sing* into the lexicon like this:

(10) *sing*: $NP_1$ sing $NP_2$ (*to* $NP_3$)

This "template" tells us that whatever NP fills the $NP_1$ slot in the

phrase structure tree will act as the subject, whatever NP is in the $NP_2$ slot will be the direct object, and so on. The functional structure template is matched with the phrase structure tree that corresponds to sentence (7). *John* is interpreted as the subject of this sentence because it is the NP dominated by S; likewise, *the Messiah* gets dubbed the functional direct object because it is the NP dominated by VP.

On the other hand, the mapping deriving the correct thematic interpretation of the passive counterpart of (7) differs from the one needed for active sentences. The grammar must encode that the position associated with the surface grammatical subject is athematic, and that the element in this position picks up the thematic role associated with the direct object position. In our running example, a rule like the following will do the trick:

Eliminate $NP_1$ ...
Replace $NP_2$ by $NP_1$ ...
(Bresnan 1978:21)

To encode the athematicity of the subject position Bresnan suggests that we bind a variable to this position.[22] The argument in surface structure is then no longer associated with the thematic role normally given by the NP under S position. It has been de-thematized and has no thematic role to be given. Following the rule above, this argument is associated with the thematic role of the $NP_2$ position. *The Messiah be+past sung by John* is interpreted as,

($\exists x$ [ $x$ be sung *the Messiah* by John])

That is, in sentence (9) the noun phrase in the surface subject position is associated with the thematic role interpreted as the functional direct object of *sing*.[23]

How is this modified grammar embedded in a parsing model? To ensure the most direct mapping one could proceed in the following way: for the passive case we would need evidence from the input string that the "typical" interpretation is to be blocked. This trigger can be supplied by the lexical entry associated with the form *sung*. Then we would force the right noncanonical interpretation via the application of the above lexical relation. Given the phrase structure tree, we interpret sentence (9) by locating the NP directly dominated by S and removing any functional arguments associated with this position. The element in the $NP_1$ position is placed and thematically interpreted in the direct object position (the $NP_2$ position).

But the theory-grammar pair as presented so far still does not square the extended lexical grammar with psycholinguistic results like Slobin's, assuming now that the lexical redundancy rule mentioned above is an active computation performed by the parser. The interpretation of passive sentences still costs one unit of time more than their active counterparts. The difference is exactly the processing cost of the lexical redundancy rule. We are still left with a model equivalent to the DTC transformational version with respect to Slobin's timing results.

In order to make the model compatible with Slobin's results one could assume that the interpretation of passive sentences works by comparing the surface string to a functional structure template that is listed in the lexicon as part of the entry of the corresponding verb, just as in active sentences. This is the method that Bresnan 1978 suggests. The effect of the lexical redundancy rule rather than the rule itself is encoded into the form of the functional template associated with a passive verb. An example may make this clear. The functional template for the passive verb form of *sing* is:[24]

*Be+sing*:
($\exists y$ [ $y$ *sing* $NP_1$ (*to* $NP_3$)])

As this sentence is parsed, the same matching operation would be carried out as in the active case, but now the passive lexical form would be retrieved and the noun phrase in the structural subject position would be placed in the functional object position (as dictated by the template). The system interprets this noun phrase as the direct object. Since the same matching operation is involved in both the active and passive sentence, namely, the retrieval of lexical templates, the processing of active and passive sentences will now take the same amount of time.[25]

In this model the complexity distinction between reversible and nonreversible passives is not due to the relative complexity of retrieving the lexical templates. Rather, interpretation of noun phrase functional roles is harder in all passives. Passives cause the parser to stumble because one cannot provide a direct assignment between NPs in the phrase structure tree and argument positions in functional structure; some kind of additional manipulation is demanded. We might expect then that any extra cues indicating which NP matches with which argument position could potentially speed up comprehension. Nonreversible passives contain such cues by virtue of the verb's selectional restrictions, such as whether a verb takes an animate or inanimate subject. Such verbs

permit only a single well-formed mapping between NP positions in the phrase structure tree and functional structural positions. For example, in a sentence like *the flowers were sniffed by John* once the verb *sniff* is recognized, its selectional restrictions become available to the parser. The parser has an additional cue to tell it that the $NP_1$ in the phrase structure tree cannot be the $NP_1$ of the functional structure, because only animate NPs may appear first in the functional structure associated with *smell*.[26]

Finally it should be noted that in the extended lexical grammar both actives and passives contrast with *wh* movement constructions, in that they are derived by lexical rules rather than by the transformations used to derive sentences with *wh* movements. Therefore, the recognition of actives and passives need not involve the same computations required to analyze *wh* constructions. Thus Bresnan assumes a weaker version of type transparency. There is no one-to-one grammar–parser rule correspondence. There is a type of grammatical rule–type of parsing computation correspondence. Because Bresnan 1978 treats *wh* movement and passive separately in the grammar, she justifiably assigns these two rules to different processing components. In contrast, she argues by implication that transformational grammar retains both these rules as transformations, and so must parse them by the same type of algorithm. Conclusion: the lexical theory can capture similarities between actives and passives and differences between passives and *wh* movement that transformational grammar cannot.[27] Bresnan claims that the theory of language use

> should map distinct grammatical rules and units into distinct processing operations and informational units in such a way that different rule types are associated with different processing functions. If distinct grammatical rules were not distinguished in a psychological model under some realization mapping... the grammar could not be said to represent the knowledge of a language user in any psychologically interesting sense. (1978:3)

It is important to point out that the ability to distinguish two rules by their contribution to "time complexity" does not automatically follow from this argument. Rather this depends on their association to different "informational units" and a particular choice of computational organization. Given that the assumed parsing organization is so crucial to the argument against transformational grammar, it is important to investigate whether the same conclusions hold under alternative assumptions about computational organization. In the next section we

shall show that by allowing a rudimentary kind of parallel computation we can bring a transformationally based parser into line with existing psycholinguistic reaction time results. In particular we claim that if the Slobin data holds up it merely suggests a processing model that allows passive morphology to act as a local cue, telling the parser that a certain computation must take place. The computation is either movement or binding, depending on whether the parser uses the Standard Theory or Extended Standard Theory. In addition, we must assume that this computation can be carried out concurrently with the recognition and attachment of the verbal element, and thus need not require any additional externally measured reaction time. If, on the other hand, Forster 1976 and Gough 1965 are correct, then we can maintain the "local cue" hypothesis within a more serially based model.

## Alternative Parsing Models

In this section we shall see that by holding the transformational grammar constant and varying the other two "parameters" of the DTC, we can readily accommodate Slobin's psycholinguistic evidence. First we consider modifications to the computational organization of parsing. Our basic approach is to introduce a slight amount of nonseriality (concurrent processing) into the execution schedule of parsing rules. We show how the crucial nonconcurrent processing can be triggered in the "passive cases" upon recognition of the predicate of a passive sentence and how this triggering can be reasonably integrated into the machine architecture we have in mind.

We shall illustrate the impact of this nonserial processing by exhibiting parsing models for two transformational theories, the Standard and the newer Extended Standard Theories (Chomsky 1977a). Thus modified, both models will prove to be compatible with the DTC timing results.

### A Parsing Model for the Standard Theory

For the purposes of constructing a parsing model we need to briefly review the key premises of the Standard Theory (ST). ST assumes that there is one level of linguistic representation relevant to phonetic interpretation and one to semantic interpretation. Phonetic interpretation is "read off" the surface structure of a sentence, while semantic interpretation is determined by the deep structure configuration. Even though the

theory specifies that two representations must be recovered from the input string, it does not specify in what order they must be recovered. Deep structure may be "computed" after the entire surface structure tree is built, or, more to the point, it may be built in parallel with the ongoing construction of the surface structure tree. It is this last alternative that we shall adopt: surface structure and deep structure are built in tandem.

We shall use the parsing model designed by Marcus 1980. We adopt this parser to be concrete. The concurrency scheme to be sketched is compatible with any number of parsing models. The added benefit of this parser is that one can show that it is compatible with a transformational grammar. One last caveat: we are not interested in justifying all the details of the Marcus parser. Rather, we are interested in showing that by assuming different machine architectures we can radically alter what a particular grammar-parser pair predicts vis-a-vis reaction time and other measures of complexity.

The Marcus parser mimics the important properties of a transformational grammar in the following way. First of all, it is equivalent to a transformational grammar in the sense that for every surface string (sentence) it associates the same annotated surface structure (annotated by traces) that would be paired with that sentence by a transformational grammar. One might dub this a condition of "input-output" equivalence.[28] Note that this definition of equivalence leaves open the question of how it is exactly that the annotated surface structure gets computed.[29] That is, a parser could reconstruct the right underlying representations using principles entirely unrelated to the grammar and yet still be input-output equivalent to that grammar. The Marcus parser uses a transformational grammar much more directly than this; some of its "internals" are similar. It maps between levels of representation in accordance with each of the principles discussed in the first chapter. It is constructed to respect both parts of the opacity conditions and the subjacency condition; and it crucially makes reference to the lexical properties of items at every level of representation, in accordance with the projection principle.

To understand the following discussion, all one has to know is that a Marcus-style parser operates by making decisions based upon two sources of information. First, it uses the features of the parse tree node currently under construction plus the features of a noun phrase or sentence phrase (cyclic node) immediately higher in the parse tree.

This information is clearly useful in determining what the parse tree already built looks like, and hence what should be built next. Second, it uses the features of items in the input stream not yet attached to the parse tree, up to a limit, almost always, of three items (though this last constraint can be relaxed in some circumstances to a "lookahead" of five items). Together predicates defined over (1) and (2) determine the parser's next move. The evidence the parser uses is strictly local, amounting to the examination of the features of nodes and input tokens in the "immediate vicinity" of the parser's activity.[30] At any given step in a parse the Marcus parser can access the contents of five "cells" in order to decide what to do next, two for the nodes corresponding to partially or completely analyzed phrases that will become part of the parse tree and three for the lookahead. We will assume that access to each of these five cells takes only constant time. The contents of these cells may be retrieved, examined, or modified, all in constant time.

Parsing an active sentence in this model is straightforward. Words enter the input stream, three at a time under most circumstances.[31] The surface structure and deep structure trees are built in parallel, elements in the input stream being placed simultaneously in their proper positions in each one. First the parser assembles the subject noun phrase and attaches it to the S node. Next, it builds the verb phrase and attaches it to the budding parse tree. A sentence such as,

(1) The girl kissed the boy.

parses in roughly eight steps of the parser: two steps to assemble the NP, and one to attach the NP to the S; five to build the VP and one to attach the VP to the S (one for the verb, one to attach the verb to the VP, and three to build the NP and attach it to the VP). In a simple active sentence the deep structure tree is isomorphic to the surface structure tree. Let us assume that recognition of the predicate of a sentence also entails the retrieval of its subcategorization frames (see Marcus 1980 for one way that this proposal may be carried out). This is the parsing analogue of the projection principle. Experiments that tap online processing support this assumption. There are significant complexity differences between the comprehension of "anomalous" and fully well-formed sentences (Forster 1979). These two classes of sentences are often distinguished solely because the "anomalous" cases fail to meet a predicate's thematic or selectional restrictions. One way to explain these results is to assume that subcategorization frames are available for online processing.

We now turn to a mechanism for analyzing passive sentences. The approach in Bresnan 1978 uses lexical lookup to analyze both passive and active structures. The extra complexity of reversible passives follows from the extra complexity in assigning the proper functional roles to NPs in the phrase structure representations of such sentences. Nonreversible passives have complexity compensating extra cues guiding the NP–functional role mapping.

We may adapt this approach to a computational model based on transformational grammar. By using a modest amount of parallel processing whereby two parsing actions can take place simultaneously, we shall show that the analysis of passive structures takes the same amount of time as the recognition of corresponding active structures.

This model can then be modified so that finding the proper NP is more difficult for passives than actives (the resolution is by actual movement in the Standard Theory or binding in EST). In previous work (Berwick and Weinberg 1983) we explained the reversibility effects in terms of properties of the online sentence processor. Bresnan and Kaplan (1982) also explore this possibility. On further reflection this alternative should be rejected. This is because Slobin found reversibility effects in active as well as passive sentences.[32] This rules out a purely syntactic processing explanation that makes all actives faster than passives. It follows that the form of the explanation for the reversibility facts should be more explicitly along the lines Slobin suggested. That is, some sort of semantic processing adds complexity to the picture matching task. The syntactic processing of active and passive sentences matched for reversibility should be equally difficult. Let us see how this might work in detail.

In almost all cases, the ability of the Marcus parser to look at three words or constituents will allow rules to simultaneously access the subject, auxiliary verb, and verbal material of the predicate. To take a concrete example consider the following sentences:

(2) The eggplant was kissed.
(3) The boy was kissed.
(4) The girl kissed Fred.

During the parse of the first sentence, the parser's input buffer will first be filled as follows:

| the | eggplant | was | kiss+ed [__NP] |
|---|---|---|---|

The NP frame below the verb says just that *kiss* requires an NP object. The *ed* is the residue of prior morphological analysis that has stripped off the affix of the verb and added the appropriate feature information that the verb is marked *ed*. We have also taken the liberty of filling four buffer cells for the sake of illustration. In reality *kiss* would not enter the buffer until slightly later, after the NP has been completely built and entered into the first buffer cell as a unit.

Here is a quick rundown of the sequence of parsing steps that would be initiated. The parser determines the categorization of the first item in the buffer cell. We assume with Marcus that recognition of the first category as a noun phrase is tied to the recognition of the determiner. The termination of the noun phrase with *the eggplant* can be deduced by looking at the item in the second buffer cell, the verbal element *was*. Because tensed verbs cannot be part of an NP, the parser knows that the NP is complete, closes it off and prepares to attach it to its mother node. The input buffer will now look like this:

| [the eggplant]$_{NP}$ | was | kiss+ed |
|---|---|---|

The parser must now fix the category of the subject NP. In the Marcus parser this is confirmed by the contiguity of a noun phrase and verb. The verb appears within the lookahead window of the device. The participle-verb sequence and particularly the morphological features of the verb signal to the parser that this predicate is passive. We assume that the passive morphology tells the parser that the preverbal noun phrase has to be inserted in postverbal position. While the parser attaches the NP to the sentence initial position in the surface structure tree, it also holds this element in the buffer of the deep structure tree that it is concurrently building. Next, just as in the active case, the parser assembles the VP by joining auxiliary and verb nodes together. Again, the ability to use morphology to recognize the predicate as passive is crucial. Upon recognition of this morphology, the parser simultaneously (1) labels the predicate *+passive* and (2) attaches the V to the VP node under construction. The passive morphology as encoded by the feature *+passive* signals the parser that the predicate

must have a postverbal NP at the deep structure level. Consequently, the parser moves the NP previously unattached at the deep structure level into postverbal position in the deep structure tree. The reader may verify that this parse of a passive takes exactly the same number of steps as the parse of a corresponding active sentence, precisely the Slobin result.[33]

The two key properties of the passive rule that permit us to achieve this speedup by concurrent processing are first, the passive expresses a local dependency and second, this dependency can be detected by examination of surface verb morphology. The detection of passive morphology within the buffer window allows both nonattachment of the NP to the tree and its appropriate postverbal insertion. It is crucial that the preverbal NP not be attached to the tree. Otherwise, moving it to its proper position would take an extra step. Put another way, the passive rule can be expressed in terms of the local, five cell "vocabulary" of the parser's lookahead and items in the current cyclic clause. Because access to any of the five cells within the parser's field of view takes constant time, given a strictly sequential execution the total time for the verb attachment and object relocation associated with a passive would be at most some multiple of the primitive execution time of parsing operations, say, three operations vs. one for the active form (one to attach the verb; one to locate the NP under the S; one to attach the NP under S as the object). This multiple cost is "recoded" into a unit cost by assuming the simultaneous attachment of the verb and the (pseudo) movement of the true object NP.[34]

More generally, such a result demonstrates that external, real world time measured by the experimenter need not bear any simple relationship to algorithmic time (the number of steps used by some procedure under some simple model of computation like a Turing machine). In fact, as we discuss directly below, the time observed via reaction time probes may be more closely related to parallel time, which corresponds more closely to the space that a serial Turing machine uses.

A bounded rule like passive is very different from a rule that operates over seemingly unbounded domains, such as *wh* movement. If the "speedup" analysis is accurate, then our inability to apply local recoding to a rule like *wh* movement implies that the processing of *wh* movement sentences should take ever longer amounts of time as the structural distance between the displaced *wh* clause and its underlying

position grows. In fact, this complexity distinction between local and unbounded dependencies is just that suggested by the extended lexical theory of Bresnan 1978. This approach aims to eliminate as transformations precisely those rules that are expressible as local dependencies (passive, *there* insertion, so-called raising, and the like), retaining only "unbounded" rules like *wh* movement. Intriguingly, just those rules alleged to be not realistically captured by transformational grammar are amenable to local analysis, hence to concurrent speedup in certain parsing models. The nonreality of these rules can be attributed to particular assumptions about the organization of processing, rather than any failure in principle of transformational grammar.

### A Parsing Model for the Extended Standard Theory

Like the Standard Theory, the Extended Standard Theory (EST) is a model incorporating both a deep and surface structure and a transformational mapping between these levels. In contrast to the Standard Theory, EST holds that both phonetic and semantic interpretations can be read off a single representation, namely, annotated surface structure or *s-structure*. In the Standard Theory, the only way to recover the thematic relations of a given noun phrase (such as *John* in *John was kissed*) is to recover the deep structure. This is because the information that *John* functions as the object of the predicate *be kissed* is only accessible at the level of deep structure. In an EST framework when a category is moved it leaves behind a structural residue, a trace, in the position from which it originated. We can use the trace to tell us what has happened.

A parser must do just the reverse of this. It must "undo" the effects of movement, determining where traces are in the input stream. This is a nontrivial task because traces have no phonetic content. It must also link displaced constituents to the traces in an appropriate way. Such a parser starts with the representation of a sentence in phonetic form (PF) and derives an annotated surface structure representation. Annotated surface structure provides an initial format for semantic interpretation, whatever that may be.

Marcus's parser builds a close variant of the EST annotated surface structure and so provides a ready-made format in which to illustrate the potential for concurrent processing. Specifically, wherever a trace is required in the parse tree, the Marcus parser creates and attaches a noun phrase node labeled *trace*, co-indexing the trace to the constituent

to which it corresponds. For instance, the output representation for the passive sentence, *John was kissed by Mary* would look like,

$[[\text{John}]_{\text{NP}}\ [[\text{was kissed}][\text{trace}]_{\text{NP}}]]$

If a base generated active form has no movement rules applied and hence no traces, its analysis via the Marcus EST parser proceeds just as in the parsing version of the ST model; the annotated surface structure tree will look just like the surface structure tree built by the ST parser. Likewise, the EST parse of the passive sentence *John was kissed by Mary* parallels the construction of the ST surface structure tree up to the point after the passive verb morphology *was sung* is detected in the input stream and labeled as passive. Then, instead of next attaching the verb form and locating and moving the noun phrase *John* as in the ST analysis, we shall assume that the parser attaches the verb and simultaneously places a dummy element, a co-indexed trace, into the input stream:

| | | |
|---|---|---|
| NP-trace$_i$ | by | Mary |

The NP now in the input stream (the trace) functions syntactically as if it were a true lexical NP, that is, as if it were any ordinary noun phrase like *John* or *Mary*. As a result it attaches in the next step as the syntactic object of the verb. The remainder of the parse proceeds as in the ST model. The *by* phrase attaches to the annotated surface structure tree, as required. The essential point again is that the passive rule expresses a local dependency, visible even in the surface string because it is associated with specific verbal morphology within with the limited lookahead of the Marcus device.

Bresnan and Kaplan (1982) claim that this local dependency property is not enough to guarantee speedup by recoding. They suggest that the feeding relationships between transformations in a grammarian's derivation pose insurmountable problems to parallel computation:

> to successfully mimic the results of the LFG-based model, however, they [Berwick and Weinberg rcb/asw] would have to demonstrate that the operations specified by *all* of the sequences of standard transformational operations—Dative, Dative–Passive, Dative–Passive, *There* insertion, etc.—can be executed in unit time. But because these operations are in true feeding relationships, in which the necessary input of one operation is created by the output of another, it is simply not possible to execute them in parallel. (1982:xxxvi)

This criticism has two major flaws. First, it assumes that the "feeding relationships" established when mapping from deep to surface structure (as in the grammarian's derivation of sentences) remain intact when one maps from surface to deep structure (as one does in parsing). Is this really so? Consider the case of passive–*there* insertion. These rules are in a feeding relationship if the structural description for *there* insertion is not met until passive has applied. Assume that the description of *there* insertion is roughly this:

(5) Structural description and change:

| NP | (Aux) | be | X | → | *There* | Aux | be | NP | X |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 3 | 4 | 5 | | 1 | 3 | 4 | 2 | 5 |

Given a deep structure like (6a), we see that the structural description for *there* insertion is not met until passive applies, yielding (6b).[35]

(6) a. [e] was being eaten an apple.
b. An apple$_i$ was being eaten $e_i$.
c. There was an apple being eaten.

There is a feeding relationship for generating such sentences. But in the corresponding parse we know that *there* insertion has applied because we can recognize the lexical item *there*.[36] We also know that this element must be linked to the postposed subject. Consequently, *there* insertion can be recognized independently of passive. There is no feeding relationship for the purpose of parsing.[37] Besides, all these operations are local. *There* insertion is marked by the presence of a specific lexical element. The relationship between it and the postposed element is also a local dependency; the parser can link this element directly upon its recognition of *there* in the subject position. There is no problem in parallel parsing here at all.

The feeding critique has another problem. It is grounded on an outdated *Aspects* style transformational theory, but many of the rules of that theory have long since gone the way of the dinosaur. For example, a problem is supposed to occur when passive interacts with dative movement, as in a sentence like, *Politicians are being given too many gifts.* The problem is that the structural description for dative is not met until the passive transformation is undone.[38] We agree that this might cause problems if we were to interpret dative as an active computation, because in this case we would have to wait until we undid the passive before we could undo the dative, thus adding to time complexity.[39]

The principles of transformational grammar however, or at least recent versions of transformationally based theory, point to dative as a lexical rule. The first empirical argument in a transformational framework to this effect appears in Oehrle 1975. Unlike passive, a transformational dative rule would violate most otherwise well-motivated principles governing the class of transformational rules as a whole. Dresher and Hornstein 1979 provide another set of arguments that would rule out a transformational treatment for the dative in English. Dresher and Hornstein motivate the "trace erasure principle," stating that only designated NP elements like *it* or *there* can erase traces. A dative rule would violate this principle. More recently the $\theta$ criterion and the projection principle of government-binding theory supplant the trace erasure principle and prohibit a dative rule. In short, many recent transformational treatments prohibit a dative transformation. Because passive and dative must be distinguished by the principles of a transformationally based grammar, we are free to treat dative as a lexical template while treating the passive as a different kind of computation. In the sentence above we could simultaneously insert the trace of *the politicians* and look up the dative subpart of the lexical entry triggered by the recognition of the verb *give*. In other words, recognition of dative constructions could proceed exactly as in the lexical-functional framework.[40]

## Two Views of Cognitive Capacity

Let us take stock of what has been discussed so far. The DTC conjoins several hypotheses: (1) a certain type of grammar, for example, a transformational grammar; (2) a transparent relation between grammatical and processing operations, for example, grammatical rule types and operations mirrored by parsing operations; (3) a certain computational organization of parsing; and (4) a particular computational complexity measure. There are then at least four easy ways in which to modify the grammar–parser relation to accommodate the DTC results. One could alter the theory of grammar, changing either the theory itself (Bresnan's approach) or the way the grammar is embedded into the sentence processor (Fodor, Bever, and Garrett's choice). One could retain a transformational grammar and make the complexity measure give way, changing initial assumptions about available computational power. This was our approach above. Surprisingly straightforward modifications suffice to accommodate the

parser to the reaction time data. We simply assume that the parser is able to perform a small finite number of grammatical operations simultaneously, rather than just one at a time. Then a rule like passive "costs" roughly the same as any other bounded rule.[41]

The concurrency model does not necessarily imply that a multi-component operation like "passive" fails to take extra computational effort.[42] The associated operations are more complex, but the difference in complexity is not measured in terms of time, but rather in terms of the extra "hardware" that is engaged to effect the computation. By expanding the computational power of the processor, increasing the amount of work we can get done per unit of externally measured time, we can make a once time intensive computation less so. In the case of passive we need not expand computational power in an unlimited way. We have assumed only that a small finite number of operations per unit time can be carried out. This is a quite modest use of the computational power of parallelism. As we shall observe below, the general use of parallel machinery admits much broader variation in the apparent time complexity of computations than this.

Stepping back a moment from the details of the fray, we can see that this approach is quite different from that of Bresnan 1978. Instead of assuming deeper computational power, Bresnan 1978 removes supposedly costly operations such as passive from "active" processing and replaces them with the retrieval of lexical forms. These "forms" reproduce the effect of rules like passive. Instead of a single lexical entry for each passivizing verb plus a single passive rule to generate or recognize corresponding passive lexical forms, Bresnan 1978 substitutes two separate lexical forms for each verb.[43] The effect of the passive rule is "precomputed" by expanding the rule over all verbs in the lexicon before any sentence is processed. Implicit in this view is the assumption that it is easier to look up a precomputed result than to compute it using some rule. Memory storage is large and retrieval is fast and nearly costless:

> Finally, I assume that it is easier for us to look something up than it is to compute it. It does in fact appear that our lexical capacity—the long-term capability to remember lexical information—is very large. (1978:14)

The underlying assumption here—that memory far exceeds computational capacity—leads directly to a theory of parsing that attempts to maximize the use of memory resources relative to computation.

We now have two divergent views on human cognitive capacity in sentence processing. On the one hand, the parser-modification approach suggests that the computational power of the language faculty is possibly quite deep and that significant resources are available for parsing. On the other hand, the Bresnan 1978 proposal implies that what can be rapidly computed is quite limited and that previously stored "remembrances of words past" are required. These two views seem to be empirically indistinguishable, at least for the restricted domain of psycholinguistic results for which any comparisons are available. We might, however, uncover other empirical reasons for choosing one approach over the other. For instance, the amount of parallel computation required might exceed any reasonable upper bound on human computational capacity.

The modest parallelism invoked earlier is unlikely to be beyond the limits of what is humanly possible. First of all, the parallel computation required is indeed quite modest, because only a small, finite number of operations need be computed at the same time. Second, by looking at other cognitive domains in which rapid processing is at a premium and in which we have some hard knowledge about the associated neural "implementation" we find that the neural hardware involved actually implements parallel computational power far beyond that required for syntactic analysis.

Consider the results on early visual processing in primates, as discussed by Marr and Poggio 1977, 1979; Marr and Hildreth 1980; Grimson and Marr 1980; Ullman 1979; Richter and Ullman 1980 and many others. It is reasonable to suppose that the primate nervous system's computational solution to such problems as finding the edges of objects, detecting motion, and matching points from left and right retinal images so as to obtain a fused stereo image involves a rich, highly parallel network of nerve cells interconnected in a quite specific fashion to compute exactly those functions demanded of it according to the theoretical account.

The same may be true for motor control. Early studies emphasized the power of memory. Horn and Raibert 1978 is a representative approach. More recently it has been shown that changes in representational format may eliminate the need for memory driven motor control (Armstrong 1979 and Silver 1981).

Interestingly then, the history of scientific investigation in two domains, visual processing and motor control, has been roughly the

same. In each case the very first computational accounts of how a particular human competence should be "realized" rested on looking up "precomputed" answers, remembering past visual, auditory or motor control "templates." In each case "look up the answer" theories now compete with theories assuming a richer computational power. These are often schemes that are more closely tailored to the particular domain under investigation. We would not be surprised if the investigation of syntactic processing turns out to recapitulate this history. Of course, we do not suggest that syntactic processing models must rely on deep computation. Whether memory-intensive or computationally-intensive methods are used in linguistic processing is an open question. Even so, recent results in neurophysiology call into question a blind reliance on memory. Poggio stresses that memory retrieval is quite slow compared to the computation that even a single neuron could carry out. This is particularly true given his discovery with Torre (Poggio and Torre 1980) that one neuron can encode many thousands of states:

> Any consideration of a tradeoff between memory and computation time applied to the brain is unlikely to give the same result when applied to computers.... On the one hand, memory access may be a relatively slow process involving several synaptic delays, while on the other, a large number of nerve cells acting in parallel may be capable of performing a large amount of processing quite quickly. The processing power of a single neuron is still largely unknown, but is probably much greater than the traditional view maintains (see Poggio and Torre, 1980). (Poggio and Rosser 1982:5)

We should, of course, be extremely cautious about generalizing these findings across cognitive domains; the computational problems the visual system solves presumably do not include the syntactic analysis demanded by the language faculty. But we can conclude from an examination of the powerful hardware in the visual system that such computational power is available at least in principle to the language faculty.

Let us attempt then to make precise the concepts "depth of processing" and "parallelism" considered informally in the previous section. First, these notions make sense only with respect to some reference model of computation. Otherwise, we cannot properly compare the resource use of one procedure relative to another. But does the choice of reference machine really matter? Many psychologists suspect that "parallel computation" might radically alter the view of what is or is not easy to compute; it is an informal aphorism that parallelism

naturally allows one to compute faster.[44] This result is a familiar story to computer scientists. By expanding the amount of hardware or space allowed, the amount of time taken to compute a given function often decreases. But we do not have in mind this standard and straightforward demonstration of the interchangeability of time and space resources. The following two results are apparently less well known. First, the standard theorems demonstrating the power of parallel speedup hold only if one posits computational circuitry that is not necessarily physically realizable.[45] Second, notwithstanding the difficulty of translating mathematical results to the real world, a similar kind of radical exchange of time for space does still hold for models assuming physically constructible parallel devices, in fact, for all such reasonable parallel models that have so far been proposed.

In a network model of "and" and "or" gates, parallelism is captured by being able to perform any finite number of gate operations ("ands" and "ors") at any single step. This formalizes the sense of "doing more than one thing at a time" mentioned in the previous section. The total time (number of steps) taken by the computation is clearly equal to the length of the longest path in the circuit from input to output; this is the depth of the circuit. Depth is one natural complexity measure of the computational work done by a circuit, one that corresponds to a measure of parallel time. We can also measure the complexity of a circuit by the amount of "hardware" (the number of gates) required to construct it. By convention this is called the size of the circuit.

These two measures, circuit depth (parallel time) and circuit size (amount of hardware), relate to the methods used to speed up parsing time at the expense of increased parallelism. Take the demonstration that the rule of passive could be incorporated into a fast parser if the amount of work allowed at any single "step" were expanded. This is just the sort of expansion captured by a circuit model. The circuit also permits the amount of hardware required to compute the answer for any particular input to vary. We are allowed to use more gates to parse a sentence ten words long than five words long. A trade-off between parallel time and hardware thus arises quite naturally in the context of the circuit model. By changing the "wiring diagram" of our machine we can effect substantial speed-up of certain computations.

This poses a specific problem for those who have already fixed upon a serial computational organization as the "right" underlying model to judge an algorithm's complexity. Suppose that the assumption of

seriality is incorrect and that the function is realized using parallel circuitry. The time it will take to compute an output is mirrored by circuit depth. If this is so, a reaction time probe (a measure of external clock time) will measure circuit depth and hence parallel time. This makes sense. If an operation is underlyingly parallel, then its execution time as measured externally should be identified with internal parallel time.[46] We can now formulate the key question. What happens if we use the wrong algorithmic model for a computation? Is there any cause for alarm? Is there any distinction between serial and parallel time possibly causing problems if we confused one with the other?

We would claim that it is potentially misleading to use a serial reference machine where a parallel one would be correct. The reason is that parallel time (circuit depth) does not map over into serial time as clocked by a Turing machine in the natural way one might expect. A circuit of size $T \log T$ can simulate a Turing machine using serial time $T$. $T$ is some function of $n$, for example, $n^3$, where $n$ is the length of the input.[47]

In brief, the "time cost" of an algorithm measured using a serial computational clock need not directly reflect the amount of externally measured time necessary for a person to carry out the procedure. Rather, the cost might be more closely allied to the amount of hardware (circuit size) engaged. The relation between externally clocked time and algorithmic time is lost because external time maps in a more complicated way to the size of a circuit. A circuit can often compute the same result as the Turing machine in less external time by becoming "wider," thus keeping its size to within the required $T \log T$ bound but compressing the needed depth. The exact compression possible would depend upon the number of gates allowed at any one level of the circuit, the number of wires feeding into and out of gates and exactly what problem was being "solved" by the circuit. The externally observed time behavior of such circuits could be diverse, ranging from little apparent difference with the serial algorithm to an apparent exponential increase in speed. In short, ticks on the external, experimental clock would no longer necessarily correspond to ticks of the internal, algorithmic one. We could have unluckily picked the wrong model for timekeeping. This is the abstract counterpart of the situation discussed above. Experimentalists have generally equated externally measured, phenomenal time with serial algorithmic time, assuming that external time complexity for sentence processing must

be proportional to the number of grammatical rules involved in the derivation of a sentence. The lack of fit between grammatical model time steps and external time steps has then been taken to imply a weakness in the theory of grammar, and a demand for more suitable theories. But this conclusion does not necessarily follow; the simplest remedy could be to move to an appropriate parallel clock.

We must bear in mind that these speed-up results hold under models of synchronous parallel computation, that is, models where the machine has been designed from the start to concurrently compute some function. The parser described in the preceding section is one such machine. It is meant to compute one basic output representation with some of its operations taking place simultaneously. These machines should not be confused with models of asynchronous parallelism, where several machines vie independently for a common set of computational resources. The submachines might or might not be computing the same function. A paradigmatic type of asynchronous parallelism is a time sharing system; here, many different programs run by different users all demand central resource use. As users of time sharing systems are well aware, there may be no speedup at all in some asynchronously designed systems. Indeed, it is often the case that as more programs compete for limited resources, the time it takes for each individual program to execute rises dramatically.[48] This distinction is crucial because many of the studies demonstrating people's rather limited abilities to "process in parallel" have assumed, probably rightly, an asynchronous model of parallelism. This assumption is natural because the tasks that have been tested are intermodal in nature, and thus naturally fall under the "operating system" rubric of competition for common resources (see notes 34 and 46 above and Posner and Mitchell 1967).

In contrast synchronous parallelism most naturally operates only intramodally, within some single component like a syntactic parser. The resource competition paradigm is not necessarily appropriate; results like Posner's indicating the apparent lack of parallel speed-up in people need not apply. Indeed, distinguishing between synchronous parallel and serial computation appears to be a difficult experimental task, as described in note 34. For example, as Anderson observes, the usual Sternberg additive componential paradigm "analyzes information processing into a sequence of stages, specifies which factors affect the time for each stage, specifies the time parameters for each stage, but does not analyze in computational detail why each stage takes as long

as it does or why it is affected by factors in the way that it is" (Anderson 1979:404–405). This means that the "inside" of any stage is opaque to further computational decomposition; one is free to choose a serial or parallel mechanism to "realize" the computational machinery of any stage, subject only to constraints of externally observable time behavior. But as we have just seen this means that the time observed via the Sternberg analysis for individual stages might actually be parallel circuit time, not Turing machine serial time, and parallel circuit time maps over into Turing machine space. If true, models based on serial time must be judged in terms of the space efficiency, not their time efficiency.

Let us sketch more carefully just how, even under the Sternberg assumptions of linear stage decomposition, parallel computation might make the interpretation of competing processing models more difficult. Suppose that the response time for some isolated stage varies as the square of the input problem size, $n^2$. Thus phenomenally observed time is quadratic. Suppose further that there are two processing models proposed to account for the computation of this stage and that the complexity of both models has been evaluated with respect to a serial reference machine. Model A uses quadratic time and linear space, model B uses cubic time and quadratic space. With respect to an assumption of seriality, model A comports with the psycholinguistic evidence, model B is too slow. But if a parallel circuit reference base is adopted, then B can run in parallel quadratic time $n^2$ (its old space requirement); model A, in parallel linear time. Model B is now closer to the psychological evidence, model A is now perhaps too fast.[49]

The parallel circuit model urges caution when we claim that one procedure is "better" than another in a cognitive domain. Claims about cognitive capacity and the trade-off between time and space resources depend upon both a precise specification of the algorithms to be compared and an underlying model of computation to be used as a reference base for comparisons. A judgment of algorithmic superiority may be empty without confidence in the fidelity of the reference model. We cannot simply stipulate an underlying model of computation on which to base our predictions of the amount of externally measured time a procedure takes without substantial support for that model. Otherwise we may be favoring certain kinds of procedures capriciously at the expense of others. Even though it may seem intuitively plausible that it is easier to "look things up" than

to compute them, it is not a logically necessary solution; in fact, the evidence from the domain of early visual processing points in quite the opposite direction. This makes it clear that cognitive science is, in the end, an empirical, biological science. While we may speculate about what is or is not the "right" computational organization of the brain (based on whatever psychological, engineering or computational biases we may have), ultimately there is a fact of the matter rendering such speculation moot. If the brain uses parallel computational power to process sentences rather than memory lookup, then that is what it uses; no stipulation can change this fact. If our aim is to discover what computational organization the brain does have, then stipulation again seems unwise. Given our current lack of understanding it would seem best to keep all (so far indistinguishable) computational organizations available, lest we rule out by fiat the "right" theory of processing.

There is one further point to discuss before concluding our comparison of extended lexical grammar and the Extended Standard Theory. In the preceding discussion we have assumed along with Bresnan that the transformations postulated by EST should be thought of as "active," time-consuming computations. We then provided a machine architecture that would make an EST based sentence processor compatible with certain psycholinguistic results. However, the necessity of this assumption is unclear unless there is also insistence upon some strong version of transparency and we state the Move NP rule and its parsing correlate in exactly the same form. If transparency is relaxed, it is possible to embed an ST or EST based parser in a serial computational model. We could do this by "precomputing" the effects of the transformational component and storing the results in the lexicon. Note that this in no way disconfirms transformational grammar either as a grammar or as a central component of a parsing model; it says merely that the way in which a grammar may be embedded as part of a model of language use is less than straightforward.

In order to understand why this is so we must be sure to distinguish the claims that a grammar makes about the system of knowledge incorporated in the language faculty from the implications of those claims for a theory of parsing. Let us make this point clear with a concrete example. Assume that psycholinguistic results show that passives and actives take less time to parse than *wh* movement constructions. In an extended lexical grammar a context-free base component derives simple active sentences, lexical rules derive passives,

and structural movement rules or their interpretive counterparts derive *wh* questions. Lexical rules and movement rules are quite distinct. "Functional" criteria govern lexical rules while structural principles govern movement. This distinction in the grammar—plus transparency—warrants a corresponding distinction between these two sorts of rules in the parser.

In contrast a transformational grammar treats both passive (NP movement in current theories) and *wh* movement as parts of the same component of the grammar, as subcases of a more general Move $\alpha$ rule. Consequently, certain critics of transformational grammar assume that we cannot draw any formal distinctions between NP and *wh* movement. Given a strong version of type transparency we should not be able to distinguish between these two sorts of rules in the parser either. We cannot have one parsing procedure for NP movement and another quite distinct computational routine for *wh* movement. The conclusion is that we cannot embed a transformational grammar in a parsing model of the sort Bresnan envisages.

The way out of this difficulty is to note that transformational grammar has long recognized that there are important formal differences between NP and *wh* movement. For example, Emonds 1970 argues that NP movement rules are structure preserving,[50] while *wh* movement rules are not.[51] Even in the most recent theories where NP and *wh* movements are taken to be special cases of a more general rule the theory posits different relations between a moved element and its trace. For example, the trace of NP movement cannot be case marked, while the trace of a *wh* movement must be in a case marked position. There are then criteria to distinguish NP movement from *wh* movement in the grammar. We are licensed to precompute the effects of NP movement and store them in the lexicon, while continuing to "realize" *wh* movement as an active computation. This is so even in a transformationally based parser. Given the assumptions of transformational grammar though, one should note that the precomputed lexical templates associated with NP movement would necessarily be governed by purely structural principles, unlike the templates of extended lexical grammar.[52]

Let us summarize. First of all, there seems to be no reason to adopt the particular machine architecture envisioned in Bresnan 1978. The psycholinguistic evidence does not force this choice. Second, there is no inherent link between a lexically oriented grammar and a precomputed memory retrieval scheme for the processing of active

and passive sentences. Transformational grammar also provides the necessary principles to distinguish among rule types so as to permit some rules to be handled by memory retrieval and others by active computation, all within a machine architecture like that assumed in the DTC or in Bresnan 1978.

## Type Transparency and the Theory of Grammar

Our story so far can be put as follows: Miller and Chomsky's assumption of a direct grammar-parser relationship seems to have foundered on confrontation with the DTC experimental results. The token-token identity between parsing and grammar rules must give way. What replaces it is a weaker type–type correspondence via the lexical-functional or revised Marcus parser approach. But have we lost anything by giving up this strict version of type transparency? We can assess any potential loss from either a logical or an empirical perspective:

(1) Logical: Is type transparency the most preferred relationship between grammar and parser?

(2) Empirical: Given what we know, is strict type transparency likely to constrain the choice of parsers or grammars?

Consider the logical perspective first. We would prefer that grammars be type transparently related to their corresponding parsers. Why? Simply because we could then use facts about the grammar to constrain directly what the parser looks like. Unfortunately this correspondence is not logically necessary.

If we insisted on strict type transparency we should reject the Bresnan system. Bresnan argues quite reasonably however that she can provide independent motivation for the rules of her system. This plus the understanding that a uniform representation for grammar and parsing must be grammatically based forces a weakening of the transparency thesis. Unfortunately even Bresnan's minimal weakening precludes the use of reaction time data to choose between her system and transformational grammar.

In any case, this discussion is premature because we know so little about the actual machinery engaged in human sentence parsing. This shortcoming greatly weakens the constraining power of type transparency. Even in the examples cited we can always change the underlying computational architecture so that otherwise well justified

theories may be transparently embedded in it. As things stand then, there is a continuum of more to less direct parsing "realizations" of grammars as parsers. There is not just an "all or none" choice between a grammar embedded directly as a computational model (the DTC model) and a total decoupling between grammatical rules and computational rules, with the structural descriptions of the grammar computed by some totally unrelated "heuristic strategies," the Fodor, Bever, and Garrett conclusion.

We might ask though just why one needs to specify a level of purely grammatical characterization at all. Why not simply dispense with grammar and just look at parsing algorithms directly, incorporating notions of computation from the start? Why should theorists interested in language use be concerned at all with a level of grammatical description? Indeed, some have suggested abandoning grammar as the proper subject matter for psycholinguistic investigation: "The proper task for the psycholinguist is not, at the moment, to determine the relationships between linguistic theory and psychological process, but to try to acquire the kind of psychological processing data which will allow the construction of a genuinely psychological theory of sentence recognition" (Tyler 1980:58).

The problems with jettisoning grammar altogether should be obvious. First of all we can use competence theory to constrain parsing theory. The choice of one parsing algorithm over another depends on a great many factors, including what structural descriptions the parser is supposed to compute and what computational organization is presumed to carry out the computation. As we have tried to make clear, very little is known about what computational organization is actually used for human language processing. We cannot even specify its basic structure with any degree of certainty. Though we have much firmer evidence about the correct characterization of linguistic knowledge, even that characterization is presumably far from the millenial theory of grammar. Consequently, we must build parsing theory on a doubly incomplete knowledge about the language faculty and human computational machinery.

Even if the grammar does not specify the exact computation or representation employed by the parser, it delimits a class of possible operations. Take the Marcus parser again. The Marcus parser's analogue of the "passive transformation" amounts to dropping a trace after a verb with passive morphology. The government-binding theory derives

the obligatoriness of this rule by the conjunction of the projection principle, $\theta$ theory, and Case theory. The projection principle ensures the parser will couple transitive verbs with postverbal NPs. $\theta$ theory links the postverbal NP to the phonologically realized NP in the subject position. Case theory insists that a postverbal NP of a passive participle will not normally be phonetically realized except where an independent nonverbal case assigner occurs in the right structural environments. The Marcus parser does not mention any of these subtheories explicitly. Nonetheless we claim that it realizes a transformational theory in a way that the heuristic strategies of Fodor, Bever and Garrett do not. The Marcus parser precomputes the effects of these principles. The ultimate consequence of the three subtheories forces a trace linked to the subject position in certain environments. Because certain sequences of deductions always apply we can precompute their effects by automatically dropping a trace upon recognition of a verb with passive morphology. This is the key insight behind the Marcus system.[53]

We claim that this system is still a "realization" of a transformational system. Only by reference to a transformational theory do we provide a principled answer to the question of why the machine performs the trace dropping automatically in such an environment. The heuristic strategy approach does not measure up. Rather, it matches passive structures directly against the following template:

Logical Object Verb+*en* (Preposition Actor)

The template would work if argument structures are linked directly to surface categories. This is true in lexical-functional grammar. But it would also work if argument structures are linked to deep structure positions and then interpreted, as in transformational grammar. So matching does not crucially depend on the assumptions of either of the two theories although it can be made compatible with either. It is this sense in which the heuristic strategy approach does not depend on the government-binding theory, and is not explained by it.

To summarize, the Marcus parser represents a fairly direct embedding of a transformational grammar in a model of language use. Logically speaking, there are other possibilities. The parser could use a very different set of representations than the grammar, yet still "realize" that grammar. In the next section we outline a general mathematical means of characterizing this kind of grammar-parser relationship, the theory of grammatical covering.

## Grammatical Covering and Type Transparency

The notion of grammatical cover appeared informally for several years prior to formalization by Reynolds 1968 and Gray and Harrison 1969. The idea that covering grammars could have value in developing parsing models for natural languages is not new. In fact, Kuno (1966) suggests precisely this. Intuitively, one grammar is said to *cover* another if the first grammar can be used to easily recover all the parses that the second grammar assigns to input sentences. This being so, the first grammar can be used instead of the second grammar itself to parse sentences of the language generated by the second grammar. More importantly, the cover relation provides a rich stock of cases where two grammars generate trees that do not necessarily look very much alike. Yet one grammar can serve as the "true" competence grammar for a language because it generates the proper structural descriptions while the other can be used for efficient parsing because of certain special structural characteristics of the trees it generates. In short, this approach allows us to hold the structural descriptions of a grammar fixed and then consider variations in parsing methods. The theory of grammar will limit the class of possible parsers to just those that cover the original competence grammar. This is possibly a strong limitation, hence of potential interest to parsing theory.

Such cases provide real examples of the existence of nontransparent ways to incorporate grammars into models of language use. Having settled the question about the existence of such grammar-parser pairs, the next question concerns the potential advantage of explicitly separating out the levels of grammar and parser in this manner. In the remainder of this section we shall advance some reasons as to why separation is advantageous. In brief, explicit decomposition into grammar, parser, and implementation permits a modular attack on the explanation of a complex information processing system, the language faculty. In addition, there are explanations of inherent psychological interest, such as accounts of language acquisition, that make crucial reference to a separate level of grammatical representation.

Returning now to the notion of grammatical cover, it is easy to show that there are well known but degenerate examples of pairs of grammars covering each other. Consider a grammar strongly equivalent to another grammar. By the usual definition of strong equivalence this means that the first grammar generates exactly the same set

of structural descriptions as the second, and so covers the second grammar.[54]

If this were the only example of grammatical covering, then the cover relation would collapse to the usual notion of strong equivalence. But it is also true that one grammar can cover another under far less stringent conditions of similarity. Informally, one grammar $G_1$ covers another grammar $G_2$ if (1) both generate the same language $L(G_1) = L(G_2)$, that is, the grammars are weakly equivalent; and (2) we can find the parses or structural descriptions that $G_2$ assigns to sentences by parsing the sentences using $G_1$ and then applying a "simple" or easily computed mapping to the resulting output. We shall be more precise shortly about what is meant by "simple." Note that these two grammars need not be strongly equivalent, and yet the first can still parse sentences generated by the second; for the purposes of parsing, such grammars are equivalent.[55]

Given some "correct" grammar for a language that is, a grammar generating the right structural descriptions, why would we want to parse sentences using a different but covering grammar rather than the correct grammar itself? The reason is that the covering grammar may be more "suitable" for parsing along any one of a number of dimensions (efficiency of processing in terms of time or space use, perspicuity, compatibility with fixed hardware, and so forth). On this view the "true" grammar provides the right structural descriptions for rules of semantic interpretation while the covering grammar provides the right format for algorithmic instantiation. These two grammars might be the same, in which case we obtain strict transparency, a one-to-one grammar-parser relation. The cover mapping takes trees generated by the covering grammar into trees generated by the "correct" grammar. Thus, if one grammar covers another, then whatever the rules of semantic interpretation either grammar can be used to pair exactly the same input strings and meanings.

What is meant by a simple mapping in the definition of grammatical cover? The mapping from parse trees to parse trees must be tightly restricted, or else any grammar could cover any other weakly equivalent grammar. The usual definition of "simple" drawn from the formal literature is that of string homomorphism (Aho and Ullman 1972:275). That is, if the parse of a sentence with respect to a grammar $G_1$ is a string of numbers corresponding to the rules that were applied to generate the sentence under some arbitrary numbering of the rules

of the grammar and some canonical derivation sequence, then the translation mapping that carries this string of numbers to a new string corresponding to another parse must be a homomorphism under concatenation. Note that the homomorphic recovery can proceed online, incrementally and left to right as the parse proceeds, and that if the mapping is fixed in advance the computation can be done quite rapidly.[56] What this means is that the desired parse tree can be recovered with little computational loss.[57] The covering grammar could look very different from the original grammar, however; the notes explore this matter in more detail.

Covering grammars provide an abundant source of examples illustrating that the grammar used by the parser or sentence processor need not directly generate structural descriptions isomorphic to those specified by the grammar of the competence theory. Indeed, the sets of structural descriptions directly constructed by the parser and generated by the grammar can look quite different, and yet the parser can still faithfully mirror the competence grammar by incorporating a cover homomorphism that recovers the proper structural descriptions as required. As far as parsing is concerned, both the theory and practice of parser design have made considerable use of a nontransparent relation between grammar and parser, that of grammatical cover.

But why should the notion of covering grammar play a role at all? That is, given that the mapping between grammar and parser may be quite abstract, why should we connect them at all? Why not just build a possibly nonlinguistically based parser? The answer is that by keeping the levels of grammar and algorithmic realization distinct, it is easier to determine just what is contributing to discrepancies between theory and surface facts. For instance, if levels are kept distinct, then one is able to hold the grammar constant and vary machine architectures to explore the possibility of a good fit between psycholinguistic evidence and model. Suppose these results came to naught. We can then try to covary machine architecture and covering mappings, still seeking model and data compatibility. If this fails, one could then try different grammars. In short, modularity of explanation permits a corresponding modularity of scientific investigation. For a complex information processing system like the language faculty, this may be the investigative method of choice. This has been the research strategy adopted by D. Marr and others in their study of early visual processing, a strategy that has paid off with impressive results:

In a system that solves an information processing problem, we may distinguish four important levels of description.... At the lowest, there is basic component and circuit analysis—how do transistors (or neurons), diodes (or synapses) work? The second level is the study of particular mechanisms: adders, multipliers, and memories, these being assemblies made from basic components. The third level is that of the algorithm, the scheme for a computation; and the top level contains the theory of the computation.... [T]ake the case of Fourier analysis. Here the computational theory of the Fourier transform—the decomposition of an arbitrary mathematical curve into a sum of sine waves of differing frequencies—is well understood, and is expressed independently of the particular way in which it might be computed. One level down, there are several algorithms for computing a Fourier transform, among them the so-called Fast Fourier Transform (FFT), which comprises a sequence of mathematical operations, and the so-called spatial algorithm, a single, global operation that is based on the mechanisms of laser optics. All such algorithms produce the same result, so the choice of which one to use depends upon the particular mechanisms that are available. If one has fast digital memory, adders, and multipliers, then one will use the FFT, and if one has a laser and photographic plates, one will use an "optical" method. (Marr and Poggio 1977:470)

In contrast, a theory that collapses grammar, parser, and machine together cannot decouple these levels of description so as to settle independently what and how questions. Should discrepancies between external facts and theory arise, we must either reformulate the entire theory or backtrack and attempt to extract properties invariant with respect to algorithmic or machine instantiation. Given our current limited understanding, we would expect many such discrepancies between data and model. Thus the "constant reformulation" strategy seems fruitless. The other route amounts to what Marr suggested. We must separate out the levels of abstract theory, algorithm, and implementation to carve out at least roughly the results at each level in a more independent fashion.

Besides the tactical advantage of a modular approach, there are basic questions of psychological interest whose answers refer to grammatical descriptions. For example, most questions of "language learning" are most perspicuously answered by reference to grammars. Confounding even the pessimists, there are even examples connecting the theories of language use and language acquisition.[58] For instance, assume that the theory of extended lexical grammar provides the optimal grammar for English, and that the processing model outlined in Bresnan 1978 is in fact the one embodied by the language faculty.

Now assume that at some stage of acquisition we encounter a new verb, say, *to disambiguate*, in the sentence: *The context disambiguated the meaning of the sentence.* Within the framework of the extended lexical theory the lexicon stores this verb with its functional representation: *disambiguate*: [$NP_1$—$NP_2$]. It is clear that a native speaker could easily recognize the passive counterpart to the above sentence, even without having met an example: *The meaning was disambiguated by the context of the sentence.* How? The extended lexical theory would claim that the lexicon also associates a passive template with the lexical entry for *disambiguate*. But on what basis? Presumably there is some sort of active-passive relation, perhaps captured by a lexical redundancy rule, that constructs the passive entry once the active entry is built. However, note that this active-passive relation is not part of the parsing algorithm itself. The lexicon encodes its effects independently. This relation is statable only at a level of grammatical description. In order to show how new lexical forms are integrated into the parser, a problem of obvious psychological interest, we must refer to a level of grammatical representation. We could cite other such examples drawn from the study of acquisition, but they would take us far beyond the scope of this chapter. Chapter 6 explains in more detail just how linguistic theory can account for actual language acquisition.

Three major points emerge from this investigation of the connection between grammars, parsers, and machine implementations. First, given our limited knowledge about possible machine implementations it is ill advised to specify directly a theory of parsing for natural language without first having a good theory of grammar. Second, the theory of grammar goes a long way to delimit the class of possible parsing algorithms because it specifies the function to be computed by the parser. Finally, some questions relevant to the theory of language use have answers only by direct appeal to a level of grammatical representation. Each of these conclusions points to a single moral. The development of an adequate theory of language use depends on a firm characterization of linguistic knowledge, a grammar. We cannot build a theory of language use directly, but must rely on theories of competence, algorithms, implementations, and the proper mappings between these explanatory levels.