North East Linguistics Society

Volume 18 *Proceedings of NELS 18 -- Volume 1* Article 11

1987

Positing Gaps in a Parallel Parser

Edward Gibson
Carnegie Mellon University

Robin Clark
Carnegie Mellon University

POSITING GAPS IN A PARALLEL PARSER

Edward Gibson and Robin Clark

Carnegie Mellon University

1. Introduction

Serial deterministic models of human sentence processing (e.g. Frazier, 1978; Frazier and Fodor, 1978; Marcus, 1980; Berwick, 1985) have had a great intuitive appeal, especially when contrasted with unlimited parallel processing. A serial deterministic parser, for example, provides a straight-forward account of why people often fail to perceive ambiguity and report a strongly favoured interpretation for sentences which are technically ambiguous. Serial deterministic processing, furthermore, gives a very plausible account of how people are misled into a garden-path parse for certain strings, *e.g.*:

(1) The horse raced past the barn fell.

By virtue of returning all possible grammatical parses for an input sentence, an unlimited parallel parser, by itself, gives us no reason to prefer one parse over another, and, hence, fails to account for why people prefer certain readings of ambiguous input. Since the non-preferred analysis of a garden-path input can be carried along by the parallelism, a parallel parser cannot be misled. Hence, a fully parallel parser provides no obvious account of garden-path phenomena.

Serial deterministic parsers, however, have always had difficulty positing gaps (Fodor, 1979; Clifton and Frazier, forthcoming). A number of strategies for handling gaps have been proposed, none of which have provided a completely satisfactory account. Two early strategies, for example, were First Resort and Last Resort positing. Roughly, the First Resort strategy posits gaps as soon as possible, while the Last Resort strategy waits until

as much of the string has been parsed as possible. Neither strategy can account for all of the sentences in (2):

(2) a. What$_i$ does the man paint [e]$_i$?
b. What$_i$ does the man paint with [e]$_i$?
c. What$_i$ does the man paint [e]$_i$ with his brush ?

Since there is no evidence that any of these sentences is a garden-path, the serial deterministic parser must posit a gap in a certain position if and only if a gap appears in that position in the final parse. No backtracking may take place during the execution of the algorithm.

First, let us assume that the serial parsing model posits gaps as a first resort. If the serial model posits the gap correctly in (2a) then it must posit a gap in (2b) following the word *paint*, since at that point precisely the same words have been processed. A garden path effect results, since the gap belongs after the word *with*. Since sentence (2b) is not a garden-path, this model cannot be psycholinguistically adequate.

To posit gaps correctly in both (2a) and (2b) the serial deterministic parser without look-ahead must choose to posit gaps as a last resort. That is, gaps are only posited when the input sentence is complete and the parser knows that a gap is necessary (by the presence of a *wh*-word, for instance). However, this strategy will not suffice for sentence (2c). Since the last resort parser is successful in parsing sentence (2b), the parse for (2c) can not include a gap following the word *paint*. This is the very place that a gap must occur in sentence (2c). As a result the last resort parser fails to parse (2c) correctly.[1]

While one may be able to imagine various elaborations on the basic serial deterministic algorithm (*e.g.* look-ahead, lexical expectation), we would like to entertain a different hypothesis. Unconstrained parallelism certainly fails to account for a good deal of the basic empirical data. In this paper we will discuss a highly constrained version of parallel processing which accounts for preferred interpretations of structurally ambiguous input and breaks down on garden-path sentences, but which circumvents many of the problems with positing gaps in serial parsers.

The Constrained Parallel Parser (CPP) is a parsing model based on the principles of Government-Binding Theory (Chomsky, 1981); crucially, CPP has no separate grammar rule module containing language-particular rules. Using the CPP model, a separate lexical entry exists for each different argument structure of a word. When a word is input, tree structures for each of its lexical entries are built and placed in the *buffer*, a one cell data structure that holds a parallel list of these trees. CPP contains a second data structure, the *stack*, which contains tree structures as in the buffer. The stack, however, may be more than one cell deep. The parser builds trees in parallel based on possible attachments made between the buffer and the top of the stack.

The parallelism in CPP is controlled by two constraints; the Exclusive Attachment

[1] The parse that posits the gap as in (2d):

(2d) What$_i$ does the man paint with his brush [e]$_i$?

is taken to be an incorrect parse, since the chain consisting of (what$_i$,[e]$_i$) cannot receive Case, because of the adjacency requirement on Case assignment (Stowell, 1981).

Constraint and the Ranked Attachment Constraint. The Exclusive Attachment Constraint requires that if an attachment is possible between two nodes (one on the stack, one in the buffer), then it is made, and all nodes in parallel that did not take part in attachment, either on the stack or in the buffer, are pruned. The Ranked Attachment Constraint prefers some types of attachments over others. As a result of these constraints, garden path effects occur (see Clark, 1987; Gibson, 1987).

The method of positing gaps in the constrained parallel architecture just described is quite simple. Positing an argument or adjunct gap in a given representation takes place if two conditions are met. First, the current representation must contain a nearby constituent in a non-thematic A-position or an A-bar position, where "nearby" is defined in terms of subjacency bounding nodes.[2] Second, the category of the antecedent must match that of the gap (Kayne, 1984). In addition, constraints such as the Extended Projection Principle, the θ–Criterion and the Empty Category Principle further limit the distribution of empty categories. Given these conditions, gaps are posited fairly freely, but few survive the pruning algorithm.

This paper will describe the CPP parsing algorithm, with demonstrations of resultant garden-path effects. The formal gap-positing algorithm follows, with suitable examples. It is hypothesized that the CPP model is both psycholinguistically plausible and simpler than its serial counterparts, and is therefore a preferable model.

2. The Constrained Parallel Parser (CPP)

2.1. $\overline{\text{X}}$ Theory in CPP

The CPP model assumes $\overline{\text{X}}$ Theory as present in (Chomsky, 1986). $\overline{\text{X}}$ Theory has two basic principles: first, each tree structure must have a head; and second, each structure must have a maximal projection. As a result of these principles and other principles built into the parser, (*e.g.*, the θ–Criterion, the Extended Projection Principle, Case Theory), the positions of arguments, specifiers and modifiers with respect to the head of a given structure are limited. In particular, a specifier may only appear as a sister to the one-bar projection below a maximal projection, and the head, along with its arguments, must appear below the one-bar projection. The orders of the specifier and arguments relative to the head is language dependent. For example, the structure of all categories for English is shown in Figure 1. Modifiers are Chomsky-adjoined to the two-bar or one-bar levels, giving one possible structure for a post-head modifier in Figure 2. If there were no specifier in Figure 2 the modifier could have Chomsky-adjoined to either the maximal projection or the one-bar projection. These would be equivalent representations.

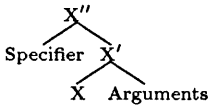


Figure 1: The X-bar structure in English

[2]For simplicity we will assume the bounding node definition of subjacency. That is, two of the nodes {IP, CP, NP} are bounding nodes ({IP, NP} in English), and successive links in a chain may be separated by at most one bounding node (see Chomsky (1981) and the references cited there). See Chomsky (1986) for an alternative approach to subjacency.

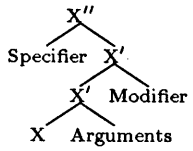


Figure 2: Chomsky-Adjunction to the one-bar level

2.2. Lexical Entries for CPP

A lexical entry accessed by CPP consists of, among other things, a *theta-grid.* A theta-grid is an unordered list of *theta structures.* Each theta structure consists of:

1. a thematic role
2. associated subcategorization information

One theta structure in a theta-grid may be marked as *indirect* to refer to its subject. For example, the word *shout* might have the following theta-grid:[3]

```
((Subcat = NOUN, Thematic-Role = AGENT, INDIRECT)
 (Subcat = PREP, Thematic-Role = GOAL)
 (Subcat = COMP, Thematic-Role = PROPOSITION))
```

When the word *shout* (or a morphological variant of *shout*) is encountered in an input phrase, the thematic role *agent* will be assigned to its subject, as long as this subject is a noun phrase. The direct thematic roles *goal* and *proposition* will be assigned to prepositional and complementizer phrases respectively, as long as each is present. Since the order of theta structures in a theta-grid is not relevant to its use in parsing, the above theta-grid for *shout* will be sufficient to parse both sentences **(3)** and **(4)**.

(3) The man shouts to the woman that Ernie sees the rock
(4) The man shouts that Ernie sees the rock to the woman.

2.3. The CPP Algorithm

The Constrained Parallel Parser parses input by using the buffer and stack data structures along with a simple attachment algorithm. Since CPP has no separate grammar containing language-specific rules, an attachment between a structure in the buffer and a structure on the stack is based on the values associated with parameters of the parser. The parser contains:

1. Constants that are independent of the language being parsed.
2. Parameters that depend on the language being parsed.

For example, the ability to attach structures of one category to structures of another is assumed to be part of unparameterized Universal Grammar. Whether these attachments take place from stack to buffer or from buffer to stack varies according to the type of attachment and the language being considered, however. For example, a determiner phrase, if it exists in a given language, is universally allowable as a specifier of a noun phrase.

[3] In a more complete theory, a syntactic category would be determined from the thematic role.

Whether this attachment is from stack to buffer or from buffer to stack depends on the word order facts concerning specifier attachment for the language being parsed. In English, specifier attachment takes place from stack to buffer, indicating that specifiers occur before the head. As a result, a parameter is set for parsing English which indicates that specifier attachment occurs from stack to buffer. Arguments, on the other hand, are attached from buffer to stack. Hence, English is head-first with respect to arguments. As with the case of Specifiers, this order is the result of setting a parameter which dictates the direction of the attachment of complements with respect to the head.

The formal CPP algorithm is given below, with parameters for attachment set to parse English.

1. (Initializations) Set the STACK to NIL. Set the BUFFER to NIL.
2. (Ending Condition) If the input string is finished and the BUFFER is empty then return the contents of the STACK and stop.
3. If the BUFFER is empty then for each lexical entry corresponding to the next word in the input string, build a maximal projection and put this list of maximal projections into the BUFFER.
4. Make all possible attachments between the STACK and the BUFFER, subject to the attachment constraints. Put the attached structures in the BUFFER. If no attachments are possible, then put the contents of the BUFFER on top of the STACK.
5. Go to 2.

Possible Attachments (parameterized for English):

- Argument Attachment: (BUFFER to STACK) If a structure B in the BUFFER is compatible with the lexical requirements of a structure A, on top of the STACK, then attach B to A as an argument.
- Specifier Attachment: (STACK to BUFFER) If a structure A, on top of the STACK, is compatible as a specifier of a structure B, in the BUFFER, then attach A to B as a specifier.
- Pre-Head Adjunct Attachment: (STACK to BUFFER) If a structure A, on top of the STACK, is compatible as a modifier of a structure B, in the BUFFER, then attach A to B as a modifier.
- Post-Head Adjunct Attachment: (BUFFER to STACK) If a structure B in the BUFFER is compatible as a modifier of a structure A, on top of the STACK, then attach B to A as an modifier.

Attachment Constraints:

- **Exclusive Attachment Constraint**: If an attachment is possible between two structures (one on the stack, one in the buffer), then it is made. All nodes in parallel that did not take part in attachment, either on the stack or in the buffer, are pruned.
- **Ranked Attachment Constraint**: Attachment to on-adjunction positions are preferred over attachments to adjunction positions. That is, attachments to specifier and complement positions are preferred to attachments to modifier positions.

Note that the above algorithm has two attachment constraints that limit the parallelism of the algorithm. Without these constraints, the parser would be parallel in an

unlimited sense, and, as a result, would not be a viable psychological model. It will be shown later in the paper that these universal attachment constraints cause garden-path effects during parses of garden-path sentences, as desired.

To illustrate the algorithm in action, consider the input noun phrase *the man on the rock*. First, a maximal projection for the determiner *the* is placed in the buffer, as seen in Figure 3. Since there is nothing in the stack, no attachments can be made, and as a result, the determiner phrase simply moves to the stack. The second word, *man*, is then read from the input string. Since *man* has both noun and verb entries in the lexicon, a maximal projection for each reading enters the buffer, as shown in Figure 4. Now that both the stack and buffer are non-empty, attachments may be tried. Argument attachment fails, since the structure on top of the stack, the determiner phrase representing *the*, has no lexical requirements. Pre-head modifier attachment fails, since a determiner is not a legal modifier of a noun or a verb. Post-head modifier attachment also fails, since neither a noun nor a verb may modify a determiner. Specifier attachment fails between the determiner and the verb since a determiner may not be the specifier of a verb phrase. Specifier attachment succeeds between the determiner and the noun, however, since a determiner is a possible specifier for a noun phrase.[4] So the determiner phrase representing *the* is attached to the noun phrase representing *man*, the attachment taking place from stack to buffer. The attached structure is then placed in the buffer. Since the verb phrase reading of *man* did not take place in the attachment, it is pruned from the parse by the Exclusive Attachment Constraint. The result is shown in Figure 5.

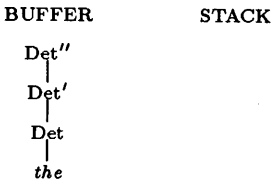


Figure 3: Parse State 1. Input: *the man on the rock*

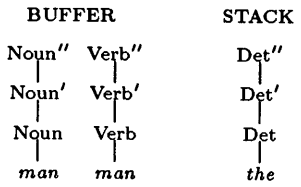


Figure 4: Parse State 3. Input: *the man on the rock*

Since there is now nothing on the stack, the contents of the buffer are moved onto the stack and a maximal projection of the next input word, the preposition *on*, is placed in the buffer. Argument attachment fails, since the noun phrase *the man* has no lexical requirements. Specifier attachment also fails, since a noun phrase is not a legal specifier of a prepositional phrase. Similarly, pre-head modifier attachment fails. Post-head modifier

[4] This is presumably a theorem of Universal Grammar. That is, a determiner may be attached as the Specifier of a noun phrase. Thus, provided that a language has determiners, they will be attached as [Spec, N]. We assume, furthermore, that UG allows only this role for determiners; they cannot be modifiers of VP for example.

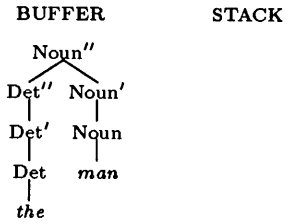


Figure 5: Parse State 4. Input: *the man on the rock*

attachment succeeds, however, since a prepositional phrase is a legal modifier of a noun phrase. This attachment takes place and the the attached structure is dropped in the buffer. Since there is nothing on the stack, this structure is then moved to the stack, as shown in Figure 6.

A maximal projection for the word *the* is then put into the buffer, representing the next word in the input. The tree structure on the stack, representing the input string *the man on*, needs a noun phrase to satisfy its lexical requirements. But argument attachment fails here, since a determiner does not satisfy these requirements. Specifier and modifier attachments also fail in this case and the determiner phrase is put on the stack.

The final word in the input, *rock*, is now read and maximal projections for its lexical entries (assumed to be a noun and a verb) are placed in the buffer (see Figure 7). Neither argument nor modifier attachment is viable here, but specifier attachment succeeds for the determiner and noun yielding a noun phrase *the rock*. The Exclusive Attachment Constraint prunes the verb phrase reading of *rock* since it does not take part in any attachment. As a result, the noun phrase *the rock* ends up in the buffer, with only the noun phrase *the man on* left on the stack. Both specifier and modifier attachments fail at this point, but argument attachment succeeds, since a noun phrase is needed to satisfy the lexical requirements of the preposition *on*. Hence buffer to stack attachment occurs with the result dropped back into the buffer. Since there is now nothing on the stack, this structure moves to the stack and the completed parse for the input *the man on the rock* is returned (see Figure 8).

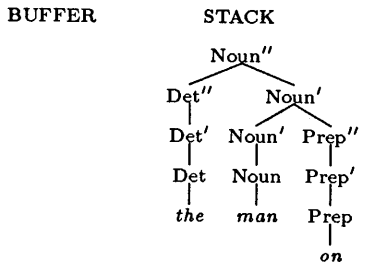


Figure 6: Parse State 8. Input: *the man on the rock*

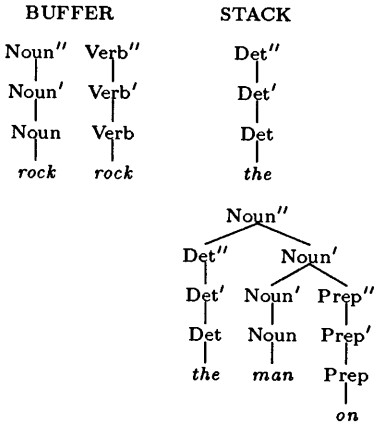


Figure 7: Parse State 11. Input: *the man on the rock*

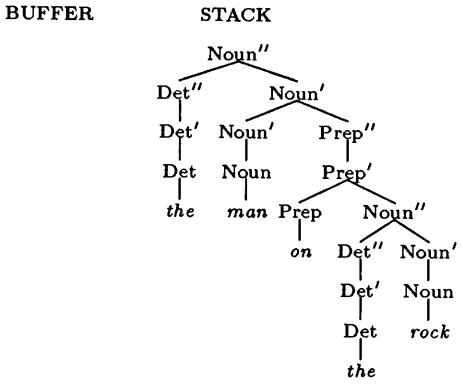


Figure 8: Parse State 14. (Final State) Input: *the man on the rock*

3. Positing Gaps in CPP

There are two distinct kinds of gaps in standard Government-Binding theory: gaps that are base-generated and gaps that appear as a result of movement in the transition from D-structure to S-structure.[5] An example of a gap that is base-generated is PRO, while a gap resulting from movement is known either as a Wh-trace or a NP-trace. PRO and gaps resulting from movement are quite different in nature. For example, PRO cannot be governed, while gaps resulting from movement must be properly governed. As a result of the differences, it is reasonable to deal with the two kinds of gaps separately.

In this paper we will consider only gaps that are a result of movement in the transition from D-structure to S-structure.[6] We will further restrict our attention to those gaps that are caused by movement from argument position. As dictated by the θ–Criterion, all constituents that require thematic roles directly receive these theta-roles at D-structure. A constituent may move from its D-structure position in the transition to S-structure for any of a number of reasons; for example, a lexical noun phrase may move to a Case-marked position to satisfy the Case Filter. Depending on the language in question, *wh* constituents may or may not move in the transition from D-structure to S-structure. If a language allows *wh* movement, then gaps result in positions that are directly assigned both Case and thematic roles.

The moved constituent, along with the gaps coindexed with it, make up a *chain*, which, because of the θ–Criterion, must have exactly one thematic role. If the head of the chain is a lexical noun phrase then the chain must also have Case.

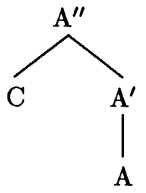
Intuitively, gap–positing should be initiated if, after an attachment takes place, a constituent that needs a thematic role has none associated with it. When looking for a constituent that lacks a thematic role, the gap–positing algorithm does not need to consider argument positions, since the the θ–Criterion disallows movement to θ–governed (complement) positions. For simplification purposes, we will assume that gaps may only be posited in argument and specifier positions, and that specifier gaps may only be posited in non-lexical categories (Infl and Comp). These assumptions are made so that examples will be less complicated; they are not true in general. The first assumption, for instance, is false, since gaps need to be posited in adjunct positions so that sentence (5) may be interpreted as ambiguous. The assumptions, however, simplify examples that do not need adjunct gaps.

(5) When$_i$ [$_{IP}$ did John say [$_{IP}$ Bill left ([e]$_i$)] ([e]$_i$)]?

[5] In *Barriers* Chomsky (1986) also postulates that two types of movement are possible: movement of a head (as in verb movement) and movement of an entire constituent (as in NP or *Wh* movement). In this paper we will consider only movement of full constituents, because of space limitations. Head movement is compatible with the algorithm presented here; see Gibson and Clark, forthcoming, for the application of this algorithm to structures resulting from head movement.

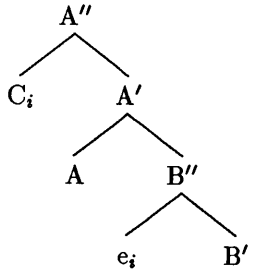
[6] See Gibson and Clark, forthcoming, for a treatment of gaps including PRO.

As a result of these assumptions, gap–positing is initiated in structure (S1) if constituent C has no thematic role, after some attachment occurs to constituent A″:

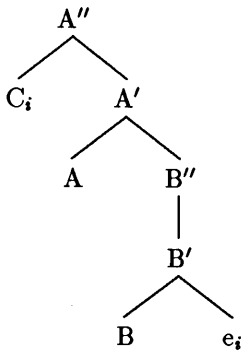


Structure (S1)

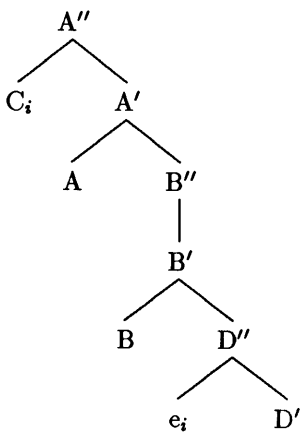
Suppose, for example, that constituent B″ attaches as a sister to A. (In the general case, it is possible that arguments of A have already been attached and that B″ is attached somewhere below one of these arguments.) Since C has no thematic role, gaps are posited everywhere possible in B″. That is, for each position in B″ that is compatible with the syntactic category of C and is subjacent to C, the tree structure A″ is copied, and a trace coindexed with C is posited in that position. This procedure is repeated recursively on each tree structure that is produced. The procedure is terminated for a given tree structure if either a posited gap receives a thematic role, thus completing a chain, or there is no further position in which to posit a gap. If a representation, A, still has a constituent in a $\overline{\theta}$-position after gap–positing has terminated, then gap–positing will need to be re-initiated on A later in the parse in order to find a thematic role for the $\overline{\theta}$-constituent in A. For example, each of the structures below might result from this gap–positing algorithm.



Structure (S2)



Structure (S3)



Structure (S4)

Structure (S2) contains a Wh-trace in the subject position of B″; this is essentially the structure associated with (6a). Structure (S3) contains a Wh-trace in object position. Sentence (6b) is an example of a sentence having such a structure. Structure (S4) contains an NP-trace in subject position. This case represents examples of subject raising, as in (6c), and traces in the specifier of Comp as in (6d).

(6) a. Who$_i$ [e]$_i$ saw the man ?
b. What$_i$ did the man eat [e]$_i$?
c. The man$_i$ seems [e]$_i$ to enjoy the meal.
d. Who$_i$ did the man think [$_{CP}$ [e]$_i$ [Mary liked [e]$_i$]] ?

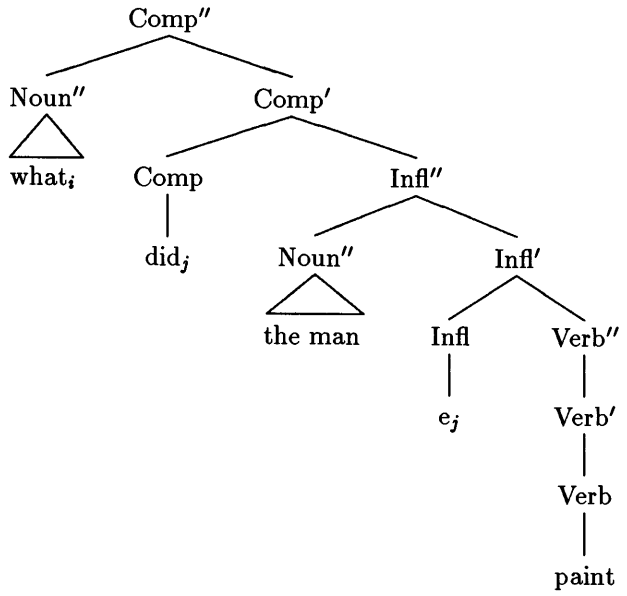
POSITING GAPS IN A PARALLEL PARSER

To illustrate gap–positing in the CPP algorithm, consider sentences (2a–c) once again.

(2) a. What$_i$ does the man paint [e]$_i$?
b. What$_i$ does the man paint with [e]$_i$?
c. What$_i$ does the man paint [e]$_i$ with his brush ?

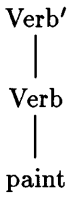
Assume that **paint** subcategorizes for an optional noun phrase (object thematic role), and an optional prepositional phrase (instrument thematic role). Four distinct structures are possible based on this theta-grid: one that has both the noun phrase and prepositional phrase complements (sentence (2c)), two that have only one complement (sentences (2a) and (2b)), and one with no complements.

In parsing one of sentences (2a–c), we will assume for simplicity that the input *what did the man paint* has already been parsed, before gap-positing. The verb phrase *paint* has just been attached to the Comp phrase *what did the man* leaving the structure (S5) in the buffer:

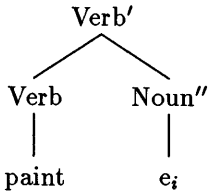


Structure (S5)

Since the noun phrase *what* is in a $\overline{\theta}$-position, gap–positing is initiated. The only possible place to posit a gap is as the object of the verb *paint*. Once this gap is placed, no further gaps can be posited, since the object position receives a thematic role, thus completing the chain. After gap–positing, therefore, two structures remain. Structure (S5) remains (the bottom of which is depicted in structure (S6)) along with a structure having a gap posited in object position of the verb (structure (S7)).



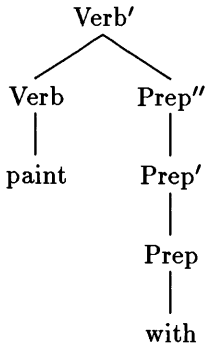
Structure (S6)



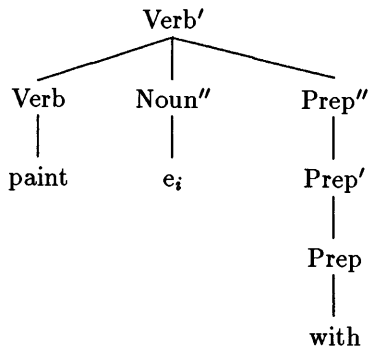
Structure (S7)

If the input finishes at this point, then structure (S6) is pruned for a number of reasons. The *wh*-operator fails to bind a gap, and the noun phrase *what* has no thematic role or Case. Structure (S7), however, remains as a successful parse.

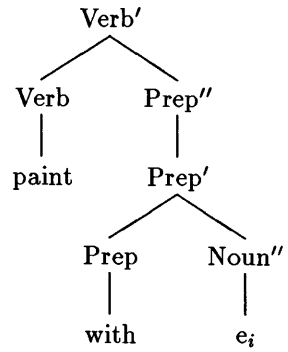
Suppose, on the other hand, that the input is not exhausted, and that the next word in the stream is *with*. Structures (S6) and (S7) move to the stack and a maximal projection for *with* enters the buffer. This prepositional phrase attaches to both structures (S6) and (S7). Gap-positing ensues and structures (S8), (S9) and (S10) result.



Structure (S8)



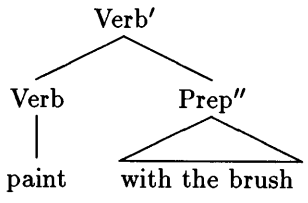
Structure (S9)



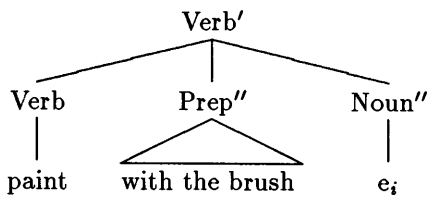
Structure (S10)

If the input finishes at this point, then structures (S8) and (S9) are pruned, since both fail the Extended Projection Principle and θ–Criterion. (Structure (S8) has additional problems: the *wh*-operator fails to bind a gap, and the noun phrase *what* lacks both a thematic role and Case.) Structure (S10) remains as a successful parse.

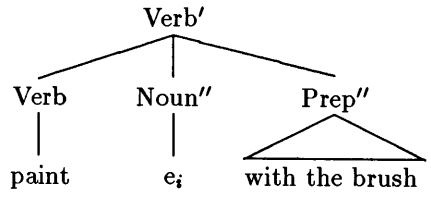
Suppose, however, that the noun phrase *the brush* follows in the input. Then this noun phrase attaches to all structures formed so far and gap-positing is applied, resulting in the following structures:



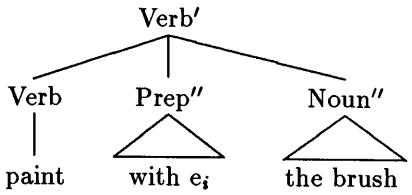
Structure (S11)



Structure (S12)



Structure (S13)



Structure (S14)

Structure (S12) is pruned since it contains a noun phrase chain that lacks Case. Structure (S14) is also pruned since the noun phrase *the brush* lacks Case. If the input finishes at this point, then structure (S11) is pruned, since the *wh*-operator fails to bind a gap, and the noun phrase *what* lacks a thematic role and Case. Structure (S13) remains as a successful parse.

Sentences (2a-c) are all parsed successfully using this simple parallel gap–positing algorithm. Hence positing gaps in a parallel parser does not have the problems of positing gaps in a serial deterministic parser. The advantage that a serial deterministic parsing architecture had over a parallel one was that it more easily explained classic psycholinguistic effects such as garden-path effects. We will now demonstrate, however, that the CPP model also obtains garden-path effects, thus re-establishing parallel models as plausible psychological models.

4. Garden-Path Effects and CPP

Consider the following sentence:

(7) The woman walked to the station ate the cake.

This is a garden-path sentence because *walked to the station* is misanalyzed as matrix level verb phrase. In order to obtain a grammatical sentence, *walked to the station* must be analyzed as a reduced relative clause modifying *the woman*. Consider the state of the parse after the input *the woman walked* in sentence (7) (shown below):

The buffer contains a noun phrase representing the input *the woman*. The stack contains a verb phrase representing the passive participle *walked*, as well as an Infl phrase representing the tensed verb *walked*. Argument attachment fails in this situation, since the noun phrase in the buffer has no lexical requirements. Specifier attachment succeeds between the Infl phrase and the noun phrase, as well as between the verb phrase and the noun phrase. This verb phrase attachment results in small clause formation, but this parse is pruned because the noun phrase *the woman* cannot receive case in this configuration. Crucial to the garden-path effect, modifier attachment fails, because of the Ranked Attachment Constraint. The Ranked Attachment Constraint prefers non-adjunction attachments to adjunction attachments. Since Specifier attachment succeeds in this parse configuration,

modifier attachment (an adjunction) cannot. As a result, the passive participle reading of *walked*, which would result in the reduced relative clause *walked to the store*, is ignored, resulting in the following parse state:

BUFFER **STACK**

A garden-path sentence eventually results, since the tensed Infl phrase representing *ate* cannot attach to the Infl phrase representing *the woman walked to the station* and the Extended Projection Principle is violated.

Because of the two attachment constraints, the CPP model obtains garden-path effects. For further exposition, see (Gibson, 1987). In addition, it will be shown in (Gibson and Clark, forthcoming) that the CPP model obtains Minimal Attachment and Late Closure effects. As a result, a parallel architecture alleviates difficulties of gap–positing, while still capturing classic psycholinguistic effects.

5. Conclusions

We have described a parallel parsing model that, unlike serial deterministic models, has no difficulty positing gaps. The design of the parser follows from current work in syntactic theory. The representations posited by the parser must obey certain constraints (the Case Filter, θ–Criterion, etc.). Furthermore, in keeping with the spirit of recent work in Government–Binding theory (Chomsky, 1985), the parser makes no use of language-particular grammar rules. Given the parser's ability to replicate phenomena like garden-path effects and preferred analyses of ambiguous input, we feel that research along these lines can do much to illuminate the relationship between syntactic knowledge and its use. Finally, we note that the Constrained Parallel Parser is a genuinely parallel parser. Although unlimited parallel parsers can also posit gaps easily, they cannot obtain classic psycholinguistic effects such as those from garden-path data. The CPP model, since it is severely constrained, does not suffer this defect; it obtains garden-path as well as other psycholinguistic effects. Hence, psycholinguists, while correct in rejecting unconstrained parallelism, were too hasty in their outright rejection of parallel algorithms.

Acknowledgements

We would like to acknowledge Rick Kazman, Maryellen Macdonald and Eric Nyberg for their helpful commments on earlier drafts of this paper.

References

Berwick, R., *The Acquisition of Syntactic Knowledge*, MIT Press, Cambridge, MA, 1985.

Bever, T., and McElree, B., *Empty Categories Access Their Antecedents during Comprehension*, Linguistic Inquiry 19, pp. 34–44, 1988.

Chomsky, N., *Lectures on Government and Binding*, Foris, Dordrecht, The Netherlands, 1981.

Chomsky, N., *Knowledge of Language: Its Nature, Origin and Use*, Praeger Publishers, New York, NY, 1985.

Chomsky, N., *Barriers*, Linguistic Inquiry Monograph 13, The MIT Press, Cambridge, Mass., 1986.

Clark, R., *Rules and Parsing*, Talk presented at Massachusetts Institute of Technology, Spring 1987.

Clifton, C., and Frazier, L., *Comprehending Sentences with Long-Distance Dependencies*, to appear in Tanenhaus, M., and Carlson, G., (eds.), *Linguistic Structure in Language Processing*, Reidel.

Fodor, J.D., *Superstrategy*, from Cooper, W., and Walker, E. (eds.), *Sentence Processing: Studies Presented to Merrill Garrett*, Lawrence Erlbaum and Assoc., 1979.

Frazier, L., and Fodor, J.D., *The Sausage Machine: A New Two-stage Parsing Model*, Cognition 6, pp. 291–325, 1978.

Frazier, L., *On Comprehending Sentences: Syntactic Parsing Strategies*, University of Massachusetts Ph.D. dissertation, 1978. Frazier, L., *Syntactic Complexity*, from Dowty, Karttunen and Zwicky (eds.), *Natural Language Parsing*, Cambridge University Press, 1985

Gibson, E.A.F., *Garden-Path Effects in a Parser with Parallel Architecture*, Eastern States Conference on Linguistics, 1987.

Jackendoff, R., *X-bar Syntax: A Study of Phrase Structure*, Linguistic Inquiry Monograph 2, The MIT Press, Cambridge, Mass., 1977.

Kayne, R., *Connectedness and Binary Branching*, Foris, Dordrecht, The Netherlands, 1984.

MacDonald, M., *Processing Binding in Passive Sentences*, Eastern States Conference on Linguistics, 1987.

Marcus, M., *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge, MA, 1980.

Stowell, T., Origins of Phrase Structure, Massachusetts Institute of Technology Ph.D. dissertation, 1981.