Incremental Construction of C- and F-Structure in a LFG-Parser

Hans-Ulrich Block Rudolf Hunze
ZTI INF 3, Siemens AG, München, West Germany

In this paper we present a parser for Lexical Functional Grammar (LFG) which is characterised by incrementally constructing the c- and f-structure of a sentence during the parse. We then discuss the possibilities of the earliest check on consistency, coherence and completeness. Incremental construction of f-structure leads to an early detection and abortion of incorrect paths and so increases parsing efficiency. Furthermore those semantic interpretation processes that operate on partial structures can be triggered at an earlier state. This also leads to a considerable improvement in parsing time. LFG seems to be well suited for such an approach because it provides for locality principles by the definition of coherence and completeness.

1. LFG consists of a context free grammar with rules in the usual format such as (1):

(1) A --> (B) C* {D/E} F

The context free analysis of a sentence is refined by attaching a set of equations to each nonterminal node on the right side of a production. They refer to syntactical features of a constituent such as number (NUM), gender (GEN) etc. and to grammatical functions of a constituent such a subject (SUBJ), object (OBJ) etc., establishing relations between a node on the right side of a production and its predecessor. An f-structure which includes this addditional information is associated with each nonterminal node. F-structures can be regarded as lists of attribute-value pairs. The attributes are the syntactical features and grammatical functions described above. The values can be f-structures as well as symbols such as singular (SG), yes (+), baby etc., or semantic forms, the latter serving as a link between syntax and semantics.
When writing down an equation we use the metavariables ↑ to refer to the parent's node f-structure and ↓ to refer to the node's f-structure. If we look at the equation

```
(2) S -->   NP              VP
         (↑ SUBJ) = ↓      ↑ = ↓
```

this means that the subject of the sentence is the NP's f-structure and that S and VP have the same f-structure.
F-structures can be used to detect ungrammatical sentences and to discard incorrect analysis of a sentence. Three possible reasons may lead to the decision to prune an interpretation of a sentence:
i) the f-structure can be inconsistent.
This is the case when information from different constituents leads to contradictory results. For instance in the sentence "the men dies" the lexical entry for men "men": (↑ PRED) = man , (↑ NUM) = PL, would assure that the NUM-attribute in the NP's f-structure has Pl as its value.On the other hand we have from the lexical entry of dies "dies": (↑ PRED) = (SEM-FORM die (↑SUBJ)), (↑ NUM) = SG and the value of the NUM-attribute of the VP's f-structure would be SG. But the sentence and the VP have the same f-structure from where we get the contradiction that the NUM-attribute of the SUBJ-attribute of the sentence's f-structure is SG and Pl at the same time.
ii) the f-structure can be incoherent.
Some of the attributes such as SUBJ or OBJ are marked as grammatical functions.It is an important principle of the LFG-theory that the value of the verbs PRED-attribute which is always a semantic-form should contain all grammatical functions. In a sentence as "The men dies the apple" we have an OBJ in the sentence but not in the lexical entry of the word "dies". Therefore this sentence is ungrammatical.
iii) the f-structure can be incomplete.
A further principle of the LFG-theory demands that all grammatical roles which appear in the verb's semantic form should be contained in the f-structure of the constituent the verb is part of. In a sentence like "Peter gives" the lexical entry for gives "gives": (↑ PRED) = (SEM-FORM give (↑SUBJ) (↑ OBJ) (↑ OBJ2)) calls for two objects but none of them exists.

2. The parser we have developped is based on Earley's algorithm. It operates on a single ordered set of states to the end of which constantly new states which are still to be worked on are added. A state is a tupel (<tree> <left> <right> <dot> <pred.-list>)

- <tree> is the current parsetree of that path
- <left> is a pointer to the input string the constituent begins with
- <right> is a pointer to the input string that immedeatly follows the constituent
- <dot> marks the current position in the right side of the cf grammar rule
- <pred.-list> is a set of pointers to all preceeding states who's tree nodes might become the mother of the current states' tree.

A tree node is a complex data structure that contains the node's label (i.e. his syntactic category), a list of its daughters and a pointer to the f-structure attached to it.
The basic actions are *predict*, *scan* and *complete* which are close to the definition in Earley (1970). For the construction of the c-structure these actions are augmented in the following way: *predict* creates an empty tree node labeled with the predicted category, *scan* attaches the next input word as the rightmost daughter to the state's <tree>, and *complete* attaches the state's <tree> as the rightmost daughter to all treenodes in the states of the current state's <pred.-list>. For the construction of the f-structure the following augmentations are performed: The <dot> part of a state not only marks the position in the cf-rule's right hand side, but also contains the functional equations associated with that position. When *predict*ing a constituent an empty f-structure is attached to it and incremented by *scann*ing a word or *complet*ing the phrase. The parser then instantiates the up- and down-arrow of the equations with copies of the mother's and daughter's f-structure. (In the former case only the mother's f-structure is needed.) After being augmented by evaluation of the equations the f-structure associated with the up-arrow becomes the f-structure of the new state's tree. As an example we show how the f-structure of the sentence *this man loved Mary* grows.

state of analysis	f-structure
predicting S	$[]_S$
predicting NP	$[]_{NP}$

```
scanning this     [DET = DPRON
                   NUM = SG ]NP
scanning man      [DET = DPRON
                   NUM = SG
                   PRED = MAN ]NP
completing NP     [SUBJ = [DET = DPRON
                           NUM = SG
                           PRED = MAN ]]S
predicting VP     []VP
scanning loved    [TENSE = PAST
                   PRED = love( ↑ SUBJ) ( ↑ OBJ)]VP
predicting NP     []NP
scanning Mary     [ PRED = MARY
                    NUM = SG    ]NP
completing NP     [TENSE = PAST
                   PRED = love( ↑ SUBJ) ( ↑ OBJ)]VP
                   OBJ = [PRED = MARY
                          NUM = SG    ]]VP
completing VP     [SUBJ = [DET = DPRON
                           NUM = SG
                           PRED = MAN]

                   TENSE = PAST
                   PRED = love( ↑ SUBJ) ( ↑ OBJ)
                   OBJ  =  [PRED = MARY
                            NUM = SG ]]S
```

Building f-structures in this incremental way allows ruling out paths that would lead to inappropriate f-structures earlier than in a sequential process that builds c-structure and f-structure.

3.1. When *scan*ning or *complet*ing a cf-grammar rule the parser can detect inconsistencies. Look at the sentence *these* **man** *loved Mary while he was waiting for a bus*. When *scan*ning *man* the parser tries to merge the information from the lexical entry of *man* with the f-structure of the NP-node so far constructed. The inconsistency in number is noticed and the analysis fails effecting a considerable abbreviation of parsing time. On the other hand in the sentence *these men loves Mary while* ...the inconsistency in number can be revealed at the moment the *complet*er tries to attach the VP-node to the S-node. It would be very effective and more plausible under the aspect of a cognitive model of parsing if one could finish the analysis of a sentence like this after *scan*ning the verb *loves*. This would imply that f-structures are partially built on the *predict*or as in:

```
predicting S []S
predicting NP     []NP
scanning these    [DET = DPRON
                   NUM = PL]NP
scanning men      [DET = DPRON
                   NUM = PL
                   PRED = MAN]NP
completing NP     [SUBJ = [DET = DPRON
                           NUM = SG
                           PRED = MAN ]]S
predicting VP     [SUBJ = [DET = DPRON
                           NUM = SG
                           PRED = MAN ]]VP
```

Whereas examples like this at first glance can be taken as an argument to build f-structures on the *predict*or the architecture of the Earley algorithm gives good reason not to do so. Remember that in the Earley algorithm the same completed constituent can not only be attached to one node, but to a set of predecessing nodes. Therefore the *predict*or must not open a new constituent at a certain input position if a constituent of the same type is already there. So if we allow new open constituents to inherit f-structures from their predecessors, the predictor has to check if a constituent of a certain type and with a certain partial f-structure has already been opened at the current position. But checking f-structures is a very costy and clumsy process that should be used sparingly.
Furthermore, if we take into account that the real profits of incremental f-structure building consists in decreasing the combinatorial explosion of c-structurally ambiguous sentences by detecting incorrect paths at an early stage, building f-structures on the *complet*er is not as bad as it seems to be at first glance. If we look for example at the sentence (3) we cannot decide from a purely c-structure oriented point of view which of the structures in (4) is the correct reading.

(3) weil Karl die Bücher seinem Vater gibt
because Charles the books his father gives
because Charles gives the books to his father

(4) a. $[_S$ weil $[_{NP}$ Karl] $[_{VP}$ $[_{NP}$ $[_{NP}$ die Bücher] $[_{NP}$ seinem Vaters]] $[_V$ gibt]]]
b. $[_S$ weil $[_{NP}$ Karl] $[_{VP}$ $[_{NP}$ die Bücher] $[_{NP}$ seinem Vaters] $[_V$ gibt]]]

But if we use f-structure information from a rule like the one in (5) we can decide from completing the NP *seinem Vater* that CASE is not GEN and therefore exclude the second reading.

(5)
$$\begin{array}{llll} NP & \rightarrow & NP & NP \\ & & (\uparrow HEAD) = \downarrow & (\uparrow CASE) = GEN \end{array}$$

Of course, if we would check the equations on the *predict*or the wrong path could be detected earlier, namely when *scan*ning the non-GEN determiner *seinem*. But what would then happen? The parse of the same NP has to continue, this time induced by the correct *$[_{VP}$ $[_{NP}$ die Bücher] $[_{NP}$ seinem* ...-path. It seems that whenever there is a constituent that is inconsistent with one of its predecessors there is good chance that it is consistent with some other element in its predecessor set.

3.2. Whereas the check for consistency is a by-product of building the f-structure, the check for coherence is not as simple as that. Checking coherence as soon as possible augments the efficiency of the parser by an early abortion of incorrect paths. Suppose for a moment that our grammar does not treat adjuncts and take the following lexicon-entries in (6) and rules in (7).

(6) put PRED = (Semform 'PUT < ↑SUBJ > < ↑OBJ > < ↑ON >)
book PRED = 'BOOK)
book PRED = (Semform 'BOOK < ↑ON >)
review PRED = 'REVIEW)
review PRED = (Semform 'REVIEW < ↑ON >)

(7)
$$\begin{array}{lllll} VP \rightarrow & V & (NP) & PP^* & \\ & & (\uparrow OBJ) = \downarrow & (\uparrow (\downarrow PCASE)) = \downarrow & \end{array}$$

$$\begin{array}{llllll} NP \rightarrow & \{ DET & N & PP^* & \mid & PN \} \\ & & & (\uparrow (\downarrow PCASE)) = \downarrow & & \end{array}$$

For the VP in a sentence like (8) the parser would first construct the two partial readings in (9).

(8) He put the book on Chomsky on the table

(9) a. put [the book on Chomsky]
b. put [the book] [on Chomsky]

The attachment of the PP *on the table* now leads to the structures in (10), but only (10b.) is coherent.

(10) a. *put [the book on Chomsky on the table]
not coherent with *book*
b. put [the book on Chomsky] [on the table]
c. *put [the book] [on Chomsky] [on the table]
not coherent with *put*

In a more realistic scenario no such strong constraints can be established as VPs and especially NPs normally not only contain objects but as well a rather large number of adjuncts. But even then checking coherence as soon as possible reduces the number of paths to follow. For example if we assume that in English adjuncts follow prepositional objects, for the sentence in (11) the parser develops 19 readings whereas without a coherence-check it had to pursue 42 different paths. The sum of 19 paths is computed out of the 14 readings of the NP in (12a.) where *on the table* is regarded as an adjunct of NP, plus the 5 readings of the complex NP in (12b.), where *on the table* is regarded as a grammatical function of the VP. (See below for the role of completeness in these examples).

(11) He has put the basket with the flowers for the father of his mother's boyfriend on the table

(12) a. the basket with the flowers for the father of his mother's boyfriend on the table

b. the basket with the flowers for the father of his mother's boyfriend

In languages that have a (partial) SOV ordering, such as German, checking for coherence does not contribute that much to the reduction of the combinatorial complexity. Only in cases like (13) where the accusative case within the PP marks the PP as a grammatical function (of either the verb or the noun) some readings may be excluded without having seen the verb.

(13) daß die Sekretärin den Brief an den Direktor an den Abteilungsleiter weiterleitete
that the typist the letter to the director to the head of the department handed

(14) a. *[den Brief an [den Direktor an den Abteilungsleiter]]
not coherent with *Direktor*

b. *[den Brief an den Direktor an den Abteilungsleiter]
not coherent with *Brief*
c. [[den Brief an den Direktor] an den Abteilungsleiter]

From our considerations above we can extract some more formal principles that are apt to check for coherence as soon as possible in all languages, independently of the position of the verb:
Let F denote the VP's f-structure, Semform be the semantic form associated with the verb, ARG(Semform) the set of its arguments and let Gramfunc be the set of subcategorisable grammatical functions. (This mechanism holds for N and NP, P and PP etc. analogously.)Three cases may occur in the coherence check:
1) Assume that the verb has already been processed and we want to attach a phrase, say an NP to which the equation ($\uparrow$ ATTR) = $\downarrow$ is attached.
coherence condition: If ATTR $\in$ Gramfunc then ATTR $\in$ ARG(Semform)
2) Assume that the verb is being processed.
coherence condition: If ATTR $\in$ Gramfunc and ATTR is an attribute in F then ATTR $\in$ ARG(Semform)
3) Assume that the VP is going to be completed. We are then forced from the trivial equation $\uparrow = \downarrow$ to merge the VP's and S' f-structure that is we must apply the coherence condition as it appears in 2) again.
Finally we want to mention that there is no need for a coherence check of the whole sentence as done in earlier works. It is just sufficient to take the three conditions above into account since in LFG global coherence (and completeness) is defined in terms of local coherence (completeness).

3.3. In a sentence like (15) that is c-structurally ambiguous in two ways the ambiguity concerning the attachment of the PP *on the table* is local within the relative clause (16).

(15) The boy that had put the book on the table came in.

(16) a. [had put [the book on the table]]
b. [had put [the book] on the table]

It cannot be solved either by means of consistency or by means of coherence checking. *On the table* is as good an adjunct for *book* as it is a grammatical function for *put*. It can be solved if we take into account the completeness requirement (see 1.iii). Unfortunately, at the current state of LFG an argument may be merged into a verb's surrounding f-structure at a later state of the parse. Consider for example (17) where the prepositional object (realized as *where*) may be merged into the clause *he put the book* by a long-distance movement equation or by a simple equation of the form ($\uparrow$ FOCUS) = ($\downarrow$ ON).

(17) I don't know [$_{S'}$ where [$_S$ he put the book]].

This implies that whenever the *completer* closes the clause, it can't be guaranteed that its f-structure is complete.

We can circumvent this deficiency by a reinterpretation of the notion of 'bounding node'. We introduce bounding categories and assume that they define strict islands. No equation of the type ($\uparrow$ ...) = ($\downarrow$ ...) may be associated with a bounding category in the grammar. We state the island-principle as follows:

The value of a grammatical function in a bounding node's f-structure is the f-structure of a constituent that is dominated by that bounding node.

We can then formulate the completeness checking mechanism: Let Cat(DOWN) be the category and F(DOWN) be the f-structure of the node that is to be attached in the *completer*. Let ARG(DOWN) be the ARG(Semform) of the PRED of F(DOWN). Let BOUNDINGCATEGORIES be the set of bounding categories in the grammar.
Then, if the completer is called in the situation such that Cat(DOWN) $\in$ BOUNDINGCATEGORIES, then continue only if ARG(DOWN) $\subseteq \{A \mid A$ is an attribute in F(DOWN)$\}$.
We have argued above that in a sentence like (11) above the parser could abort 21 wrong paths under the assumption that in English all PPs that are grammatical functions (i.e. all prepositional objects, PO) precede all adjuncts. As the verb selects *on* as a grammatical function none of the PPs preceding the PP *on the table* can be either adjunct or a grammatical function of the

VP. It is obvious that this restriction only holds if the verb has an obligatory prepositional object that **occurs within the VP**. Unfortunately, at the current state of the development of the LFG-formalism we cannot force the PO to occur within the VP. Even if the verb selects a PO, theoretically, this PO may come into the verbs f-structure by some equation of the form in () or by the trivial equation $\uparrow = \downarrow$.

(18) $(\uparrow \text{VCOMP ON}) = (\uparrow \text{X})$

In practice, however, except in cases of long distance movement, only the SUBJ is merged into the VP's f-structure. This implies, given an adequate treatment of long-distance movement, that VPs form an island except for their SUBJects. We therefore suggest to change the above definition of the completeness-checker in the following way: Let EXTERNALS be the set of grammatical functions that may not be realized within the VP. (trivially EXTERNALS = {SUBJ}).

Then, if the completer is called in the situation such that Cat(DOWN) $\in$ BOUNDINGCATEGORIES, then continue only if ARG(DOWN) - EXTERNALS $\subseteq$ {A | A is an attribute in F(DOWN)}.

4.We hope to have shown that with the aid of locality principles incremental construction of f-structures can achieve an enormous reduction of the ambiguity factor of a sentence.

The problem of wether the f-structure of a sentence is wellformed can be decomposed by applying locality principles. This makes it possible to check the wellformedness of f-structures of phrases and facilitates especially the completeness test for the VP.

A further reduction of the VP's ambiguity can be obtained by additional considerations on the possible order of grammatical functions and adjuncts.

As a last examples, consider:

The boy that has put the basket with the flowers for the father of his mother's boyfriend on the table has forgotten to remove the vase with the flowers for the mother of his sister's boyfriend from the table in the kitchen .

From the theory of catalan numbers we get an ambiguity of 42 for the embedded relative clause .This is multiplied with the ambiguity 132 for the main clause resulting in a c-structure that is 5544 ambiguous. If we exploit all the facilities mentioned above we can reduce the relative clause to be 5 times ambiguous (including completeness check that rules out the 14 readings of (12 a)) whereas the main clause is $10 = 5 \cdot 2$ ambiguous (the factor 2 resulting from *in the kitchen* which can be adjunct to the VP as well as adjunct to the NP) and the ambiguity of the whole sentence decreases to 50.

Literature

Church,K. , Patil,R. 82
Coping with syntactic ambiguity or how to put the block in the box on the table
jacl Vol 8, number 3-4, July-December 1982

Kaplan,R. , Bresnan,J. 82
Lexical-functional grammar: a formal system for grammatical representation
in Bresnan,J., ed., the mental representation of grammatical relations
MIT press series on cognitive theory and mental representation 1982

Acknowledgements
We are greatly indepted to our colleagues Hans Haugeneder and Manfred Gehrke.