The computational complexity of sentence derivation in functional unification grammar

Graeme Ritchie
Department of Artificial Intelligence
University of Edinburgh
Edinburgh EH1 1HN

Abstract

Functional unification (FU) grammar is a general linguistic formalism based on the merging of feature-sets. An informal outline is given of how the definition of derivation within FU grammar can be used to represent the satisfiability of an arbitrary logical formula in conjunctive normal form. This suggests that the generation of a structure from an arbitrary FU grammar is NP-hard, which is an undesirably high level of computational complexity.

1. Functional Unification Grammar

There is not space here to give a full definition of FU grammar (see Kay (1979, 1984, 1985), Ritchie(1984)); the aim is rather to outline how the problem of satisfiability of a propositional logic expression in conjunctive normal form (CNF) can be expressed as a derivation in FU grammar, thereby suggesting that the derivation question in FU grammar is "NP-hard" (Garey and Johnson (1979)). Only those aspects of FU grammar which are relevant to the sketch of the proof will be outlined. The argument here is wholly independent of the generative power discussion in Ritchie(1984).

Functional unification (FU) grammar is a grammatical formalism which allows descriptions of linguistic structures to be expressed as functional descriptions (FDs), which are sets of "features" (attribute-value pairs), and grammatical derivation is expressed in terms of these structures. Within a level of an FD, each feature-name can appear only once; i.e. no feature can appear with two different values. Constituent structure within FDs is indicated as follows. In an FD E, any feature F whose feature-name is listed in the value of the PATTERN feature at the same level of nesting within E is a constituent. Feature-values written in angle-brackets (e.g. <DEFINITE>) are not simple data-values, but are pointers to other positions within the structure. These "paths" indicate a structural position that can be found by starting at the outermost level of nesting and tracing feature-names inward along the path.

An FD E1 is said to be an extension of another FD E2 if there is a sub-structure of E1 which is isomorphic to E2, including identity of feature-names and all feature-values. In determining if E1 is an extension of E2, the comparison process must start at the outermost level.

An FU grammar can be thought of as a set of FDs, each one describing a possible shape for a constituent in the language. A FD F is well-formed with respect to the grammar G if there is an FD E in G such that F is an extension of E, and every constituent of F (see above) is well-formed with respect to G. An arbitrary FD can be used as the initial structure in deriving a fuller FD. Suppose G is a FU grammar, F1 and F2 are FDs. Then F1 derives F2 using grammar G if F2 is well-formed with respect to G, and F2 is an extension of F1.

In the textual representation of an FU grammar, it is normal to represent several similar FDs by writing just one FD containing disjunctive lists of the possible variations between braces (curly brackets). This is an abbreviation for the full set of basic FDs, each corresponding to choosing one item from each disjunctive list.

2. Representing CNF expressions

In representing CNF-satisfiability as FU grammar derivation, we will divide the information contained in the CNF expression between two structures - an FD (which will act as the initial functional description for the derivation) and an FU grammar (with respect to which the derivation is defined). The former encodes, in a very direct way, the structure of the CNF expression, whereas the latter is of a very general form which varies only in size (according to the number of propositional symbols and number of conjuncts in the CNF expression).

Suppose the CNF expression has n propositional symbols P1,..Pn, and k conjuncts. The FU representation will involve the feature-names "CAT", "PATTERN", "P1",...."Pn", "NOT-P1",.....,"NOT-Pn", "C1", "C2",..."Ck", with the feature-values CNF-EXPRESSION, CONJUNCT, TRUE, FALSE, NONE and the k-tuple (C1,...Ck). A conjunct of the CNF expression which mentions the literals A(1), A(2),...A(m) explicitly but omits A(m+1),....A(2n) (each A(i) being either an atomic proposition or a negated atomic proposition) will be represented by an FD of the general form given in (1).

```
[CAT = CONJUNCT                    (1)
 A(1) = <A(1)>
 A(2) = <A(2)>
 ......
 A(m) = <A(m)>
 A(m+1) = NONE
 A(m+2) = NONE
  ......
 A(2n) = NONE
]
```

The whole CNF expression will be represented by an FD of the general form in (2), where each of the feature values for the Ci are representations of the individual conjuncts as described in (1).

```
[ CAT = CNF-EXPRESSION                    (2)
   C1 = .......
   C2 = .......
    ..........
   Ck = .......
]
```

The FU grammar will contain two FDs. The first of these will be an FD representing the overall form of any CNF expression with n symbols and k conjuncts, including a set of disjunctive lists representing all possible choices of truth-values for the propositional variables involved; this will be of the general form in (3).

```
[ CAT = CNF-EXPRESSION                    (3)
  PATTERN = [C1 C2 ......Ck]

  { [P1 = TRUE
     NOT-P1 = FALSE]
    [P1 = FALSE
     NOT-P1 = TRUE]
  }
  { [P2 = TRUE
     NOT-P2 = FALSE]
    [P2 = FALSE
     NOT-P2 = TRUE]
  }
    ..........

  { [Pn = TRUE
     NOT-Pn = FALSE]
    [Pn = FALSE
     NOT-Pn = TRUE]
  }
]
```

The FU grammar also contains a FD which contains a disjunction listing all the possible propositional literals linked to "TRUE", as in (4).

```
[CAT = CONJUNCT                           (4)
  { [P1 = TRUE]
    [P2 = TRUE]
     .......
    [Pn = TRUE]
    [NOT-P1 = TRUE]
    [NOT-P2 = TRUE]
     ......
    [NOT-Pn = TRUE]
  }
]
```

The FD that should be the outcome of the derivation process is one which has truth-values explicitly marked in for some of the literals, in such a way that consistent assignments are given to a propositional symbol and its negation, and each conjunct contains (at least) one literal feature with TRUE as its value. For example, the derivation of an FD from the initial FD and grammar representing the CNF expression

$$(P1 \vee \sim P2) \wedge P3$$

could result in an FD as in (5).

```
[ CAT = CNF-EXPRESSION                    (5)
  PATTERN = [C1 C2]

  P1 = TRUE
  NOT-P1 = FALSE
  P2 = FALSE
  NOT-P2 = TRUE
  P3 = TRUE
  NOT-P3 = FALSE

  C1 = [ CAT = CONJUNCT
         P1 = <P1>
         NOT-P1 = NONE
         P2 = NONE
         NOT-P2 = <NOT-P2>
         P3 = NONE
         NOT-P3 = NONE ]

  C2 = [ CAT = CONJUNCT
         P1 = NONE
         NOT-P1 = NONE
         P2 = NONE
         NOT-P2 = NONE
         P3 = <P3>
         NOT-P3 = NONE]
]
```

It is straightforward to check that this is derivable from the original CNF FD; i.e. this FD is an extension of the FD like (2), and this FD is well-formed w.r.t. a CNF FU grammar like (3) and (4) (since each of its constituents is the extension of some FD in that grammar).

3. Outline of proof

In order to prove that FU derivation is NP-hard, we have to establish that the problem of whether a CNF expression is satisfiable can be reduced by a polynomial-time algorithm to the problem of whether an FD can be the basis of a successful derivation with respect to a grammar. (It is not necessary to establish the reverse reduction - it is not a symmetrical relationship). Thus the following must be established:

1. there is a polynomial time algorithm which converts any CNF expression into the representation outlined above ((1), (2), (3), (4)).

2. the CNF expression is satisfiable if and only if the FD produced by this algorithm leads to a successful derivation w.r.t. the grammar constructed by the algorithm.

It should be intuitively plausible that a polynomial time algorithm exists for the conversion, as the FU representation is so directly related to the CNF formula. (The use of disjunctions in the grammar is relevant here, since the expanded form of the grammar would have an exponential number of entries).

The central result (satisfiability iff derivability) can be proved separately in the two directions:

satisfiability => derivability.

If a CNF expression Q is satisfiable, an FD can be created as in (6), such that, for all i in the range 1 to n:

(a) Xi, Yi are in {TRUE, FALSE}

(b) Xi =/= Yi

and for each j in the range 1 to k:

(a) if a literal A appears in the jth conjunct of Q, then the feature named A in the FD labelled Cj has the value <A>.

(b) if a literal A does not appear in the jth conjunct of Q, then the feature named A in the jth FD has the value NONE.

(c) there is at least one feature value Z(j,i) or W(j,i) which is of the form <A> where the feature-value labelled A at the outer level is TRUE. (i.e. either Z(j,i) is <Pi> and Xi is TRUE, or W(j,i) is <NOT-Pi> and Yi is TRUE.

```
[ CAT = CNF-EXPRESSION                          (6)
  PATTERN = [C1 .. Ck]
  P1 = X1
  NOT-P1 = Y1
  .....
  Pn = Xn
  NOT-Pn = Yn

  C1 = [ CAT = CONJUNCT
         P1 = Z(1,1)
         NOT-P1 = W(1,1)
         P2 = Z(1,2)
         NOT-P2 = W(1,2)
         ....
         NOT-Pn = W(1,n) ]
  ............

  Ck = [ CAT = CONJUNCT
         P1 = Z(k,1)
         NOT-P1 = W(k,1)
         ....
         Pn = Z(k,n)
         NOT-Pn = W(k,n)]
]
```

This is an extension of the original CNF FD (cf.(1),(2)), and is well-formed w.r.t. the FU grammar for CNF expressions ((3),(4)). Hence there is a derivation from the CNF FD and CNF grammar as in (1), (2), (3) and (4).

derivability => satisfiability

If the FD as in (2) above can lead to a successful derivation w.r.t. the grammar containing (3) and (4) above, there must be an FD F such that F is an extension of both (2) and some FD in the grammar. Since (2) contains the feature [CAT = CNF-EXPRESSION], the only grammar FD of which F could also be an extension is one of those represented in (3), containing [CAT = CNF-EXPRESSION] together with features denoting a consistent assignment of truth-values to the Pi (i.e. an FD formed by selecting features from the disjunctive representation in (3)). Thus F must contain sub-structures like (2) and (3) at its outermost level, including the PATTERN = [C1...Ck] feature. Since F is well-formed w.r.t. the grammar, each constituent of F must be well-formed w.r.t. the grammar. F's constituents are exactly the values of the features Ci (as in (1)), so for each of these FDs there must be an FD in the grammar of which the constituent FD is an extension. Since the constituent FDs all include the feature CAT = CONJUNCT, the only grammar FDs pertinent are those which contain CAT = CONJUNCT and a single feature representing an assignment of TRUE to a particular literal (i.e. FDs from (4)). The constituent can be an extension of such a grammar FD only if it also contains the same feature with the feature-value TRUE (since all literals appear in (1), either with NONE or TRUE). This will be possible only if (a) the corresponding literal appeared in that conjunct in the CNF expression (b) the path given in (1) links the feature to a TRUE value at the outer-level. Since the outer level's features represent a consistent truth-value assignment, it follows that the structure of F imposed by the derivation demands that there exist a truth-assignment which satisfies each conjunct.

4. Some consequences of this demonstration

Berwick(1982) provides a similarly semi-formal proof of the NP-hardness of parsing with respect to lexical-functional grammar, a formalism with many similarities to FU grammar, although his proof is radically different from the scheme presented here for FU grammar. Berwick gives an explanation of why computational complexity is relevant to linguistic theory, and why NP-hardness is an undesirable property for a linguistic computation. The fact that derivation in FU grammar is computationally complex is particularly worrying, since it suggests that the obvious generation algorithm outlined in Kay's papers is also NP-hard, even though FU grammar was intended as a computationally useful formalism. The idea that sentence generation (production) is NP-hard is perhaps surprising - parsing has always been viewed as a non-deterministic search process, which might well have exponential complexity for certain types of grammar, but computational linguists have probably viewed sentence generation as a slightly more deterministic process.

References

Berwick,R.C.(1982) Computational Complexity and Lexical Functional Grammar. Pp.97-109, AJCL 8, No.3-4.

Garey,M.R. and Johnson,D.S.(1979) Computers and Intractability - a guide to the theory of NP-completeness. San Francisco : Freeman.

Kay,M.(1979) Functional Grammar. Proceedings of Fifth Annual Meeting of the Berkeley Linguistics Society. Berkeley,CA : U.C. Berkeley.

Kay,M.(1984) Functional Unification Grammar - A Formalism for Machine Translation. Pp.75-78 in Proceedings of COLING-84, Stanford University. (Available from the Association for Computational Linguistics).

Kay,M.(1985) Parsing with Functional Unification Grammar. In "Natural Language Parsing", ed. D.R.Dowty, A.Zwicky and L.Karttunen. Cambridge : Cambridge University Press.

Ritchie,G.D.(1984) Simulating a Turing machine using functional unification grammar. Pp.127-136 in Proceedings of ECAI-84, Pisa.