EXPRESSING DISJUNCTIVE AND NEGATIVE FEATURE CONSTRAINTS WITH CLASSICAL FIRST-ORDER LOGIC.

Mark Johnson,
Cognitive and Linguistic Sciences, Box 1978,
Brown University, Providence, RI 02912.
mj@cs.brown.edu

ABSTRACT

In contrast to the "designer logic" approach, this paper shows how the attribute-value feature structures of unification grammar and constraints on them can be axiomatized in classical first-order logic, which can express disjunctive and negative constraints. Because only quantifier-free formulae are used in the axiomatization, the satisfiability problem is NP-complete.

INTRODUCTION.

Many modern linguistic theories, such as Lexical-Functional Grammar [1], Functional Unification Grammar [12] Generalized Phrase-Structure Grammar [6], Categorial Unification Grammar [20] and Head-driven Phrase-Structure Grammar [18], replace the atomic categories of a context-free grammar with a "feature structure" that represents the syntactic and semantic properties of the phrase. These feature structures are specified in terms of constraints that they must satisfy. Lexical entries constrain the feature structures that can be associated with terminal nodes of the syntactic tree, and phrase structure rules simultaneously constrain the feature structures that can be associated with a parent and its immediate descendants. The tree is well-formed if and only if all of these constraints are simultaneously satisfiable. Thus for the purposes of recognition a method for determining the satisfiability of such constraints is required: the nature of the satisfying feature structures is of secondary importance.

Example: (from [16]). The attribute-value structure (1) has six complex elements labelled e1 ... e6 and two constant elements, singular and third. The complex element e1 has two attributes, subj and pred, the value of which are the complex elements e2 and e3 respectively.

The unification of elements e1 of (1) and e7 of (2) results in the attribute-value structure (3), the

values may be any other element in the structure (including a complex element) and any element can be the value of zero, one or several attributes. Attributes are partial: it need not be the case that every attribute is defined for every complex element. The set of attribute-value structures partially ordered by the subsumption relation (together with an additional entity that every attribute-value structure subsumes) forms a lattice, and the join operation on this lattice is called the unification operation [19].
minimal structure in the subsumption lattice which subsumes both (1) and (2).

If constraints on attribute-value structures are restricted to conjunctions of equality constraints (i.e., requirements that the value of a path of attributes is equal to a constant or the value of another path) then the set of satisfying attribute-value structures is the principal filter generated by the minimal structure that satisfies the constraints. The generator of the satisfying principal filter of the conjunction of such constraints is the unification of the generators of the satisfying principal filters of each of the conjuncts. Thus the set of attribute-value structures that simultaneously satisfy a set of such constraints can be characterized by computing the unification of the generators of the corresponding principal filters, and the constraints are satisfiable iff the resulting generator is not T (i.e., T represents unification failure). Standard unification-based parsers use unification in exactly this way.

When disjunctions and negations of constraints are permitted, the set of satisfying attribute-value structures does not always form a principal filter [11], so the simple unification-based technique for determining the satisfiability of feature structure constraints must be extended. Kasper and Rounds [11] provide a formal framework for investigating such constraints by reviving a distinction originally made (as far as I am aware) by Kaplan and Bresnan [10] between the language in which feature structure constraints are expressed and the structures that satisfy these constraints. Unification is supplanted by conjunction of constraints, and disjunction and negation appear only in the constraint language, not in the feature structures themselves (an exception is [3] and [2], where feature bundles may contain negative arcs).

Research in this genre usually follows a general pattern: an abstract model for feature structures and a specialized language for expressing constraints on such structures are “custom-crafted” to treat some problematic feature constraint (such as negative feature constraints). Table 1 sketches some of the variety of feature structure models and constraint types that previous analyses have used.

This paper follows Kasper and Rounds and most proposals listed in Table 1 by distinguishing the constraint language from feature structures, and restricts disjunction and negation to the constraint language alone. It

| Author          | Model of Feature Structures   | Constraint Language Features       |
|-----------------|-------------------------------|------------------------------------|
| Kaplan and Bresnan [10] | Partial functions            | Disjunction, negation, set-values  |
| Pereira and Shieber [17]   | Information Domain F = [ A → F ] + C |                              |
| Kasper and Rounds [11]     | Acyclic finite automata       | Disjunction                        |
| Moshier and Rounds [14]    | Forcing sets of acyclic finite automata | Intuitionistic negation            |
| Dawar and Vijayashankar [3] | Acyclic finite automata       | Three truth values, negation       |
| Gazdar, Pullum, Carpenter, Klein, Hukari and Levine [7] | Category structures | Based on propositional modal logic |
| Johnson [9]            | “Attribute-value structures”  | Classical negation, disjunction...  |
For all constants $c$ and attributes $a$, $a(c) = \bot$.

For all distinct pairs of constants $c_1, c_2$, $c_1 \neq c_2$.

For all attributes $a$, $a(\bot) = \bot$.

For all constants $c$, $c \neq \bot$.

For all terms $u, v$, $u = v \leftrightarrow (u = v \land u \neq \bot)$

Figure 1: The axiom schemata that define attribute-value structures.

differs by not proposing a custom-built "designer logic" for describing feature structures, but instead uses standard first-order logic to axiomatize attribute-value structures and express constraints on them, including disjunctive and negative constraints. The resulting system is a simplified version of Attribute-Value Logic [9] which does not allow values to be used as attributes (although it would be easy to do this). The soundness and completeness proofs in [9] and other papers listed in Table 1 are not required here because these results are well-known properties of first-order logic.

Since both the axiomatization and the constraints are actually expressed in a decidable class of first-order formulae, viz. quantifier-free formulae with equality, the decidability of feature structure constraints follows trivially. In fact, because the satisfiability problem for quantifier-free formulae is $\mathcal{NP}$-complete [15] and the relevant portion of the axiomatization and translation of constraints can be constructed in polynomial time, the satisfiability problem for feature constraints (including negation) is also $\mathcal{NP}$-complete.

AXIOMATIZING ATTRIBUTE-VALUE STRUCTURES

This section shows how attribute-value structures can be axiomatized using first-order quantifier-free formulae with equality. In the next section we see that equality and inequality constraints on the values of the attributes can also be expressed as such formulae, so systems of these constraints can be solved using standard techniques such as the Congruence Closure algorithm [15], [5].

The elements of the attribute-value structure, both constant and complex, together with an additional element $\bot$ constitute the domain of individuals of the intended interpretation. The attributes are unary partial functions over this domain (i.e. mappings from elements to elements) which are always undefined on constant elements. We capture this partiality by the standard technique of adding an additional element $\bot$ to the domain to serve as the value 'undefined'. Thus $a(x) = \bot$ if $x$ does not have an attribute $a$, otherwise $a(x)$ is the value of $x$'s attribute $a$.

We proceed by specifying the conditions an interpretation must satisfy to be an attribute-value structure. Modelling attributes with functions automatically requires attributes to be single-valued, as required.

Axiom schema (A1) describes the properties of constants. It expresses the requirement that constants have no attributes.

Axiom schema (A2) requires that distinct constant symbols denote distinct elements in any satisfying model. Without (A2) it would be possible for two distinct constant elements, say singular and plural, to denote the same individual.

Axiom schema (A3) and (A4) state the properties of the "undefined value" $\bot$. It has no attributes, and it is distinct from all of the constants (and from all other elements as well - this will be enforced by the translation of equality constraints).

This completes the axiomatization. This axiomatization is finite if the sets of attribute symbols and constant symbols are finite: in the intended computational and linguistic applications this is always the case. The claim is that any interpretation which satisfies all of these

1 The close relationship between quantifier-free formulae and attribute-value constraints was first noted in Kaplan and Bresnan [10].

2 Such a schema is required because we are concerned with satisfiability rather than validity (as in e.g. logic programming).
axioms is an attribute-value structure; i.e. (A1) – (A4) constitute a definition of attribute-value structures.

Example (continued): The interpretation corresponding to the attribute-value structure (1) has as its domain the set \( D = \{ e_1, \ldots, e_6, \text{singular}, \text{third}, \bot \} \). The attributes denote functions from \( D \) to \( D \). For example, \( \text{agr} \) denotes the function whose value is \( 3_\bot \) except on \( e_2 \) and \( e_5 \), where its values are \( e_4 \) and \( e_6 \) respectively. It is straightforward to check that all the axioms hold in the three attribute-value structures given above.

In fact, any model for these axioms can be regarded as a (possibly infinite and disconnected) attribute-value feature structure, where the model’s individuals are the elements or nodes, the unary functions define how attributes take their values, the constant symbols denote constant elements, and \( \bot \) is a sink state.

**Expressing Constraints as Quantifier-Free Formulae.**

Various notations are currently used to express attribute-value constraints: the constraint requiring that the value of attribute \( a \) of (the entity denoted by) \( x \) is (the entity denoted by) \( y \) is written as \( (x \ a) = y \) in PATR-II [19], as \( (x \ a) = y \) in LFG [10], and as \( x(a) = y \) in [9], for example. At the risk of further confusion we use another notation here, and write the constraint requiring that the value of attribute \( a \) of \( x \) is \( y \) as \( a(x) = y \). This notation emphasizes the fact that attributes are modelled by functions, and simplifies the definition of ‘\( \neq \)’.

Clearly for an attribute-value structure to satisfy the constraint \( u = v \) then \( u \) and \( v \) must denote the same element, i.e. \( u = v \). However this is not a sufficient condition: \( \text{num}(x) = \text{num}(y) \) is not satisfied if \( \text{num}(x) \) or \( \text{num}(y) \) is \( \bot \). Thus it is necessary that the arguments of ‘\( \neq \)’ denote identical elements distinct from the denotation of \( \bot \).

Even though there are infinitely many instances of the schema in (A5) (since there are infinitely many terms) this is not problematic, since \( u = v \) can be regarded as an abbreviation for \( u = v \land u \neq \bot \).

Thus equality constraints on attribute-value structures can be expressed with quantifier-free formulae with equality. We use classically interpreted boolean connectives to express conjunctive, disjunctive and negative feature constraints.

Example (continued): Suppose each variable \( x_i \) denotes the corresponding \( e_i, 1 \leq i \leq 11 \), of (1) and (2). Then \( \text{subj}(x_1) = x_2 \), \( \text{number}(x_4) = \text{singular} \) and \( \text{number}(\text{agr}(x_2)) = \text{number}(x_4) \) are true, for example. Since \( e_4 \) and \( e_5 \) are distinct elements, \( x_8 = x_{11} \) is false and hence \( x_8 \neq x_{11} \) is true. Thus ‘\( \neq \)’ means ‘not identical to’ or ‘not unified with’, rather than ‘not unifiable with’.

Further, since \( \text{agr}(x_1) = \bot \), \( \text{agr}(x_1) = \text{agr}(x_1) \) is false, even though \( \text{agr}(x_1) = \text{agr}(x_1) \) is true. Thus \( t = t \) is not a theorem because of the possibility that \( t = \bot \).

**Satisfaction and Unification.**

Given any two formulae \( \phi \) and \( \psi \), the set of models that satisfy both \( \phi \) and \( \psi \) is exactly the set of models that satisfy \( \phi \land \psi \). That is, the conjunction operation can be used to describe the intersection of two sets of models each of which is described by a constraint formula, even if these satisfying models do not form principal filters [11] [9]. Since conjunction is idempotent, associative and commutative, the satisfiability of a conjunction of constraint formulae is independent of the order in which the conjuncts are presented, irrespective of whether they contain negation. Thus the evaluation (i.e. simplification) of constraints containing negation can be freely interleaved with other constraints.

Unification identifies or merges exactly the elements that the axiomatization implies are equal. The unification of two complex elements \( e \) and \( e' \) causes the unification of elements \( a(e) \) and \( a(e') \) for all attributes \( a \) that are defined on both \( e \) and \( e' \). The constraint \( x = x' \) implies \( a(x) = a(x') \) in exactly the same circumstances; i.e. when \( a(x) \) and \( a(x') \) are both distinct from \( \bot \). Unification fails either when two different constant elements are to be unified, or when a complex element and a constant element are unified (i.e. constant-constant clashes and constant-complex clashes). The constraint \( x = x' \) is unsatisfiable under exactly the same circumstances. \( x = x' \) is unsatisfiable when \( x \) and \( x' \) are also required to satisfy \( x = c \) and \( x' = c' \) for distinct constants \( c, c' \), since \( c \neq c' \) by axiom schema (A2). \( x = x' \) is also unsatisfiable when \( x \) and \( x' \) are required to satisfy \( a(x) = t \) and \( x' = c' \).
for any attribute \( a \), term \( t \) and constant \( c' \), since 
\[ a(c') = _t \] by axiom schema (A3).

Since unification is a technique for determining the satisfiability of conjunctions of atomic equality constraints, the result of a unification operation is exactly the set of atomic consequences of the corresponding constraints. Since unification fails precisely when the corresponding constraints are unsatisfiable, failure of unification occurs exactly when the corresponding constraints are equivalent to False.

Example (continued): The sets of satisfying models for the formulae (1') and (2') are precisely the principal filters generated by (1) and (2) above.

(1') \[ \text{subj}(x_1) = x_2 \land \text{agr}(x_2) = x_4 \land \text{number}(x_4) = \text{singular} \land \text{pred}(x_1) = x_3 \land \text{verb}(x_3) = x_5 \land \text{agr}(x_5) = x_6 \land \text{person}(x_6) = \text{third} \]

(2') \[ \text{subj}(x_7) = x_8 \land \text{agr}(x_8) = x_{11} \land \text{pred}(x_7) = x_9 \land \text{verb}(x_9) = x_{10} \land \text{agr}(x_{10}) = x_{11} \]

Because the principal filter generated by the unification of \( e_1 \) and \( e_7 \) is the intersection of the principal filters generated by (1) and (2), it is also the set of satisfying models for the conjunction of (1') and (2') with the formula 
\[ x_1 = x_7 \] (3').

(3') \[ \text{subj}(x_1) = x_2 \land \text{agr}(x_2) = x_4 \land \text{number}(x_4) = \text{singular} \land \text{pred}(x_1) = x_3 \land \text{verb}(x_3) = x_5 \land \text{agr}(x_5) = x_6 \land \text{person}(x_6) = \text{third} \land \text{subj}(x_7) = x_8 \land \text{agr}(x_8) = x_{11} \land \text{pred}(x_7) = x_9 \land \text{verb}(x_9) = x_{10} \land \text{agr}(x_{10}) = x_{11} \land x_1 = x_7 \]

The satisfiability of a formula like (3') can be shown using standard techniques such as the Congruence Closure Algorithm [15], [5]. In fact, using the substitutivity and transitivity of equality, (3') can be simplified to (3''). It is easy to check that (3) is a satisfying model for both (3'') and the axioms for attribute-value structures.

(3'') \[ \text{subj}(x_1) = x_2 \land \text{agr}(x_2) = x_4 \land \text{number}(x_4) = \text{singular} \land \text{pred}(x_1) = x_3 \land \text{verb}(x_3) = x_5 \land \text{agr}(x_5) = x_6 \land x_1 = x_7 \land x_2 = x_5 \land x_3 = x_9 \land x_5 = x_{10} \land x_4 = x_6 \land x_4 = x_7 \]

The treatment of negative and disjunctive constraints is straightforward. Since negation is interpreted classically, the set of satisfying models do not always form a filter (i.e. they are not always upward closed [16]). Nevertheless, the quantifier-free language itself is capable of characterizing exactly the set of feature structures that satisfy any boolean combination of constraints, so the failure of upward closure is not a fatal flaw of this approach.

At a methodological level, I claim that after the mathematical consequences of two different interpretations of feature structure constraints have been investigated, such as the classical and intuitionistic interpretations of negation in feature structure constraints [14], it is primarily a linguistic question as to which is better suited to the description of natural language. I have been unable to find any linguistic analyses which can yield a set of constraints whose satisfiability varies under the classical and intuitionistic interpretations, so the choice between classical and intuitionistic negation may be moot.

For reasons of space the following example (based on Pereira's example [16] demonstrating a purported problem arising from the failure of upward closure with classical negation) exhibits only negative constraints.

Example: The conjunction of the formulae
\[ \text{number}(\text{agr}(x)) = \text{singular} \]
and
\[ \text{agr}(x) = y \land \neg (\text{pers}(y) = 3rd \land \text{number}(y) = \text{singular}) \]
can be simplified by substitution and transitivity of equality and boolean equivalences to
\[ (4') \text{agr}(x) = y \land \text{number}(y) = \text{singular} \land \text{pers}(y) \neq 3rd. \]

This formula is satisfied by the structure (4) when \( x \) denotes \( e \) and \( y \) denotes \( f \). Note the failure of upward closure, e.g. (5) does not satisfy (4'), even though (4) subsumes (5).
An objection that might be raised to this general approach is that classical first-order logic cannot adequately express the inherently "partial information" that feature structures represent. While the truth value of any formula with respect to a model (i.e., an interpretation and variable assignment function) is completely determined, in general there will be many models that satisfy a given formula, i.e. a formula only partially identifies a satisfying model (i.e., attribute-value structure). The claim is that this partiality suffices to describe the partiality of feature structures.

**BIBLIOGRAPHY**

1. Bresnan, J. *The Mental Representation of Grammatical Relations*. 1982 The MIT Press. Cambridge, Mass.

2. Dawar, A. and K. Vijayashanker. *Three-Valued Interpretation of Negation in Feature Structure Descriptions*. University of Delaware Technical Report 90-03. 1989.

3. Dawar, A. and K. Vijayashanker. "A Three-Valued Interpretation of Negation in Feature Structures", in *The 27th Annual Meeting of the Association of Computational Linguistics*, Vancouver, 1989.

4. Dreben, B. and W. D. Goldfarb. *The Decision Problem: Solvable Classes of Quantificational Formulas*. 1979 Addison-Wesley. Reading, Mass.

5. Gallier, J. H. *Logic for Computer Science*. 1986 Harper and Row. New York.

6. Gazdar, G., E. Klein, G. Pullum and I. Sag. *Generalized Phrase Structure Grammar*. 1985 Blackwell. Oxford, England.

7. Gazdar, G., G. K. Pullum, R. Carpenter, E. Klein, T. E. Hukari and R. D. Levine. "Category Structures." *Computational Linguistics*. 14.1:1 – 20, 1988.

8. Gurevich, Y. "The Decision Problem for Standard Classes." *JSL*. 41.2: 460-464, 1976.

9. Johnson, M. *Attribute-Value Logic and the Theory of Grammar*. CSLI Lecture Notes Series. 1988 University of Chicago Press. Chicago.
10. Kaplan, R. and J. Bresnan. “Lexical-functional grammar, a formal system for grammatical representation,” in The Mental Representation of Grammatical Relations, Bresnan ed., 1982 The MIT Press. Cambridge, Mass.

11. Kasper, R. T. and W. C. Rounds. “A logical semantics for feature structures”, in The Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics, Columbia University, New York, 1986, 257-266.

12. Kay, M. “Unification in Grammar,” in Natural Language Understanding and Logic Programming, Dahl and Saint–Dizier ed., 1985 North Holland. Amsterdam, The Netherlands.

13. Maxwell, J. T., III and R. Kaplan. “An Overview of Disjunctive Constraint Satisfaction”, in International Workshop on Parsing Technologies, Pittsburgh, PA., 1989, 18 - 27. Carnegie Mellon.

14. Moshier, M. D. and W. C. Rounds. “A logic for partially specified data structures”, in The ACM Symposium on the Principles of Programming Languages, Munich, Germany, 1987, Association for Computing Machinery.

15. Nelson, G. and D. C. Oppen. “Fast Decision Procedures based on Congruence Closure.” J. ACM. 27.2: 245-257, 1980.

16. Pereira, F. C. N. “Grammars and Logics of Partial Information”, in The Proceedings of the International Conference on Logic Programming, Melbourne, Australia, 1987.

17. Pereira, F. C. N. and S. M. Shieber. “The semantics of grammar formalisms seen as computer languages”, in COLING-84, Stanford University, 1984, 123-129. The Association for Computational Linguistics.

18. Pollard, C. and I. Sag. Information-based Syntax and Semantics, Volume 1. CSLI Lecture Notes. 1987 Chicago University Press. Chicago.

19. Shieber, S. M. An Introduction to Unification-based Approaches to Grammar.