An Interpretation of Negation in Feature Structure Descriptions

Anuj Dawar

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

K. Vijay-Shanker

Department of Computer and Information Science
University of Delaware
Newark, DE 19716

Feature structures are informational elements that have been used in several linguistic theories and in computational systems for natural language processing. A logical calculus has been developed and used as a description language for feature structures. In the present work, a framework in three-valued logic is suggested for defining the semantics of a feature structure description language, allowing for a more complete set of logical operators. In particular, an interpretation of the negation and implication operators is examined within this framework. We extend this approach to interpret descriptions that involve existence (or nonexistence) of values for attributes. A definition of augmented feature structures is proposed, and one particular interpretation of the description language with a negation operator is described. A sound and complete proof system is presented for the logic thus obtained and its computational aspects studied.

1 Introduction

A number of linguistic theories and computational approaches to parsing natural language have employed the notion of associating informational elements, consisting of features and their values, with phrases. Such elements, called *feature structures,* have been used in linguistic theories, such as Generalized Phrase Structure Grammar (GPSG; Gadzar et al. 1985) and Lexical Functional Grammar (Kaplan and Bresnan 1983), and in computational formalisms, such as Functional Unification Grammar (Kay 1979) and PATR-II (Shieber 1984).

Rounds and Kasper introduced a logical formalism to describe feature structures with disjunctive specification (Kasper 1987; Kasper and Rounds 1986; Rounds and Kasper 1986). The language is a form of modal propositional logic. To define the semantics of this language, feature structures are formally defined as *acyclic finite automata.* The detailed definition is given in Section 2. A fundamental property of the semantics is that it is monotonic in the sense that the set of automata satisfying a given formula is *upward-closed* under the operation of subsumption. This is important, because we consider a formula to be only a partial description of a feature structure. This property is precisely formulated in Section 2.

Several researchers have expressed a need for extending this logic to include the operators of negation and implication. These two are related in that, in most logical systems, it is possible to use one to define the other (in the presence of a disjunction operator). In this paper, we shall concentrate on the problem of extending the logic to include negation, while also showing that it yields a satisfactory interpretation of implication.

Karttunen (1984), for instance, provides examples of feature structures in which a negation operator might be useful. For instance, the most natural way to represent the *number* and *person* attributes of a verb such as *sleep* would be to say that it is not third person singular, rather than expressing it as a disjunction of the other possibilities. We express this *agreement* constraint by the following formula:

$$\text{agreement} : \neg(\text{person} : \text{third} \wedge \text{number} : \text{singular}) \quad (1)$$

Pereira (1987) provides the following example formula that expresses the semantic constraint that the subject and object of a clause cannot be coreferential unless the object is a reflexive pronoun:

$$\text{obj : type : reflexive} \vee \neg(\text{subj : ref} \approx \text{obj : ref}) \quad (2)$$

This constraint can, in fact, be represented just as naturally as the following implication:

$$(\text{subj : ref} \approx \text{obj : ref}) \Rightarrow \text{obj : type : reflexive} \quad (3)$$

Similarly, the *feature co-occurrence* constraints in GPSG (Gadzar et al. 1985) include implications of the form $\exists l \Rightarrow \phi$ (where ϕ is some description). While a formula of the form $\exists l$ is not part of the Rounds–Kasper logic, we intend it here as asserting the existence of a feature l in a structure. This would normally be expressed in the Rounds–Kasper formalism by the formula $l{:}NIL$. As we see later, formulae of the kind we have in this example, i.e. in which an existential appears negated, require special treatment and will motivate an extension to the feature structure formalism.

Various interpretations have been suggested that define a semantics for these operators (see Section 3), but none has gained universal acceptance. Pereira (1987) set forth certain properties that any such interpretation should satisfy. We suggested that three-valued logic provides us with a framework appropriate for defining the semantics of a feature description logic (which we will call FDL) that includes a negation operator (Dawar and Vijay-Shanker 1989). We also showed that the three-valued framework (based on Kleene's three-valued logic; Kleene 1952) is powerful enough to express most of the existing definitions of negation and implication. It is therefore possible to compare these different approaches. We also presented one particular three-valued interpretation for FDL, motivated by the approach to negation given by Karttunen (1984), that meets the conditions stated by Pereira.

In the present work, we give an exposition of these results, and we also examine another three-valued interpretation for FDL, obtained by using a modified notion of the feature structures that serve as models. This new interpretation, while preserving the desirable properties of the previous one, also provides a satisfactory semantics for the problematic case, mentioned above, of formulae with a negated existential.

In Section 2 we present an exposition of the Rounds–Kasper logic. In Section 3 we examine some existing approaches to defining the semantics of negation, and we also present the framework of three-valued logic within which we define our own interpretation. In Section 4 we exhibit the modified notion of feature structures as models for FDL, and we give the semantics of FDL in terms of these modified feature structures. Finally, in Section 5, we present a proof system for the language and establish some computational results.

2 Rounds–Kasper Logic

In this section, we take a look at the calculus developed by Rounds and Kasper to describe feature structures. The symbols in the language are taken from two primitive domains:

1. *Atoms* (A), and
2. *Labels* (L).

The set of well-formed formulae (W), is given by:

NIL	
TOP	
a	where $a \in A$
$l : \phi$	where $l \in L$ and $\phi \in W$
$\phi \wedge \psi$	where $\phi, \psi \in W$
$\phi \vee \psi$	where $\phi, \psi \in W$
$p_1 \approx p_2$	where $p_1, p_2 \in L^*$

To define the semantics of this language, feature structures are defined as acyclic finite automata. These are formally defined as follows:

Definition 1. An **acyclic finite automation** is a 7-tuple $A = \langle Q, \Sigma, \Gamma, \delta, q_0, F, \lambda \rangle$, where:

1. Q is a nonempty finite set (of states),
2. Σ is a countable set (the alphabet),
3. Γ is a countable set (the output alphabet),
4. $\delta : Q \times \Sigma \rightarrow Q$ is a finite partial function (the transition function),
5. $q_0 \in Q$ (the initial state),
6. $F \subseteq Q$ (the set of final states),
7. $\lambda : F \rightarrow \Gamma$ is a total function (the output function),[1]
8. the directed graph (Q, E) is acyclic, where pEq iff for some $l \in \Sigma$, $\delta(p, l) = q$,
9. for every $q \in Q$, there exists a directed path from q_0 to q in (Q, E), and
10. for every $q \in F$, $\delta(q, l)$ is not defined for any l.

We can define a partial ordering of information on acyclic finite automata. This partial ordering is given by the subsumption relation, defined as follows:

Definition 2. Given two acyclic finite automata, $A = \langle Q_A, \Sigma_A, \Gamma_A, \delta_A, q_{0A}, F_A, \lambda_A \rangle$ and $B = \langle Q_B, \Sigma_B, \Gamma_B, \delta_B, q_{0B}, F_B, \lambda_B \rangle$, we say that A **subsumes** B ($A \sqsubseteq B$) iff there is a homomorphism from A to B, i.e. there is a mapping $h : Q_A \rightarrow Q_B$ such that:

1. $h(\delta_A(q, l)) = \sigma_B(h(q), l)$,
2. $\lambda_B(h(q)) = \lambda_A(q)$ for all $q \in F_A$, and
3. $h(q_{0A}) = q_{0B}$

Unification, which is the primary information-combining operation on feature structures, can now be simply defined as the operation of finding a least upper-bound (if any upper-bound exists) under the above ordering.[2]

We can now give the semantics of a formula over the set of labels L and the set of atoms A. The domain over which this is done is the set of acyclic finite automata $A = Q, L, A, \delta, q_0, F, \lambda\rangle$. The *satisfies* relation ($\models$) is defined as follows:

Definition 3. An acyclic finite automation $A = (Q, L, A, \delta, q_0, F, \lambda\rangle$ **satisfies** ($\models$) **a formula** in the following cases:

$A \models NIL$ always
$A \models TOP$ never
$A \models a$ iff $Q = F = \{q_0\}$ and $\lambda(q_0) = a$
$A \models l : \phi$ iff $A/l \models \phi$
$A \models \phi \wedge \psi$ iff $A \models \phi$ and $A \models \psi$
$A \models \phi \vee \psi$ iff $A \models \phi$ or $A \models \psi$
$A \models p_1 \approx p_2$ iff $\delta(q_0, p_1) = \delta(q_0, p_2)$

In the above, δ is extended in the standard way to members of Σ^*, i.e. $\delta(q, \epsilon) = q$ and $\delta(q, wl) = \delta(\delta(q, w), l)$ and A/l is the automaton obtained from A by making $\delta(q_0, l)$ the initial state and eliminating all unreachable states.

A fundamental property of the semantics given above is that the set of automata satisfying a given formula is upward-closed under the operation of subsumption. The property is stated in the following theorem (Rounds and Kasper 1986):

Theorem 1. $A \sqsubseteq B$ if and only if for every formula, ϕ, if $A \models \phi$ then $B \models \phi$.

Rounds and Kasper also showed that the satisfiability problem for their logic is NP-complete.

3 Previous Approaches to Negation

In this section we examine the problem of adding a negation operator to the language described in the previous section. We do this by presenting various approaches to defining the semantics of the extended language. We look at these approaches in terms of both their linguistic appropriateness and their computational properties. We will also show that the framework of three-valued logic that we present can be used as a basis for comparison of the different approaches.

3.1 CLASSICAL NEGATION

By classical negation, we mean an interpretation in which an automaton A satisfies a formula $\neg\phi$ if and only if it does not satisfy ϕ. Johnson (1987) defined an Attribute Value Logic (AVL), similar to the Rounds–Kasper Logic, that included a classical form of negation. Smolka (1988) presented a classical semantics for negation in a Rounds–Kasper-like framework. While such approaches are appropriate under one view of feature structures, they are not satisfactory from the viewpoint of feature structures seen as partial descriptions. This is because the crucial property of monotonicity is lost, as can be seen from the following example:

Example 1.

$A =$ [person : second]

$B = \begin{bmatrix} \text{person : second} \\ \text{number : singular} \end{bmatrix}$

$\phi = \neg$(person : second $\wedge$ number : singular)

As can easily be seen, by the classical semantics, $A \models \phi$ and $A \sqsubseteq B$, but $B \not\models \phi$.

Kasper (1988a) discusses an interpretation of negation and implication in an implementation of Functional Unification Grammar that is extended to include conditionals. Kasper's semantics is classical, but his unification procedure uses notions similar to those of three-valued logic.[3] Kasper also localized the effects of negation by disallowing path expressions within the scope of a negation. This restriction may not be linguistically warranted as can be seen from Pereira's formula example in Section 1.

3.2 INTUITIONISTIC LOGIC

Moshier and Rounds (1987) described an extension of the Rounds–Kasper logic, including an implication operator and hence, by extension, negation. The semantics is based on intuitionistic techniques. The notion of satisfying is replaced by one of *forcing*. Given a set of automata K, a formula ϕ, and A such that $A \in K$, A *forces in* K $\neg\phi$ ($A \vdash_K \neg\phi$) if and only if for all $B \in K$ such that $A \sqsubseteq B$, B does not force ϕ in K. Thus, to show that a formula, ϕ, is satisfiable, we have to find a set K and an automaton A such that A *forces in* K ϕ.

Moshier and Rounds also gave a complete proof system for their logic, and showed that the satisfiability problem, while decidable, was PSPACE-complete, thus making it even more intractable than the original Rounds–Kasper logic. Furthermore, Langholm (1989) has shown that not all formulae in the Moshier–Rounds logic can have hereditarily finite sets of minimal models. These computational problems, along with questions about the linguistic appropriateness of its semantics, render the linguistic value of the intuitionistic approach questionable.

3.3 THREE-VALUED LOGIC

Here we take a look at how three-valued logic can be used to define the semantics of FDL. We also take a look at one particular interpretation of FDL that uses the automata of Section 2 as models. This interpretation is essentially the same one we presented earlier (Dawar and Vijay-Shanker 1989). This is an interpretation of negation that is intuitively appealing, formally simple, and computationally no harder than the original Rounds–Kasper logic. The primary intention here (as in our earlier paper) is, however, to explore the use of three-valued logic in defining the semantics of FDL with negation. To this end, we will examine other interpretations also within the three-valued framework. Then, in the next section, we motivate a modified notion of automata models and redefine our interpretation with respect to it.

3.3.1 THE THREE-VALUED FRAMEWORK

With each formula we associate the set (*Tset*) of automata that satisfy the formula, a set (*Fset*) of automata that *contradict* it, and a set (*Uset*) of automata that neither satisfy nor contradict it.[4] The *Uset* contains all automata that are not in either of the other two sets. Different interpretations of negation are obtained by varying definitions of what constitutes "contradiction." The reason for

having some automata that neither satisfy nor contradict a formula is as follows: an automaton is to be viewed as a partial information structure. Given a description (formula), ϕ, a feature structure A may not carry enough information to suggest that it satisfies or falsifies ϕ. However, it may be possible to extend A to either satisfy or falsify ϕ. For example, we will place [*person* : *third*] in the *Uset* of ϕ = (*number* : *singular* $\wedge$ *person* : *third*). Of course, this feature structure can be extended to falsify or satisfy ϕ as in:

$$\begin{bmatrix} number : singular \\ person : third \end{bmatrix}$$

and

$$\begin{bmatrix} number : plural \\ person : third \end{bmatrix}$$

We will define the *Tset* and the *Fset* so that they are upward-closed with respect to subsumption for all formulae. Thus, we avoid the problem of nonmonotonicity associated with the classical interpretation of negation. In our logic, negation is defined so that an automaton A satisfies $\neg\phi$ if and only if it contradicts ϕ.

Formally, the semantics is defined by a partial interpretation function, I. If **WFF** is the set of well-formed formulae of FDL, and **A** the set of acyclic finite automata,[5] the interpretation I is a partial function:

$$I : \mathbf{WFF} \times \mathbf{A} \rightarrow \{True, False\}$$

$I(\phi, A)$ is *True* iff A satisfied ϕ. It is *False* if A contradicts ϕ[6] and is undefined otherwise. Thus, the following hold:

$$Tset(\phi) = \{A | I(\phi, A) = True\} \text{ and}$$
$$Fset(\phi) = \{A | I(\phi, A) = False\}$$

3.3.2 A THREE-VALUED INTERPRETATION

We now look at one such interpretation function that uses the strong Kleene truth definition for conjunction and disjunction.

Definition 4. The **partial interpretation function** I is defined as follows:

1. $I(NIL, A) = True$ for all A;
2. $I(TOP, A) = False$ for all A;
3. $I(a, A) = True$
   if A is atomic and $\lambda(q_0) = a$
   $I(a, A) = False$
   if A is atomic and $\lambda(q_0) = b$
   for some b, $b \neq a$ (see Note 2.)
   $I(a, A)$ is undefined otherwise;
4. $I(l : \phi, A) = I(\phi, A/l)$ if A/l is defined. (see Note 3.)
   $I(l : \phi, A)$ is undefined otherwise;
5. $I(\phi_1 \wedge \phi_2, A) = True$
   if $I(\phi_1, A) = True$ and $I(\phi_2, A) = True$
   $I(\phi_1 \wedge \phi_2, A) = False$
   if $I(\phi_1, A) = False$ or $I(\phi_2, A) = False$
   $I(\phi_1 \wedge \phi_2, A)$ is undefined otherwise;
6. $I(\phi_1 \vee \phi_2, A) = True$
   if $I(\phi_1, A) = True$ or $I(\phi_2, A) = True$
   $I(\phi_1 \vee \phi_2, A) = False$
   if $I(\phi_1, A) = False$ and $I(\phi_2, A) = False$
   $I(\phi_1 \vee \phi_2, A)$ is undefined otherwise;
7. $I(\neg\phi, A) = True$ if $I(\phi, A) = False$
   $I(\neg\phi, A) = False$ if $I(\phi, A) = True$
   $I(\neg\phi, A)$ is undefined otherwise;
8. $I(p_1 \approx p_2, A) = True$
   if $\delta(q_0, p_1)$ and $\delta(q_0, p_2)$ are defined
   and $\delta(q_0, p_1) = \delta(q_0, p_2)$
   $I(p_1 \approx p_2, A) = False$
   if A/p_1 and A/p_2 are both defined
   and are not unifiable
   $I(p_1 \approx p_2, A)$ is undefined otherwise (see Note 4.).

where,

$\phi, \phi_1, \phi_2 \in \mathbf{WFF}$
$A = < Q, L, A, \delta, q_0, F, \lambda > \in \mathbf{A}$
$a, b \in A$
$l \in L$
$p_1, p_2 \in L^*$

NOTES

1. We have not included an implication operator in the formal language, since we find that defining implication in terms of negation and disjunction (i.e. $\phi \Rightarrow \psi \equiv \neg\phi \vee \psi$) yields a semantics for implication that corresponds exactly to our intuitive understanding of implication.
2. As one would expect, an atomic formula is satisfied by the corresponding atomic feature structure. On the other hand, only atomic feature structures are defined as contradicting an atomic formula. An interpretation of negation that defines a complex feature structure as contradiciting a (and hence satisfying $\neg a$) is also possible. Our definition was motivated by the linguistic intention of the negation operator as given by Karttunen (1984), where, for instance, we require that an automaton satisfying the formula *case* : $\neg$*dative* have an atomic value for the *case* feature. However, we now feel that this problem would best be dealt with in a multi-sorted logic and hence, in the interpretation we present in the next section, we have adopted the other alternative mentioned here.
3. In definition 4 above, we state that: $I(l : \phi, A) = I(\phi, A/l)$ if A/l is defined. When A/l is defined, $I(\phi, A/l)$ may still be *True, False,* or undefined. In any of these cases, $I(l : \phi, A) = I(\phi, A/l)$.[7] $I(l : \phi, A)$ is not defined if A/l is not defined (as illustrated by the example given earlier where ϕ = (*person* : *third* $\wedge$ *number* : *singular*). Not only is this condition required to preserve upward closure, it is also linguistically motivated.

In the next section, we will make the distinction in feature structures between *not being defined* versus *cannot be defined.* In this section, we will say that $I(l : \phi, A)$ is not defined if A is not defined for l and $I(l : \phi, A) = false$ if A cannot be defined for l.

4. We have chosen to state that the set of automata that are incompatible with the formula $p_1 \approx p_2$ is *not* the set of automata for which $\delta(q_0, p_1)$ and $\delta(q_0, p_2)$ are defined and $\delta(q_0, p_1) \neq \delta(q_0, p_2)$, since such an automaton could subsume one in which $\delta(q_0, p_1) = \delta(q_0, p_2)$. Thus, we would lose the property of upward closure under subsumption. However, an automaton, A, in which $\delta(q_0, p_1)$ and $\delta(q_0, p_2)$ are defined, and A/p_1 is not unifiable[8] with A/p_2 cannot subsume one in which $\delta(q_0, p_1) = \delta(q_0, p_2)$.

The monotonicity property for the above interpretation can be stated as follows:

Theorem 2. *Tset(ϕ) is upward-closed under the subsumption relation for all formulae ϕ.*

Proof. The proof is by induction on the structure of ϕ and can be found in Dawar 1988.□

We now take a look at some examples mentioned earlier and see how they are interpreted in the logic just defined. The first example expressed the *agreement* attribute of the verb *sleep* by the following formula:

$$agreement : \neg(person : third \wedge number : singular) \quad (4)$$

This formula is satisfied by any structure that has an *agreement* feature which, in turn, either has a *person* feature with a value other than *third,* or a *number* feature with a value other than *singular.* Thus, for instance, the first two structures satisfy the given formula, whereas the third structure is undefined with respect to the formula.

$$\begin{bmatrix} cat : NP \\ agreement : [person : second] \end{bmatrix}$$

$$\begin{bmatrix} agreement : \begin{bmatrix} person : third \\ number : plural \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} cat : NP \\ agreement : [person : third] \end{bmatrix}$$

On the other hand, for a structure to contradict formula (4), it must have an *agreement* feature defined for both *person* and *number* with values *third* and *singular* respectively.

Turning to another example mentioned earlier, the formula:

$$obj : type : reflexive \vee \neg(subj : ref \approx obj : ref) \quad (5)$$

is satisfied by the first two of the following structures, but is contradicted by the third (here co-index boxes are used to indicate co-reference of path-equivalence).

$$[obj : [type : reflexive]]$$

$$\begin{bmatrix} obj : \begin{bmatrix} ref : \boxed{1} \\ type : reflexive \end{bmatrix} \\ subj : [ref : \boxed{1}] \end{bmatrix}$$

$$\begin{bmatrix} obj : \begin{bmatrix} ref : \boxed{1} \\ type : nonreflexive \end{bmatrix} \\ subj : [ref : \boxed{1}] \end{bmatrix}$$

3.3.3 OTHER THREE-VALUED INTERPRETATIONS OF NEGATION

We briefly examine here how the three-valued framework may be used to provide interpretations other than the one presented above.

The classical interpretation of negation can, of course, be expressed by making I a total function such that wherever $I(\phi, A)$ was previously undefined, it is now defined to be *False.*

Moshier and Rounds consider a version in which forcing is always done with respect to the set of all automata, i.e. K^*. This means that the set of feature structures that satisfy $\neg\phi$ is the largest upward-closed set of feature structures that do not satisfy ϕ (i.e. the set of feature structures *incompatible* with ϕ). We can capture this in the three-valued framework described above by modifying the definition of I in the following cases:

- $I(a, A) = True$
  if A is atomic and $\lambda(q_0) = a$
  $I(a, A) = False$ otherwise
- $I(l : \phi, A) = True$
  if A/l is defined and $I(\phi, A/l) = True$
  $I(l : \phi, A) = False$
  if A/l is defined and
  $\forall B(A/l \sqsubseteq B \Rightarrow I(\phi, B) = False)$
  $I(l : \phi, A)$ is undefined otherwise.
- $I(\phi_1 \wedge \phi_2, A) = True$
  if $I(\phi_1, A) = True$ *and* $I(\phi_2, A) = True$
  $I(\phi_1 \wedge \phi_2, A) = False$
  if
  $\forall B(A \sqsubseteq B \Rightarrow I(\phi_1, B) \neq True$ *or* $I(\phi_2, B) \neq True)$
  $I(\phi_1 \wedge \phi_2, A)$ is defined otherwise;
- $I(\phi_1 \vee \phi_2, A) = True$
  if $I(\phi_1, A) = True$ *or* $I(\phi_2, A) = True$
  $I(\phi_1 \vee \phi_2, A) = False$
  if
  $\forall B(A \sqsubseteq B \Rightarrow I(\phi_1, B) \neq True$ *and* $I(\phi_2, B) \neq True)$
  $I(\phi_1 \wedge \phi_2, A)$ is undefined otherwise;
- $I(p_1 \approx p_2, A) = True$
  if $\delta(q_0, p_1)$ and $\delta(q_0, p_2)$ are defined
  and $\delta(q_0, p_1) = \delta(q_0, p_2)$

$I(p_1 \approx p_2, A) = False$
if A/p_1 and A/p_2 are both defined
and are not unifiable or if A is atomic
$I(p_1 \approx p_2, A)$ is undefined otherwise.

As mentioned earlier, our approach was motivated by Karttunen's implementation as described in Karttunen 1984. In the unification algorithm given, negative constraints are attached to feature structures or automata (which themselves do not have any negative values). When the feature structure is extended to have enough information to determine whether it satisfies or falsifies the formula, then the constraints may be dropped. We feel that our definition of the *Uset* captures the notion of associating constraints with automata that do not have sufficient information to determine whether they satisfy or contradict a given formula.

As discussed in Section 3.1, Kasper (1988a) used the operations of negation and implication in extending Functional Unification Grammar. Though the semantics defined for these operators is a classical one, for the purposes of the algorithm Kasper identified three classes of automata associated with any formula: those that *satisfy* it, those that are *incompatible* with it, and those that are *merely compatible* with it. We can observe that these are closely related to our *Tset, Fset,* and *Uset* respectively. For instance, Kasper states that an automaton A satisfies a formula $f: v$ if it is defined for f with value v; it is incompatible with $f: v$ if it is defined for f with value $x (x \neq v)$ and it is merely compatible with $f: v$ if it is not defined for f. In three-valued logic, we incorporate these notions into the formal semantics, thus providing a formal basis for the unification procedure given by Kasper. Our logic also gives a more uniform treatment to the negation operator, since we have removed the restriction that disallowed path equivalences in the scope of a negation.

4 Interpreting FDL with Augmented Feature Structures

We have seen examples (Section 1) of formulae that assert the existence of certain features. While $\exists l$ is not a formula in the Rounds–Kasper syntax, we can regard it as syntactic sugar for the formula $l{:}NIL$, which is indeed satisfied exactly by those automata that have a feature l.

However, the formula $\neg l{:}NIL$ is not satisfiable in the logic we have defined. This is because any automaton that does not have a feature labeled l subsumes one that does. We have, however, seen examples of formulae where $\exists l$ occurs in the scope of a negation (for instance, Kasper [1988b] uses the formula $\exists\, Mood\text{-}type \rightarrow Rank : Clause$). We certainly intend that such formulae be satisfiable.

Since feature structures are partial information structures, if they are not defined for an attribute l, it could be due to lack of information about the value for the attribute l. On the other hand, here we wish to capture the fact that if a feature structure A satisfies the description $\neg\exists l$, then not only is A not defined for l, but it is also the case that it *cannot* be defined for l. That is, it is erroneous to extend A to state a value for the attribute l.

The problem stems from the fact that in the formula $\neg\exists l$, we are trying to capture the information that a feature structure not only does not have a value for the feature l, but cannot be extended to have a value for l; i.e. we have the information that, in the current context, the information structure that we are building is not going to acquire a value for the feature l at any future time. This kind of "negative" information is not expressible in automata models as we have defined them. As they stand, they can only capture "positive" information. To include the negative information we need, we will define an augmented notion of feature structures and redefine our interpretation function accordingly.

To use the analogy with finite state automata, note that in a deterministic fsa we often consider states that do not have outgoing arcs defined on certain labels as having those arcs leading to an "error" state. Since we view fsas as complete structures, this distinction between arcs that are not defined and those that cannot be defined is unimportant. However, when we view our automata models as partial information structures, we must distinguish between the case in which a feature is simply not defined (leaving open the possibility that it may be defined in some extension) and the case in which we know that a certain feature cannot be defined.

In what follows, we capture the information of certain labels leading to "error" states without explicitly defining such states, but by attaching to each state in the structure a finite set of labels. This set contains those labels that cannot be defined from that state. We already have an elementary form of this notion in our restriction on final states, when we specify that they cannot have any outgoing arcs. We are effectively saying that no label can be defined from these states. We formalize all these notions below.

4.1 Augmented Feature Structures

In this section, we give definitions relating to our augmented notion of f-structures. As we stated above, the augmentation consists of attaching to each nonfinal node in the f-structure graph a finite set of labels. These labels are exactly those for which we know that no outgoing arcs can be defined from that node. The set is finite since we require that our information structure at any point be finite. We formally define our extended notion of *f-structure* as follows:[9]

Definition 5. An acyclic finite automaton is an 8-tuple $A = \langle Q, \Sigma, \Gamma, \delta, q_0, F, \lambda, S\rangle$, where:

1. Q is a nonempty finite set (of states),
2. Σ is a countable set (the alphabet),
3. Γ is a countable set (the output alphabet),
4. $\delta: Q \times \Sigma \rightharpoonup Q$ is a finite partial function (the transition function),
5. $q_0 \in Q$ (the initial state),

6. $F \subseteq Q$ (the set of final states),
7. $\lambda: F \rightarrow \Gamma$ is a total function (the output function),
8. $S: Q \backslash F \rightarrow P^{fin(\Sigma)}$ is a function from the nonfinal states to finite subsets of Σ,
9. the directed graph (Q, E) is acyclic, where pEq iff for some $l \in \Sigma$, $\delta(p, l) = q$,
10. for every $q \in Q$, there exists a directed path from q_0 to q in (Q, E),
11. for every $q \in F$, $\delta(q, l)$ is not defined for any l, and
12. whenever $l \in S(q)$, $\delta(q, l)$ is not defined.

We can now define the subsumption ordering on these structures as follows:

Definition 6. Given two f-structures, $A = \langle Q_A, \Sigma_A, \Gamma_A, \delta_A, q_{0A}, F_A, \lambda_A, S_A \rangle$ and $B = \langle Q_B, \Sigma_B, \Gamma_B, \delta_B, q_{0B}, F_B, \lambda_B, S_B \rangle$, we say that A subsumes B ($A \sqsubseteq B$) iff there is a homomorphism from A to B, i.e. there is a mapping $h: Q_A \rightarrow Q_B$ such that:

1. $h(q_{0A}) = q_{0B}$,
2. $h(\delta_A(q, l)) = \delta_B(h(q), l)$,
3. $\lambda_B(h(q)) = \lambda_A(q)$ for all $q \in F_A$, and
4. $S_A(q) \subseteq S_B(h(q))$ for all $q \in Q_A \backslash F_A$.

This definition of subsumption ensures that, for any automaton A, if $l \in S_A(\delta(q_0, p))$ then, for any automaton subsumed by A, the path p is defined, but the path pl cannot be defined.

4.2 THE LANGUAGE

We now give the interpretation of FDL in terms of f-structure models as we have just defined them. The syntax of the language is the same as before.

We first give the following auxiliary definitions:

Definition 7. An f-structure $A = \langle Q, \Sigma, \Gamma, \delta, q_0, F, \lambda, S \rangle$ is:

- **atomic** if and only if $Q = F = \{q_0\}$,
- **null** if and only if $Q = \{q_0\}$ and $F = \varnothing$ and
- **complex** otherwise.

We can now define the revised semantics:

Definition 8. The (revised) **partial interpretation function** I is defined as follows:

1. $I(NIL, A) = True$ for all A;
2. $I(TOP, A) = False$ for all A;
3. $I(a, A) = True$ if A is atomic and $\lambda(q_0) = a$
   $I(a, A) = False$ if A is atomic with $\lambda(q_0) \neq a$ or if A is complex
   $I(a, A)$ is undefined otherwise;
4. $I(l{:}\phi, A) = I(\phi, A/l)$ if A/l is defined.
   $I(l{:}\phi, A) = False$ if $l \in S(q_0)$ or if A is atomic
   $I(l{:}\phi, A)$ is undefined otherwise;
5. $I(\phi_1 \wedge \phi_2, A) = True$
   if $I(\phi_1, A) = True$ and $I(\phi_2, A) = True$
   $I(\phi_1 \wedge \phi_2, A) = False$
   if $I(\phi_1, A) = False$ or $I(\phi_2, A) = False$
   $I(\phi_1 \wedge \phi_2, A)$ is undefined otherwise;
6. $I(\phi_1 \vee \phi_2, A) = True$
   if $I(\phi_1, A) = True$ or $I(\phi_2, A) = True$
   $I(\phi_1 \vee \phi_2, A) = False$
   if $I(\phi_1, A) = False$ and $I(\phi_2, A) = False$
   $I(\phi_1 \vee \phi_2, A)$ is undefined otherwise;
7. $I(\neg\phi, A) = True$ if $I(\phi, A) = False$
   $I(\neg\phi, A) = False$ if $I(\phi, A) = True$
   $I(\neg\phi, A)$ is undefined otherwise;
8. $I(p_1 \approx p_2, A) = True$
   if $\delta(q_0, p_1)$ and $\delta(q_0, p_2)$ are defined
   and $\delta(q_0, p_1) = \delta(q_0, p_2)$
   $I(p_1 \approx p_2, A) = False$
   if A/p_1 and A/p_2 are both defined and are not unifiable or
   $p_1 = wlx$ and $l \in S(\delta(q_0, w))$, or
   $p_2 = wlx$ and $l \in S(\delta(q_0, w))$
   $I(p_1 \approx p_2, A)$ is undefined otherwise (see Note 4).

where,

$\phi, \phi_1, \phi_2 \in \mathbf{WFF}$
$A = \langle Q, L, A, \delta, q_0, F, \lambda \rangle \in \mathbf{A}$
$a, b \in A$
$l \in \mathbf{L}$
$p_1, p_2, w, x \in L^*$

We are now in a position to prove the following monotonicity property for our logic. We express it in terms of the knowledge (or information) ordering $\leq_k$ on the truth values $\{\perp, True, False\}$ defined by $\perp <_k True$, $\perp <_k False$, $True \not\leq_k False$ and $False \not\leq_k True$. In the following, $I(\phi, A) = \perp$ is used for $I(\phi, A)$ undefined.

Theorem 3.1 $A \sqsubseteq B$ if and only if for every formula, ϕ, $I(\phi, A) \leq_k I(\phi, B)$.

Proof. $\Leftarrow$ Suppose for every formula, ϕ, $I(\phi, A) \leq_k I(\phi, B)$. Every path p defined in A must also be defined in B, since $I(p{:}NIL, A) = True$ and hence $I(p{:}NIL, B) = True$. Since for every state q_i in A, there is a path p_i such that $q_i = \delta_A(q_{0A}, p_i)$ we can define a map h such that $h(q_i) = \delta_B(q_{0B}, p_i)$. To see that this map is indeed functional, note that, if there is a $q \in Q_A$ such that $q = \delta_A(q_{0A}, p_1) = \delta_A(q_{0A}, p_2)$ for distinct p_1 and p_2, then $I(p_1 \approx p_2, A) = True$. Thus $I(p_1 \approx p_2, B) = True$ and $\delta_B(q_{0B}, p_1)$ and $\delta_B(q_{0B}, p_2)$ do indeed describe the same state.

One can immediately see that this map satisfies properties 1 and 2 of being a homomorphism given above in the definition of subsumption. To verify the other two conditions, note that if $\lambda_A(q_i) = a$ for some $q_i \in A$, then, $I(p_i{:}a, A) = True$. Hence $I(p_i{:}a, B) = True$ and $\lambda_B(\delta_B(q_{0B}, p_i)) = a$. Thus condition 3 is satisfied. The argument for condition 4 is similar. We have, therefore, established that h is a homomorphism and hence that $A \sqsubseteq B$. $\Rightarrow$ The consequent is trivially true with $I(\phi, A) = \perp$, so we will only consider the case when it is either *True*

or False. The proof is by induction on the structure of the formula.

Basis:

NIL
Trivial, since $I(NIL, A) = True$, for all A.
TOP
Trivial, since $I(TOP, A) = False$, for all A.
a
Note that if A is atomic and $A \sqsubseteq B$, then $A = B$.[10]
Thus, if $I(a, A) = True$, then $A = B$ and we are done.
If $I(a, A) = False$, either A is atomic and the argument is the same as before, or A is complex. But then, since $A \sqsubseteq B$, B is also complex and $I(a, B) = False$.

$p_1 \approx p_2$
If $I(p_1 \approx p_2, A) = True$ then there is a $q \in Q_A$ such that $q = \delta_A(q_{0A}, p_1) = \delta_A(q_{0A}, p_2)$. Let h be a homomorphism witnessing $A \sqsubseteq B$. Then, by the definition of a homomorphism, $h(q) = \delta_B(q_{0B}, p_1) = \delta_B(q_{0B}, p_2)$ and therefore, $I(p_1 \approx p_2, B) = True$.
In the case in which $I(p_1 \approx p_2, A) = False$, we have two possibilities. Either A/p_1 and A/p_2 are both defined and not unifiable, in which case, clearly by the definition of subsumption, the same will be true of B, or $p_1 = wlx$ (choosing p_1 without loss of generality), for some label l and some paths (possibly empty) w and x such that $l \in S_A(\delta_A(q_{0A}, w))$. But then, as we pointed out earlier, this would mean that the path wl and hence the path p_1 cannot be defined in B either. Thus, in either case, $I(p_1 \approx p_2, B) = False$.

Induction Step:

$l{:}\phi$
Since $A \sqsubseteq B$, if A/l is defined, so is B/l and $A/l \sqsubseteq B/l$. But then, by induction hypothesis, $I(\phi, A/l) \leq_k I[\phi, B/l)$ and therefore $I(l:\phi, A) \leq_k I](l{:}\phi, B)$.
If A/l is not defined and $I(l{:}\phi, A) = False$, one of two possible cases applies: either A is atomic, in which case $A = B$ or $l \in S_A(q_0 A)$, in which case $l \in S_B(q_0 B)$ by the definition of subsumption, and we are done.

$\phi \wedge \psi$
If $I(\phi \wedge \psi, A) = True$, then $I(\phi, A) = True$ and $I(\psi, A) = True$. But then, by induction hypothesis, $I(\phi, B) = True$ and $I(\psi, B) = True$. Thus $I(\phi \wedge \psi, B) = True$.
Similarly, if $I(\phi \wedge \psi, A) = False$, $I(\phi, A) = False$ or $I(\psi, A) = False$. Hence, by induction hypothesis, $I(\phi, B) = False$ or $I(\psi, A) = False$, and therefore $I(\phi \wedge \psi, B) = False$.

$\phi \vee \psi$
The argument is similar to the one in the previous case.
$\neg\phi$
Since $I(\neg\phi, A) = True$ if and only if $I(\phi, A) = False$ and vice versa, clearly $I(\neg\phi, A) \leq_k I(\neg\phi, B)$, since $I(\phi, A) \leq_k I(\phi, B)$.□

The following simple corollary corresponds to the monotonicity result we established for our original three-valued semantics.

Corollary. For all ϕ, $Tset(\phi)$ is an upward-closed set.

As we mentioned earlier in this section, Langholm (1989) describes *negatively extended feature structures* in a fashion very similar to what is described above. The interpretation he chooses for the description language is, however, intuitionistic in character. We believe that the modifications that we suggested to our interpretation (in Section 3.3.3) to capture the special case of intuitionistic logic in which forcing is always done with respect to K^*, when applied to our new interpretation yield exactly the interpretation described by Langholm.

5 Proof System

In this section, we give a proof system for the logic described above that is essentially an adaptation of the tableau proof system described by Moshier and Rounds (1987) for their intuitionistic interpretation of the feature logic.

The proof system works, not with individual formulae, but with sets of labeled signed formulae. The Moshier–Rounds tableau proof method worked with sets of sets of labeled signed formulae. However, this extra level of complexity is not needed here.

We first introduce the notion of a labeled signed formula:

Definition 9. A **labeled signed formula** is a triplet $\langle w, X, \phi \rangle$, where $w \in L^*$, $X \in \{True, False\}$ and $\phi \in$ **WFF**. $\langle w, True, \phi \rangle$ will be written as $wT\phi$, and $\langle w, False, \phi \rangle$ as $wF\phi$.

We can now define the notion of an f-structure satisfying a labeled signed formulae:

Definition 10. An f-structure, A, **satisfies** a labeled signed formula Φ (written $A \models \Phi$) in the following cases:
$A \models wT\phi$ if and only if A/w is defined and $I(\phi, A/w) = True$
$A \models wF\phi$ if and only if A/w is defined and $I(\phi, A/w) = False$

Definition 11. A set c of labeled signed formulae is **closed** if and only if at least one of the following holds:

- $wT\phi, wF\phi \in c$,
- $wTa, wTb \in c$,
- $wTa, wxT\phi \in c$,
- $wTa, wTl{:}NIL \in c$,
- $wT(p \approx px) \in c$,
- $wFNIL \in c$, or
- $wTTOP \in c$

for some $l \in L$, $w, p \in L^*$, $x \in L^+$, $a, b \in A$ and $\phi \in$ **WFF**.

Lemma 1. Any closed set of labeled signed formulae is unsatisfiable.

Proof. Immediate from the definition of a closed set.□

Definition 12. A set of labeled signed formulae, c, is **downward-saturated** if and only if c is not closed, and

$wT\neg\phi \in c \Rightarrow wF\phi \in c$
$wF\neg\phi \in c \Rightarrow wT\phi \in c$
$wT(l{:}\phi) \in c \Rightarrow wlT\phi \in c$ and $wTl{:}NIL \in c$
$wF(l{:}\phi) \in c \Rightarrow wlF\phi \in c$ or $wFl{:}NIL \in c$
$wlT(p_1 \approx p_2) \in c \Rightarrow wT(lp_1 \approx lp_2) \in c$
$wlF(p_1 \approx p_2) \in c \Rightarrow wF(lp_1 \approx lp_2) \in c$
$wT(p_1 \approx p_2) \in c \Rightarrow wTp_1{:}NIL \in c$ and $wTp_2{:}NIL \in c$
$wT(p_1 \approx p_2) \in c \Rightarrow wT(p_2 \approx p_1) \in c$
$wT(p_1 \approx p_2), wT(p_2 \approx p_3) \in c \Rightarrow wT(p_1 \approx p_3) \in c$
$wT(p_1 \approx p_2), wp_1T\phi \in c \Rightarrow wp_2T\phi \in c$
$wT(p_1 \approx p_2), wp_1F\phi \in c \Rightarrow wp_2F\phi \in c$
$wF(p_1 \approx p_2), wp_1Ta \in c \Rightarrow wp_2Fa \in c$
$wT(\phi \wedge \psi) \in c \Rightarrow wT\phi \in c$ and $wT\psi \in c$
$wF(\phi \wedge \psi) \in c \Rightarrow wF\phi \in c$ or $wF\psi \in c$
$wT(\phi \vee \psi) \in c \Rightarrow wT\phi \in c$ or $wT\psi \in c$
$wF(\phi \vee \psi) \in c \Rightarrow wF\phi \in c$ and $wF\psi \in c$

Lemma 2. If a finite set of labeled formulae, c, is downward-saturated, it is satisfiable.

Proof. Consider the automaton, $A = \langle Q, L, A, \delta, q_0, F, \lambda, S \rangle$, constructed from c as follows:

1. For every path w for which there is a formula ϕ such that $wT\phi \in c$ or $wF\phi \in c$, include a state q_w in Q, with δ defining a path from q_0 to q_w labeled w.
2. For every pair of paths $p_1 = wx_1$ and $p_2 = wx_2$ such that $wT(x_1 \approx x_2) \in c$, let q_{p1} and q_{p2} be the same state.
3. For every formula $wTa \in c$, include q_w in F and let $\lambda(q_w) = a$.
4. For every formula $wFa \in c$, if there is no label l such that there is a state $q_{wl} \in Q$, then include q_w in F and let $\lambda(q_w) = b$ for any atomic value b such that b does not occur in any formula in c.[11]
5. For every formula $wFl{:}NIL \in c$, include l in $S(q_w)$
6. For every formula $wF(p_1 \approx p_2) \in c$, if states q_{wp_1} and q_{wp_2} are defined and neither of them is in F, then add new states q_1 and q_2 to Q and F, and for some label l that does not occur in any formula of c, define $\delta(q_{wp_1}, l) = q_1$ and $\delta(q_{wp_2}, l) = q_1$ with $\lambda(q_1) = a$ and $\lambda(q_2) = b$ where a and b are distinct atomic values. If exactly one of the two states (say, q_{wp_1}) is not in F, add just one new state q to Q and let $\delta(q_{wp_1}, l) = q$ for a label l that does not occur in c.
   If, however, one of the paths (say, p_1) does not have the associated state (q_{wp_1}) defined, let p be the longest prefix of p_1 such that q_{wp} does exist, and let $p_1 = plx$. Include l in $S(q_p)$.

Claim 1:

The above construction of an automaton is well defined.

We need to verify that the above definition yields an automaton that meets our definition of an acyclic finite automaton without any conflicts. The possible conflicts that could arise would be that: λ does not define a function; the graph of the automaton had a cycle; for some $q \in F$ and some label l, $\delta(q, l)$ is defined; or, for some state q and some label l, $l \in S(q)$ and $\delta(q, l)$ is defined. However, in each of these cases, it is easy to see that were it to arise in the construction given above, the original set c would in fact be closed, contradicting the hypothesis that it is downward-saturated.

Claim 2:

The automaton so constructed satisfies all the labeled signed formulae in c.

We establish this claim by induction over the structure of the formulae in c. For the base cases (namely labeled signed formulae of the forms: wXa, $wXNIL$, $wFl{:}NIL$, and $wX(p_1 \approx p_2)$) it follows immediately from the construction that they are satisfied by A. For the other cases ($wX(\phi \vee \psi)$, $wX(\phi \wedge \psi)$, and $wX\neg\phi$), their sub-formulae are also in c since it is downward-saturated. But by the induction hypothesis, these sub-formulae are satisfied by A. That completes the result.□

The entailment relation ($\vdash$) on sets of labeled signed formulae is defined as follows:

Definition 13. Let c and d be two sets of labeled signed formulae. Then $c \vdash d$ if and only if $c \neq d$ and one of the following holds:

1. $wT\neg\phi \in c$ and $d = c \cup \{wF\phi\}$
2. $wF\neg\phi \in c$ and $d = c \cup \{wT\phi\}$
3. $wTl{:}\phi \in c$ and $d = c \cup \{wlT\phi, wTl{:}NIL\}$
4. $wFl{:}\phi \in c$ and $d = c \cup \{wlF\phi\}$
5. $wFl{:}\phi \in c$ and $d = c \cup \{wFl{:}NIL\}$
6. $wlT(p_1 \approx p_2) \in c$ and $d = c \cup \{wT(lp_1 \approx lp_2)\}$
7. $wlF(p_1 \approx p_2) \in c$ and $d = c \cup \{wF(lp_1 \approx lp_2)\}$
8. $wT(p_1 \approx p_2) \in c$ and $d = c \cup \{wTp_1{:}NIL, wTp_2{:}NIL\}$
9. $wT(p_1 \approx p_2) \in c$ and $d = c \cup \{wT(p_2 \approx p_1)\}$
10. $wT(p_1 \approx p_2), wT(p_2 \approx p_3) \in c$ and $d = c \cup \{wT(p_1 \approx p_3)\}$
11. $wT(p_1 \approx p_2), wp_1T\phi \in c$ and $d = c \cup \{wp_2T\phi\}$
12. $wT(p_1 \approx p_2), wp_1F\phi \in c$ and $d = c \cup \{wp_2F\phi\}$
13. $wF(p_1 \approx p_2), wp_1Ta \in c$ and $d = c \cup \{wp_2Fa\}$
14. $wT(\phi \wedge \psi) \in c$ and $d = c \cup \{wT\phi, wT\psi\}$
15. $wF(\phi \wedge \psi) \in c$ and $d = c \cup \{wF\phi\}$
16. $wF(\phi \wedge \psi) \in c$ and $d = c \cup \{wF\psi\}$
17. $wT(\phi \vee \psi) \in c$ and $d = c \cup \{wT\phi\}$
18. $wT(\phi \vee \psi) \in c$ and $d = c \cup \{wT\psi\}$
19. $wF(\phi \vee \psi) \in c$ and $d = c \cup \{wF\phi, wF\psi\}$

We denote by $\vdash^*$ the reflexive and transitive closure of this entailment relation.

Theorem 4. (Soundness) If $c \vdash^* d$ for sets of labeled signed formulae c and d, and d is downward-saturated, then c is satisfiable.

Proof. This follows immediately from Lemma 2 and the fact that $c \vdash^* d$ implies $c \subseteq d$.□

Lemma 3. For any set of labeled signed formulae c, there are only finitely many sets of labeled signed formulae d such that $c \vdash^* d$.

Proof. To prove this, we inductively define the notion of length of a formula, as follows:
$ln(a) = ln(NIL) = ln(TOP) = 1$
$ln(p_1 \approx p_2) = length(p_1) + length(p_2)$
$ln(\neg\phi) = ln(\phi) + 1$
$ln(\phi \vee \psi) = ln(\phi \wedge \psi) = ln(\phi) + ln(\psi) + 1$
where *length* denotes string length.

Also, define the length of a labeled signed formula $wX\phi$ as $ln(\phi) + length(w)$. Let Φ be any labeled signed formula such that $\Phi \notin c$ but $\Phi \in d$ for some d such that $c \vdash^* d$. Observe that the length of Φ is bounded by the length of the longest formula in c and that Φ does not contain any symbols that do not occur in c. The result follows.□

Lemma 4. For any set of labeled signed formulae c, if there is no set of labeled signed formulae d such that $c \vdash d$, then c is either closed or downward-saturated.

Proof. Clearly, if c is closed under all the entailment rules listed above, then it satisfies all the implications listed in the definition of downward saturation. Hence, if it is not downward-saturated, it must be closed.□

Lemma 5. For any satisfiable set of labeled signed formulae c that is not downward-saturated, there is a satisfiable set of labeled signed formulae d such that $c \vdash d$.

Proof. Since c is satisfiable, it is not closed. Since it is not downward-saturated, by hypothesis, there must be a d such that $c \vdash d$. However, it is clear from the definition of entailment that if all such d are unsatisfiable, then so is c.□

Theorem 5. (Completeness) For any satisfiable set of labeled signed formulae, c, there is a downward-saturated set of labeled signed formulae d such that $c \vdash^* d$.

Proof. By Lemma 3, there must be a d such that $c \vdash^* d$ and for no d' $d \vdash d'$. All such d are either closed or downward-saturated by Lemma 4. However, not all of them can be closed since then by Lemma 5, c would be unsatisfiable. Hence, at least one of them is downward-saturated.□

Theorem 6. (NP-Completeness) The satisfiability problem for the logic we have defined is NP-Complete.

Proof. It follows from the proof of Lemma 3 that the length of any derivation $c \vdash^* d$ is bounded by n^2, where n is the sum of the lengths of the formulae in c. Since this bound is polynomial, the problem is in *NP*. It is *NP*-hard because the satisfiability problem for the Rounds–Kasper logic, which is a special case, is *NP*-hard.□

6. Conclusions

A logical formalism with a complete set of logical operators has come to be accepted as a means of describing feature structures. While the intended semantics of most of these operators is well understood, the negation and implication operators have raised some problems, leading to a variety of approaches in their interpretation.

In Dawar and Vijay-Shanker 1989 and the present work, we introduced the framework of three-valued logic as a means of defining the semantics of a feature structure description language with negation. This framework permits us to say that a formula such as $\neg l:\phi$ does not have a truth value defined in a feature structure that does not have a feature l. This enables us to define an interpretation that, unlike the classical approach to negation, is monotonic, as a logic describing partial structures should be.

We presented one particular interpretation of FDL within this three-valued framework and compared it with other approaches to defining the semantics of negation. We showed that several different such approaches could be cast in the three-valued framework. In particular, we showed that the special case of the Moshier–Rounds intuitionistic approach, in which forcing is always considered with respect to K^* could be captured in our framework.

One motivation cited by Moshier and Rounds for considering forcing sets other than K^* was so that formulae of the form $\neg l:NIL$ could be considered satisfiable. The same reason led us to examine an augmented notion of feature structure models for FDL that yields an interpretation that is conceptually simple, motivated by the preservation of monotonicity, and is computationally no harder than the original Rounds–Kasper logic. We also showed that our interpretation meets the conditions set out by Pereira (1987) for a satisfactory interpretation of negation.

References

Dawar, A. 1988 *The Semantics of Negation in Feature Structure Descriptions.* Master's Thesis, University of Delaware, Newark, DE.

Dawar, A. and Vijay-Shanker, K. 1989 "A Three-Valued Interpretation of Negation in Feature Structure Descriptions." In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics:* 18–24.

Gazdar, G.; Klein, E.; Pullum, G.; and Sag, I. 1985 *Generalised Phrase Structure Grammar.* Harvard University Press, Cambridge, MA.

Johnson, M. 1987 *Attribute Value Logic and the Theory of Grammar.* Ph.D. Thesis, Stanford University, Stanford, CA.

Karttunen, L. 1984 "Features and Values." In *Proceedings of the Tenth International Conference on Computational Linguistics.* 28–33.

Kasper, R. T. 1987 *Feature Structures: A Logical Theory with Application to Language Analysis.* Ph.D. Thesis, University of Michigan, Ann Arbor, MI.

Kasper, R. T. 1988a "Conditional Descriptions in Functional Unification Grammar." In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics:* 233–240.

Kasper, R. T. 1988b "An Experimental Parser for Systemic Grammars." In *Proceedings of the Twelfth International Conference on Computational Linguistics:* 309–312.

Kay, M. 1979 "Functional Grammar." In *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistics Society:* 142–158.

Kaplan, R. and Bresnan, J. 1983 "Lexical Functional Grammar: a Formal System for Grammatical Representation." In J. Bresnan (ed.), *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, MA.

Kleene, S. C. 1952 *Introduction to Metamathematics.* Van Nostrand, New York, NY.

Kasper, R. T. and Rounds, W. C. 1986 "A Logical Semantics for Feature Structures." In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics:* 257–266.

Langholm, T. 1989. *How to Say No with Feature Structures.* Personal communication.

Moshier, M. D. and Rounds, W. C. 1987 "A Logic for Partially Specified Data Structures." In *ACM Symposium on the Principles of Programming Languages:* 156–167.

Pereira, F. C. N. 1987. "Grammars and Logics of Partial Information." In J. L. Lassez (ed.), *Proceedings of the 4th International Conference on Logic Programming.* 989–1013.

Rounds, W. C. and Kasper, R. T. 1986 "A Complete Logical Calculus for Record Structures Representing Linguistic Information." In *IEEE Symposium on Logic in Computer Science:* 34–43

Shieber, S. M. 1984 "The Design of a Computer Language for Linguistic Information." In *Proceedings of the Tenth International Conference on Computational Linguistics:* 362–366.

Smolka, G. 1988 *A Feature Logic with Subsorts.* LILOG report 33, IBM Deutschland, Stuttgart, F. R. G.

NOTES

1. In the original Rounds–Kasper formulation, the output function is not required to be total. This is because every terminal node in the transition graph is considered to be a final state. However, since the notion of finality of a state is not crucial to the formalism, we have chosen this equivalent alternative for presentation.
2. Strictly speaking, we should be taking the least upper bound in the ordering on equivalence classes of automata under isomorphism.
3. See Section 3.3.3.
4. A similar notion was used by Kasper (1988a), who introduces the notion of *compatibility*. We shall compare this approach with ours in greater detail in Section 3.3.3.
5. In this paper we will not consider cyclic feature structures.
6. And therefore it satisfies the formula $\neg\phi$.
7. Equality here is strong equality (i.e. if $I(\phi, (A/l)$ is undefined then so is $I(l{:}\phi, A)$).
8. Two automata are not unifiable if and only if they do not have a least upper bound.
9. Langholm (1989) has defined a similar notion of *negatively extended feature structures*. We will take up a comparison of his approach with ours later in this section.
10. Up to isomorphism.
11. We are implicitly assuming that the sets of atoms and labels are both infinite. If this is not the case, the definition of *closure* of a set of labeled signed formulae and this construction can be suitably modified.