DISJUNCTIVE FEATURE STRUCTURES AS HYPERGRAPHS

JEAN VÉRONIS

Groupe Représentation et Traitement des Connaissances,
Centre National de la Recherche Scientifique,
31, Chemin Joseph Aiguier, 13402 Marseille Cedex 09, France

and

Department of Computer Science, Vassar College
Poughkeepsie, New York 12601, U.S.A.

e-mail : veronis@vassar.edu

Abstract -- In this paper, we present a new mathematical framework in which disjunctive feature structures are defined as directed acyclic hypergraphs. Disjunction is defined in the feature structure domain, and not at the syntactic level in feature descriptions. This enables us to study properties and specify operations in terms of properties of, or operations on, hypergraphs rather than in syntactic terms. We illustrate the expressive power of this framework by defining a class of disjunctive feature structures with interesting properties (factored normal form or FNF), such as closure under factoring, unfactoring, unification, and generalization. Unification, in particular, has the intuitive appeal of preserving as much as possible the particular factoring of the disjunctive feature structures to be unified. We also show that unification in the FNF class can be extremely efficient in practical applications.

1. INTRODUCTION

It has become common to make a distinction between a language for the description of feature structures and feature structures themselves, which are seen as directed acyclic graphs (*dags*) or automata (see, for instance, Kasper and Rounds, 1986). To avoid confusion, the terms of the representation language are often referred to as *feature descriptions*. Disjunction is a representation tool in the representation language, intended to describe *sets* of feature structures. In this framework, there are no disjunctive feature *structures*, but only disjunctive feature *descriptions*.

This framework has enabled researchers to explore the computational complexity of unification. However, it has some drawbacks. First, properties have to be stated (and proofs carried out) at the syntactic level. This implies using a complicated calculus based on formula equivalence rules, rather than using graph-theoretical properties. In addition, unification is not well-defined with respects to disjunction. There is reference in the literature to the "unification of disjunctive feature descriptions", but, formally, we should instead speak of the unification of the *sets* of feature structures the descriptions represent.

For example, unifying the sets of feature structures represented by the disjunctive feature descriptions in Fig. 1 yields a set of four (non-disjunctive) feature structures, which can be described by several equally legitimate formulae: A factored, B factored, disjunctive normal form (DNF), etc. Depending on the algorithm that is used, the description of the result will be one or the other of these formulae. Some algorithms require expansion to DNF and will therefore produce a DNF representation as a result, but other algorithms may produce different representations.

There is an important body of research concerned with the development of algorithms that avoid the expensive expansion to DNF (e.g., Kasper, 1987). These algorithms typically produce descriptions of the unification, in which some of the disjunctions in the original descriptions are retained. However, these descriptions are produced as a computational *side-effect* (potentially different depending on the algorithm) rather than as a result of the application of a formal definition.

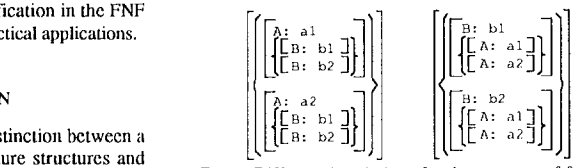

Fig. 1. Different descriptions for the same set of feature structures

In this paper, we first consider disjunctive feature structures as objects in themselves, defined in terms of *directed acyclic hypergraphs*. This enables us to build a mathematical framework based on graph theory in order to study the properties of disjunctive feature structures and specify operations (such as unification) in algebraic rather that syntactic terms. It also enables the specification of algorithms in terms of graph manipulations, and suggests a data structure for implementation.

We then illustrate the expressive power of this framework by defining a class of disjunctive feature structures with interesting properties (*factored normal form* or *FNF*), such as closure under factoring, unfactoring, unification, and generalization. These operations (and the relation of subsumption) are defined in terms of operations on (or relations among) hypergraphs. Unification, in particular, has the intuitive appeal to preserve as much as possible the particular factoring of the disjunctive feature structures to be unified. We also show that unification in the FNF class can be extremely efficient in practical applications.

For lack of space, proofs will be omitted or only suggested.

2. BASIC FRAMEWORK

2.1 Disjunctive feature structures as hypergraphs

(Disjunctive) feature structures will be defined as *directed acyclic hypergraphs*. In a hypergraph (see Berge, 1970), arcs (*hyperarcs*) connect sets of nodes instead of pairs of nodes, as in usual graphs. We will consider hyperarcs as directed from their first node to all other nodes. More precisely, each hyperarc will be an ordered pair consisting of an *input node* n_{i_0}, and a (non-empty) set of *output nodes* $n_{i_1}, ..., n_{i_k}$. We will say that $(n_{i_0}, \{n_{i_1}, ..., n_{i_k}\})$ is a *k-arc* from n_{i_0} to $n_{i_1}, ..., n_{i_k}$, that n_{i_0} is an *immediate predecessor* of $n_{i_1}, ..., n_{i_k}$, and that $n_{i_1}, ..., n_{i_k}$ are *immediate successors* of n_{i_0}..

A *path*[1] in a hypergraph is a sequence of nodes $n_{i_1}, ..., n_{i_p}$ such that for $j = 1, ..., p - 1$, n_{i_j} is an immediate predecessor of $n_{i_{j+1}}$. If there exists a path from a node n_i to a node n_j, we will write $n_i \Rightarrow n_j$. A hypergraph is acyclic if there is no node such that $n_i \Rightarrow n_i$. A hypergraph has a *root* n_0 if for each node $n_i \neq n_0$, $n_0 \Rightarrow n_i$. The *leaves* of a hypergraph are those nodes with no successor. A path terminating with a leaf is a *maximal path*. Nodes with more than one immediate predecessor are called *merging nodes*.

Definition 2.1 Let L be a set of labels and A be a set of atomic values. A *(disjunctive) feature structure* on (L, A) is a quadruple $F = (D, n_0, \lambda, \alpha)$, respecting the consistency conditions 2.1 below, where D is a finite directed acyclic hypergraph with a root n_0, λ is a partial function from the 1-arcs of D into L, and α is a partial function from the leaves of D into A.

Feature structures which have isomorphic hypergraphs, whose corresponding leaves have the same value, and whose corresponding feature-arcs have the same labels, are isomorphic. We will consider such feature structures to be *equal* up to isomorphism.

Definition 2.2 Labeled 1-arcs are called *feature-arcs*. Non-labeled hyperarcs are called OR-*arcs*.

Note that OR-arcs are usually *k*-arcs with $k > 1$, but (non-labeled) 1-arcs can be OR-arcs as a special case. We will use a graphic representation for disjunctive feature structures in which OR-arcs are represented as *k* lines connected together[2] (see Fig. 2).

Definition 2.3 The *extended label* of a given path is the concatenation of all labels along that path. We will use the notation $l_1{:}l_2{:} \ldots l_n$ to represent extended labels. A *maximal extended label* from a node is an extended label for a maximal path from that node.

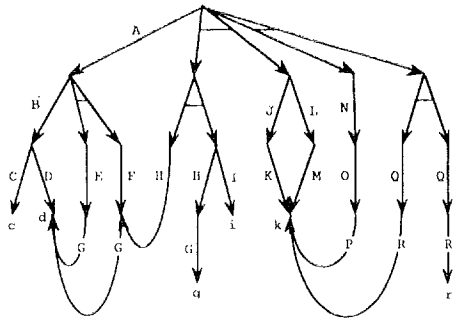

Fig. 2. Graphic representation

Conditions 2.1 Disjunctive feature structures must verify the following consistency conditions:

(C_1) No output node of an OR-arc is a leaf;
(C_2) Output nodes of OR-arcs are not merging nodes;
(C_3) All feature-arcs from the same node have different labels;
(C_4) No maximal extended label from a given node is a prefix of a non-maximal extended label obtained by following a different hyperarc from the same node.

C_1 and C_2 constrain OR-arcs to represent only disjunctions. C_3 and C_4 are extensions of the determinism that is usually imposed on dags (no outgoing arcs with the same label from any given node).

Definition 2.4 A *dag feature structure* is a feature structure with no OR-arc.

Definition 2.5 A *projection* of a feature structure x is a hypergraph obtained by removing all but one output node of all OR-arcs of x.

Therefore, a projection has only 1-arcs.

Definition 2.6 A dag feature structure y is a *dag-projection* of a feature structure x if there exist some projection y' of x and a function h mapping nodes of y' into nodes of y such that:

(1) the root of y' is mapped to the root of y;
(2) if $(n_{i_0}, \{n_{i_1}\})$ is a feature-arc of y', then $(h(n_{i_0}), \{h(n_{i_1})\})$ is a feature-arc of y with the same label;
(3) if $(n_{i_0}, \{n_{i_1}\})$ is a 1-OR-arc of y', then $h(n_{i_0}) = h(n_{i_1})$;
(4) the value associated with a node n_i in y' is the same as the value associated with $h(n_i)$ in y, or both have no value;
(5) each feature arc in y is the image of at least one feature arc in y'.

In other terms, a dag-projection is obtained from a projection by merging the input and output nodes of each 1-OR-arc, and merging paths with common prefixes to ensure determinism.

Definition 2.7 A *sub-feature structure* rooted at a node n_i is a quadruple composed of a sub-hypergraph rooted at that node, the root n_i, together with the

[1] We use this term in the sense usual in graph theory. It should not be confused with the term *path* used in many feature structure studies, which is a string of labels, and for which we will introduce the term *extended label* later in the paper.

[2] In some work involving AND/OR graphs, this convention is used for AND-arcs. This should not create further confusion.

restrictions of the label and value functions to this sub-hypergraph. The AND-*part* of a node is the sub-feature structure rooted at that node, starting with only the feature-arcs from that node. The OR-*parts* of a node are the different sub-feature structures rooted at that node, starting with each of the OR-arcs. The *disjuncts* of an OR-arc are the sub-feature structures rooted at each of the output nodes of that OR-arc. If a node has only one OR-arc, we will call its disjuncts the disjuncts of the node.

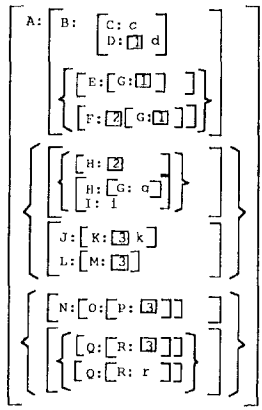

Fig. 3. Description of the feature structure in Fig. 2.

2.2 Representation language

Definition 2.8 The *representation language* for (disjunctive) feature structures described above is defined by the following grammar:

$$
\begin{aligned}
F &\rightarrow [T, \ldots, T] \\
T &\rightarrow l : I\ V \\
T &\rightarrow \{F, \ldots, F\} \\
I &\rightarrow i \mid \varepsilon \\
V &\rightarrow F \mid a \mid \varepsilon
\end{aligned}
$$

where F is the axiom, ε is the empty string, l belongs to the set of labels L, a belongs to the set of atomic values A, and i belongs to a set I of *identifiers* (we use the symbols □, ②, etc.), disjoint from L. A formula Φ of that language is called a *(disjunctive) feature description.*

The mapping between feature structures and feature descriptions is straightforward (Fig. 3). Translating between feature descriptions and feature structures and checking that a description is valid (that is, corresponds to a valid feature structure) is computationally trivial, and does not rely on the (potentially expensive) application of equivalence rules as in Kasper and Rounds (1986).

3. A TYPOLOGY OF NORMAL FORMS

In this section, we will first define the disjunctive normal form (DNF) in terms of hypergraphs. We will then define a family of increasingly restricted normal forms, the most restricted of which is the DNF. One of them, the *factored normal form* (FNF) enables a clear definition of the "format" of a feature structure. It also imposes a strict hierarchical view of the data, and is exactly the class of feature structures that are reachable from the DNF through sequences of factoring operations. We believe that the FNF class is of great linguistic interest, since it is clear that disjunction is often used to reflect hierarchical organization, factoring, etc., and thus is more than just a space-saving device. In the sections that follow, factoring operations in the FNF class will be defined formally, along with appropriate extentions to the notions of subsumption and unification.

3.1 Disjunctive Normal Form

Definition 3.1 A (disjunctive) feature structure is said to be in *disjunctive normal form* (DNF) if:

(1) the root has only one OR-part, and no AND-part;
(2) each disjunct is a dag feature structure;
(3) all the disjuncts are disjoint and different (non-isomorphic).

Note that the disjunctive normal form is defined for feature structures themselves, not for their descriptions.

Definition 3.2 The *disjunctive normal form* of a given feature structure x, noted DNF(x), is a DNF feature structure, in which the set of disjuncts D_i is equal to the set of dag-projections of x.

Definition 3.3 Two feature structures x and y are *DNF-equivalent* if DNF(x) = DNF(y). We will note $x \equiv_{dnf} y$.

3.2 Typology of normal forms

We can define several interesting restrictions on feature structures, which in turn define a typology of increasingly restricted normal forms.

Condition 3.1 Dag-projections obtained by different selections of output nodes of OR-arcs are different.

Condition 3.2 Each node has at most one OR-part.

Condition 3.3 The AND-part of each node is a dag.

Definition 3.4 When combined, the three conditions above define several normal forms:

(1) 3.1: *non-redundant normal form* (NRNF);
(2) 3.1 and 3.2: *hierarchical normal form* (HNF);
(3) 3.1 and 3.3: AND-*normal form* (ANF);
(4) 3.1, 3.2 and 3.3: *layered normal form* (LNF).

Definition 3.5 In an ANF feature structure x, the AND-part of a node n_i is a *maximal* AND-*part* of x if n_i is the output node of no feature arc.

Definition 3.6 The *layers* of a LNF feature structure are defined recursively as follows:

(1) Layer 0 is the AND-part of the root;

(2) Layer $n+1$ is set of (maximal) AND-parts of all the output nodes of OR-arcs originating in layer n.

Let us now turn back to formats.

Definition 3.7 The *format* of a dag feature structure is the set of maximal extended labels starting at its root. The *format* of a layer is the union of formats of all the maximal AND-parts in that layer.

Definition 3.8 A LNF feature structure is said to be in *factored normal form* (FNF) if the following properties hold:

(1) the formats of all layers are disjoint;
(2) paths originating in two distinct maximal AND-parts of a layer n can merge only in a node belonging to an AND-part in a layer n' such that $n' < n$.

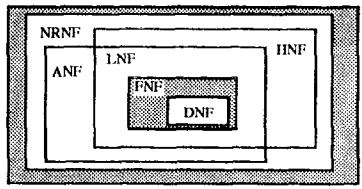

Fig. 3. A typology of normal forms.

Fig. 3 shows the typology of normal forms. Note that the DNF is obviously in FNF.

In the rest of the paper, we will study only the properties of FNF, in which formats are homogeneous.

Definition 3.9 The *format* of a FNF feature structure, noted $f(x)$, is the sequence $<s_0, ..., s_n>$ of the formats of each of its layers, in increasing order starting with the root.

Definition 3.10 We will call sets of extended labels *dag-formats*, and sequences $<s_0, ..., s_n>$ of dag-formats with all s_i disjoint, *fs-formats*.

Proposition 3.2 If two FNF feature structures have the same DNF and the same format, they are equal.

4. RESTRUCTURING OPERATORS

4.1 *Factor* and *unfactor*

Let us give first a few auxiliary definitions.

Definition 4.1 Let x be a dag feature structure, and s a dag-format. The *spanning* of x according to s, noted $span_s(x)$, is the greatest sub-dag of x such that of all the paths in $span_s(x)$ have their extended labels in s.

Note that $f(span_s(x)) \subseteq s$.

Definition 4.2 A dag feature structure F is a *common factor* of a feature structure x if the AND-part of all the disjuncts at the top level of x contain F. A dag format s is said to *span a common factor* of x if the spanning of the AND-part of all the disjuncts at the top level of x according to s is a common factor.

Let us now define the factoring and unfactoring operations. Informally, the *factor* operator extracts a factor common to all the top-level disjuncts, and raises it to the root level.

Definition 4.3 Let x be a FNF feature structure such that $f(x) = <s_0, s_1, s_2, ..., s_n>$ and s a dag-format. If s spans a common factor F, the *factoring* of x according to s, noted $\phi_s(x)$, is the FNF feature structure DNF-equivalent to x with format $<s_0 \cup s', s_1\text{-}s', s_2, ..., s_n>$ where $s' = f(F)$.

Definition 4.4 Let x be a FNF feature structure with an AND-part A, such that $f(x) = <s_0, s_1, s_2, ..., s_n>$, and s be a dag-format. If $F = span_s(x)$, the *unfactoring* of x according to s, noted $\bar{\phi}_s(x)$, is the FNF feature structure that is DNF-equivalent to x with the format $<s_0\text{-}s', s_1 \cup s', s_2, ..., s_n>$, where $s' = f(F)$.

Example. See Fig. 4

Proposition 4.1 $\phi_s(\bar{\phi}_s(x)) = \bar{\phi}_s(\phi_s(x)) = x$

Proposition 4.2

(1) $\phi_s(\phi_{s'}(x)) = \phi_{s'}(\phi_s(x)) = \phi_{s \cup s'}(x)$

(2) $\bar{\phi}_s(\bar{\phi}_{s'}(x)) = \bar{\phi}_{s'}(\bar{\phi}_s(x)) = \bar{\phi}_{s \cup s'}(x)$

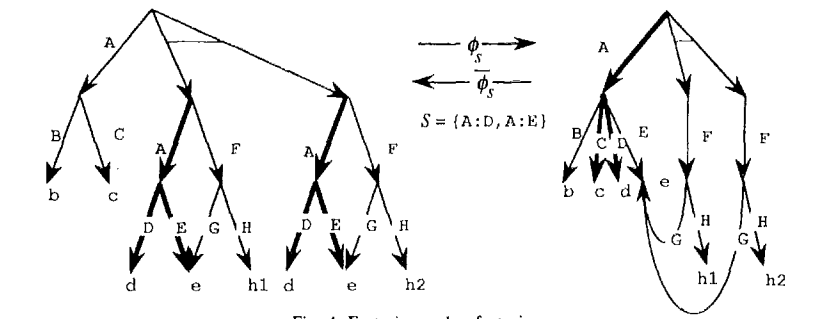

Fig. 4. Factoring and unfactoring

4.2 *Group* and *ungroup*

The *factor* operator requires that there is a common factor. In many cases there is no common factor; however, it is possible to define a *group* operator that first splits feature structures into groups of disjuncts that have common factors with respect to a given format, and then factors them.

Definition 4.5 Let x be a FNF feature structure such that $f(x) = <\emptyset, s_1, s_2, ..., s_n>, A_1, ..., A_n$ be the AND-parts of the top-level disjuncts of x, and s be a dag-format. The *grouping* of x according to s, noted $\gamma_s(x)$, is the FNF feature structure DNF-equivalent to x with format $<\emptyset, s', s_1\text{-}s', s_2, ..., s_n>$ where $s' = \bigcup_i f(span_s(A_i))$.

Definition 4.6 Let x be a FNF feature structure such that $f(x) = <\emptyset, s_1, s_2, ..., s_n>, A_1, ..., A_n$ be the AND-parts of the top-level disjuncts of x, s be a dag-format, and $s' = \bigcup_i f(span_s(A_i))$. We will note $\bar{\gamma}_s(x)$ the *ungrouping* of x according to s :

(1) if $s'=s_1$ then $\bar{\gamma}_s(x)$ is the FNF feature structure DNF-equivalent to x with format $<\emptyset, s_1 \cup s_2, s_3, ..., s_n>$;

(2) if $s' \neq s_1$ then $\bar{\gamma}_s(x)$ is the FNF feature structure DNF-equivalent to x with format $<\emptyset, s_1\text{-}s', s' \cup s_2, s_3, ..., s_n>$.

Example. See Fig. 5.

Proposition 4.3 $\gamma_s(\bar{\gamma}_s(x)) = \bar{\gamma}_s(\gamma_s(x)) = x$

Proposition 4.4 The class of FNF feature structures is closed under factoring, unfactoring, grouping and ungrouping.

4.3 *Format* operator

Definition 4.7 Let S be a fs-format $<s_0, s_1, ..., s_n>$. The *formatting* of a DNF feature structure x according to S, noted $v_S(x)$, is the result of the following sequence of operations:

$$v_S(x) = \phi_{s_0}(\gamma_{s_1 \cup s_0}(\gamma_{s_2 \cup s_1 \cup s_0}(....(\gamma_{s_n \cup ... \cup s_0}(x)))))$$

It is clear that $v_S(x)$ is in FNF, and is DNF-equivalent to x.

Proposition 4.5 Any FNF feature structure x can be reached from its DNF though a sequence of grouping and factoring operations. More precisely, if $x' = \text{DNF}(x)$ then $x = v_{f(x)}(x')$.

Definition 4.8 Let S be a fs-format $<s_0, s_1, ..., s_n>$. The *unformatting* of a FNF feature structure x according to S, noted $\bar{v}_S(x)$, is the result of the following sequence of operations:

$$\bar{v}_S(x) = \bar{\gamma}_{s_n \cup ... \cup s_0}(...(\bar{\gamma}_{s_2 \cup s_1 \cup s_0}(\bar{\gamma}_{s_1 \cup s_0}(\bar{\phi}_{s_0}(x)))))$$

Proposition 4.6 Any FNF feature structure x can be transformed into its DNF though a sequence of unfactoring and ungrouping operations. More precisely, $\bar{v}_{f(x)}(x) = \text{DNF}(x)$.

Proposition 4.7 $v_S(\bar{v}_S(x)) = \bar{v}_S(v_S(x)) = x$

5. SUBSUMPTION, UNIFICATION AND GENERALIZATION

As mentioned in the introduction, the format of the result of unification is not defined in the classical approach. Our goal will be to define unification on FNF disjunctive feature structures in such a way that the format of the result is unique and predictable. Intuitively, when feature descriptions have compatible formats (as in Fig. 6), it seems that unification should preserve it. On the other hand, when two feature descriptions have completely incompatible formats (as in Fig. 1), the resulting format should be in DNF. When formats are only partially compatible, a limited amount of unfactoring should be performed, and the compatible part should be preserved in the result. These considerations lead us to define compatibility of formats, and to extend the notions of subsumption, unification, and generalization to feature structure *formats*. We then define unification and generalization on disjunctive feature structures in such a way that important properties

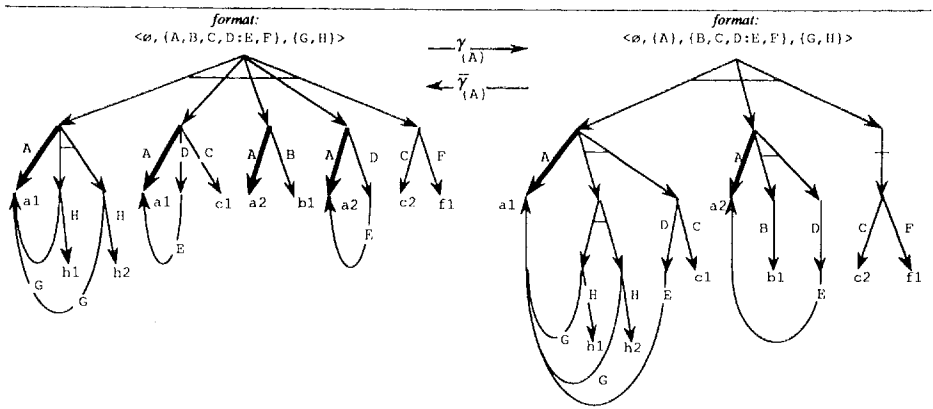

Fig. 5. Grouping and ungrouping

hold. In particular, reduction to DNF, factoring, and grouping are homomorphisms with respect to unification (that is, $DNF(x \sqcup y) = DNF(x) \sqcup DNF(y)$, $\gamma_s(x \sqcup y) = \gamma_s(x) \sqcup \gamma_s(y)$, etc.).

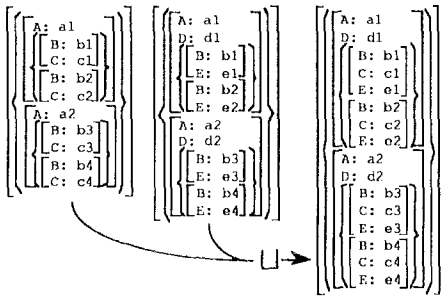

Fig. 6. Compatible formats

In what follows, we will call the classical subsumption, unification, generalization of dag feature structures *dag-subsumption*, *dag-unification* and *dag-generalization* (noted $\leq_{dag}$, $\sqcup_{dag}$, $\sqcap_{dag}$, repectively). The classical subsumption, unification, generalization of DNF feature structures will be called *dnf-subsumption*, *dnf-unification* and *dnf-generalization* (noted $\leq_{dnf}$, $\sqcup_{dnf}$, $\sqcap_{dnf}$, repectively).

5.1 Subsumption, unification, generalization of formats

Definition 5.1 Let S_1 be a fs-format $<s_{1_0}, s_{1_1},..., s_{1_n}>$ and S_2 be a fs-format $<s_{2_0}, s_{2_1},..., s_{2_p}>$. We will say that S_1 *subsumes* S_2 if each p in s_{1_i} belongs to some s_{2_j} with $i \leq j$, for all i in $\{1, n\}$. We will note $S_1 \leq_{fmt} S_2$

Definition 5.2 Let S_1 and S_2 be two fs-formats. The *unification* of S_1 and S_2, noted $S_1 \sqcup_{fmt} S_2$. is the greatest lower bound of S_1 and S_2 according to the format subsumption relation. The *generalization* of S_1 and S_2, noted $S_1 \sqcap_{fmt} S_2$, is the least upper bound of S_1 and S_2 according to the format subsumption relation.

It is easy to prove that these bounds exist. They can be built recursively. For example, let $S_1 = <s_{1_0}, s_{1_1},..., s_{1_n}>$ and $S_2 = <s_{2_0}, s_{2_1},..., s_{2_n}>$ (for the sake of simplicity, we will consider the shorter of S_1 and S_2 to be padded on the right with an appropriate number of $\emptyset$'s in order to ensure the same length). $S = S_1 \sqcup_{fmt} S_2 = <s_0, s_1,..., s_n>$ can be constructed recursively:

(1) $s_n = s_{1_n} \cup s_{2_n}$

(2) $s_i = (s_{1_i} \cup s_{2_i}) - \bigcup_{j=i+1}^{n} s_j$ for all i, $0 \leq i < n$.

Definition 5.3 Let S_1 be a fs-format $<s_{1_0}, s_{1_1},..., s_{1_n}>$ and S_2 a fs-format $<s_{2_0}, s_{2_1},..., s_{2_p}>$. We will say that S_2 is a *sub-format* of S_1 if s_{1_0} is included in s_{2_0} for all i in $\{1, n\}$. We will say that S_1 and S_2 are *compatible* if both S_1 and S_2 are sub-formats of the same format.

5.2 Subsumption, unification, generalization of disjunctive feature structures

Definition 5.4 We will say that a FNF feature structure x *subsumes* a FNF feature structure y, and note $x \leq y$, if

(1) $x \leq_{dnf} y$

(2) $f(x) \leq_{fmt} f(y)$

Definition 5.5 Let x and y be two FNF feature structures. The *unification* of x and y, noted $x \sqcup y$, is the greatest lower bound of x and y according to the subsumption relation. The *generalization* of S_1 and S_2, noted $x \sqcap y$, is the least upper bound of x and y according to the format subsumption relation.

The following proposition states that $x \sqcup y$ is dnf-equivalent to the dnf-unification of the DNFs of x and y, and the format of $x \sqcup y$ is the unification of the formats of x and y:

Proposition 5.1

(1) $DNF(x \sqcup y) = DNF(x) \sqcup_{dnf} DNF(y)$

(2) $f(x \sqcup y) = f(x) \sqcup_{fmt} f(x)$

As a result, the unification of x and y can be computed by completely unformatting both x and y, unifying them, and formatting the result according to the unification of their formats:

Proposition 5.2

$$x \sqcup y = \nu_{f(x) \sqcup_{fmt} f(y)} (\bar{\nu}_{f(x)}(x) \sqcup_{dnf} \bar{\nu}_{f(y)}(y))$$

(Dual proposition holds for generalization.)

Proposition 5.3 The class of FNF feature structures is closed under factoring, unfactoring, unification, and generalization.

This follows directly from the definitions.

Proposition 5.4

(1) $\gamma_s(x \sqcup y) = \gamma_s(x) \sqcup \gamma_s(y)$

(2) $\bar{\gamma}_s(x \sqcup y) = \bar{\gamma}_s(x) \sqcup \bar{\gamma}_s(y)$

(3) $\phi_s(x \sqcup y) = \phi_s(x) \sqcup \phi_s(y)$

(4) $\bar{\phi}_s(x \sqcup y) = \bar{\phi}_s(x) \sqcup \bar{\phi}_s(y)$

(Dual propositions hold for generalization.)

5.3 Algorithm

Proposition 5.2 does not imply that complete unfactoring and re-factoring is the most efficient computation of unification and generalization. Because of the properties given in proposition 5.4, unification can be carried out layer by layer, and only partial unfactoring is needed (algorithm 5.1). In the extreme case, when the formats of x and y are compatible, no unfactoring is needed, and the procedure *match-formats* does nothing.

Algorithm 5.1 *Unification of FNF feature structures*

```
function unify(x, y: feature-structure): feature-structure
    match-formats(x, y)
    // Unify AND-parts
    z.AND ← dag-unify(x.AND, y.AND)
    if z.AND = failure then return failure
    // Unify OR-parts
    z.OR ← unify-disjuncts(x.OR, y.OR)
    if z.OR = failure then return failure else return z

function unify-disjuncts(x, y: feature-structure):
                                        feature-structure
    //assume x.AND and y.AND are empty
    match-formats(x, y)
    k ← 0
    for each x.DISJ_i
        for each y.DISJ_j
            t ← dag-unify(x.DISJ_i.AND, y.DISJ_j.AND)
            if t ≠ failure then
                u ← unify-disjuncts
                        (x.DISJ_i.OR, y.DISJ_j.OR)
                if u ≠ failure then
                    k ← k+1
                    z.DISJ_k.AND ← t
                    z.DISJ_k.OR ← u
    if k = 0 then return failure elsereturn z
```

We will consider the complexity of this algorithm in terms of the number of dag-unifications, which is the only costly operation ($O(n \log(n))$, where n is the total number of symbols in the two dag feature structures--see Aït-Kaci, 1984). We will first consider the case where the formats are compatible. One dag-unification is performed in the *unify* function, but the bulk of the dag-unifications are performed in the *unify-disjuncts* function. There are two nested loops, and the function is applied recursively through all the layers. Therefore, in the worst case, the algorithm requires $O(d^2)$ dag-unifications, whre d is the total number of disjuncts.

When the formats are not compatible, some unfactoring and ungrouping has to be performed by the *match-formats* function in order to force the formats to match. The number of operations can be limited if the two formats are partially compatible, due to the properties of FNF. Complete unformatting will be necessary only in cases where the two formats are completely incompatible.

For example, if $f(x)$ = <{A}, {B,C}, {D,E}, {F}, {G}, {H}>, and $f(y)$ = <{I}, {B,J}, {D,F}, {E,K}, {G}, {L}>, the resulting format is <{A,I}, {B,C,J}, {D}, {E,F,K}, {G}, {H,L}>. The two first layers can be computed without unfactoring. Unfactoring is required for disjuncts at the next level, yielding the formats <{D}, {E,F}, {G}, {H}> and <{D}, {E,F,K}, {G}, {L}>, respectively. When this is accomplished the formats match, and the algorithm can resume with no more unfactoring.

It is clear that, in the worst case, when the algorithm requires the complete unformatting of the two feature structures, the total number of dag-unifications grows exponentially with the number of disjuncts. However, in most pratical cases, the algorithm is likely to perform better. We saw, in particular, that when the two feature structures have completely compatible formats, the complexity is only quadratic. There is obviously a range of possible behaviors between these two extremes.

It seems to us that in practical applications, disjunction is not random, but, instead, reflects some systematic linguistic properties. A high degree of compatibility among formats is therefore expected. It should also be noted that the algorithm can easily be modified so that only one feature structure is unfactored and re-formatted into a format that is compatible with the format of the other. This is especially useful in the common situation in which a small feature structure, containing a small number of disjuncts (e.g. a constituent at a given stage of parsing) is matched against a very large feature structure (e.g. a grammar). In this case, the time required for unformatting and reformatting the "small" feature structure is negligible, and the overall number of dag-unifications grows linearly with the number of disjuncts in the "large" feature structure.

6. CONCLUSION

In this paper, we present a new mathematical framework in which disjunctive feature structures are defined as directed acyclic hypergraphs. Disjunction is defined in the feature structure domain, and not at the syntactic level in feature descriptions. This enables us to study properties and specify operations (such as unification) and relations (such as subsumption) in terms of algebraic operations on (or relations among) hypergraphs rather than in syntactic terms. We illustrate the expressive power of this framework by defining a class of disjunctive feature structures with interesting properties (factored normal form, or FNF), such as closure under factoring, unfactoring, unification, and generalization. Unification, in particular, has the intuitive appeal of preserving as much as possible the particular factoring of the disjunctive feature structures to be unified. We also show that unification in the FNF class can be extremely efficient in practical applications.

Acknowledgments -- The present research has been partially funded by the GRECO-PRC Communication Homme-Machine of the French Ministery of Research and Technology and U.S.-French NSF/CNRS grant INT-9016554 for collaborative research. The author would like to thank Nancy Ide for her valuable comments and help in the preparation of this paper.

REFERENCES

AÏT-KACI, H. (1984) *A New Model of Computation Based on a Calculus of Type Subsumption*. Ph. D. Thesis, Univ. of Pennsylvania.

BERGE, C. (1970). *Graphes et Hypergraphes*. Paris: Dunod. [translation: *Graphs and Hypergraphs*, Amsterdam : North-Holland, 1973]

KASPER, R. T. (1987). A unification method for disjunctive feature descriptions. *Proc. 25th Annual Meeting of the Association for Computational Linguistics*. Stanford, California, 235-242.

KASPER, R., ROUNDS, W. C. (1986). A logical semantics for feature structures. *Proc. 24th Annual Meeting of the Association for Computational Linguistics*. New York, 257-266.