Off-line Parsability and the Well-foundedness of Subsumption*

Shuly Wintner and Nissim Francez

Department Computer Science
Technion, Israel Institute of Technology
32000 Haifa, Israel

Abstract

Typed feature structures are used extensively for the specification of linguistic information in many formalisms. The subsumption relation orders TFSs by their information content. We prove that subsumption of acyclic TFSs is well-founded, whereas in the presence of cycles general TFS subsumption is not well-founded. We show an application of this result for parsing, where the well-foundedness of subsumption is used to guarantee termination for grammars that are off-line parsable. We define a new version of off-line parsability that is less strict than the existing one; thus termination is guaranteed for parsing with a larger set of grammars.

Keywords: Computational Linguistics, Parsing, Feature structures, Unification
This paper has not been submitted elsewhere in identical or similar form

1 Introduction

Feature structures serve as a means for the specification of linguistic information in current linguistic formalisms such as LFG (Kaplan and Bresnan, 1982), HPSG (Pollard and Sag, 1994) or (some versions of) Categorial Grammar (Haddock, Klein, and Morill, 1987). This paper focuses on typed feature structures (TFSs) that are a generalization of their untyped counterparts. TFSs are related by *subsumption* (see (Carpenter, 1992b)) according to their

Journal of Logic, Language and Information, accepted for publication

information content. We show that the subsumption relation is well-founded for acyclic TFSs, but not for cyclic ones. We use this result to prove that parsing is terminating for grammars that are *off-line parsable*: this proposition is cited, but not proved, in (Shieber, 1992). We also suggest a less strict definition for off-line parsability that guarantees termination in the case of acyclic TFSs.

This work has originated out of our interest in the theoretical aspects of parsing with grammars that are based on TFSs (see (Wintner and Francez, 1995b)). While the results presented here are basically theoretical, we have implemented a system for efficient processing of such grammars, based on abstract machine techniques; this work is presented in (Wintner and Francez, 1995a). The rest of this paper is organized as follows: section 2 outlines the theory of TFSs of (Carpenter, 1992b). In section 3 we discuss the well-foundedness of TFS subsumption. We sketch a theory of parsing in section 4 and discuss off-line parsability of TFS-based grammars in section 5.

2 Theory of feature structures

This section summarizes some preliminary notions along the lines of (Carpenter, 1992b). While we use the terminology of *typed* feature structures, all the results are valid for untyped structures, that are a special case of TFSs. (Carpenter, 1992b) defines *well-typed* and *totally well-typed* feature structures that are subsets of the set of TFSs; for generality, we assume nothing about the well-typedness of TFSs below. However, the larger context of our work is done in a setup where features are assigned to types through an *appropriateness* specification, and hence we retain the term *typed* feature structures rather than *sorted* ones. For the following discussion we fix non-empty, finite, disjoint sets TYPES and FEATS of types and features, respectively, and an infinite set NODES of nodes, disjoint of TYPES and FEATS, each member of which is assigned a type from TYPES through a fixed typing function θ : NODES $\rightarrow$ TYPES. The set NODES is 'rich' in the sense that for every $t \in$ TYPES, the set $\{q \in \text{NODES} \mid \theta(q) = t\}$ is infinite.

Below, the meta-variable T ranges over subsets of TYPES, t – over types, f – over features and q – over nodes. For a partial function F, '$F(x)\downarrow$' means that F is defined for the value x. Whenever the result of an application of a partial function is used as an operand, it is

meant that the function is defined for its arguments. $I\!N$ denotes the set of natural numbers.

A partial order $\sqsubseteq$ over TYPES is a **type hierarchy** if it is bounded complete, i.e., if every up-bounded subset T of TYPES has a (unique) least upper bound, $\sqcup T$. If $t \sqsubseteq t'$, t is said to be more general than t', which is more specific than t. t' is a subtype of t. $\bot = \sqcup\phi$ is the most general type; $\top = \sqcup$TYPES is the most specific, inconsistent type. All occurrences of $\top$ are identified.

A **feature structure** (over the parameters NODES, TYPES and FEATS) is a directed, connected, labeled graph consisting of a finite, nonempty set of nodes $Q \subseteq$ NODES, a root $\bar{q} \in Q$, and a partial function $\delta : Q \times \text{FEATS} \rightarrow Q$ specifying the arcs, such that every node $q \in Q$ is accessible from $\bar{q}$. A, B (with or without subscripts) range over feature structures and $Q, \bar{q}, \delta$ (with the same subscripts) over their constituents.[1] Let TFSs be the set of all typed feature structures (over the fixed parameters NODES, TYPES and FEATS).

A **path** is a finite sequence of features, and the set PATHS $=$ FEATS* is the collection of paths. ϵ is the empty path. π, α (with or without subscripts) range over paths. The definition of δ is extended to paths in the natural way: $\delta(q, \epsilon) = q$ and $\delta(q, f\pi) = \delta(\delta(q, f), \pi)$. The paths of a feature structure A are $\Pi(A) = \{\pi \mid \pi \in \text{PATHS and } \delta_A(\bar{q}_A, \pi)\downarrow\}$. Note that for every TFS A, $\Pi(A) \neq \phi$ since $\epsilon \in \Pi(A)$ for every A.

A feature structure $A = (Q, \bar{q}, \delta)$ is **cyclic** if there exist a non-empty path $\alpha \in$ PATHS and a node $q \in Q$ such that $\delta(q, \alpha) = q$. It is **acyclic** otherwise. Let ATFSs be the set of all acyclic TFSs (over the fixed parameters). A feature structure $A = (Q, \bar{q}, \delta)$ is **reentrant** iff there exist two different paths $\pi_1, \pi_2 \in \Pi(A)$ such that $\delta(\bar{q}, \pi_1) = \delta(\bar{q}, \pi_2)$.

Definition 2.1 (Subsumption) $A_1 = (Q_1, \bar{q}_1, \delta_1)$ **subsumes** $A_2 = (Q_2, \bar{q}_2, \delta_2)$ *iff there exists a total function* $h : Q_1 \rightarrow Q_2$ *(a* **subsumption morphism***) such that*

- $h(\bar{q}_1) = \bar{q}_2$
- *for every* $q \in Q_1$, $\theta(q) \sqsubseteq \theta(h(q))$
- *for every* $q \in Q_1$ *and for every* f *such that* $\delta_1(q, f)\downarrow$, $h(\delta_1(q, f)) = \delta_2(h(q), f)$

[1] Untyped feature structures can be modeled by TFSs: consider a particular type hierarchy in which the set of types is the set of atoms, plus the types *complex* and $\bot$, $\bot$ subsumes every other type, and the rest of the types are incomparable. All features are appropriate for the type *complex* only, with $\bot$ as their appropriate value. Atomic nodes are labeled by an atom, non-atomic nodes – by *complex* and variables – by $\bot$.

The symbol '$\sqsubseteq$' is overloaded to denote subsumption (in addition to the subtype relation).

The morphism h associates with every node in Q_1 a node in Q_2 with at least as specific a type; moreover, if an arc labeled f connects q with q' in A_1, then such an arc connects $h(q)$ with $h(q')$ in A_2. Two properties follow directly from the definition: If $A \sqsubseteq B$ then every path defined in A is defined in B, and if two paths are reentrant in A they are reentrant in B.

If two feature structures subsume each other then they have exactly the same structure. The only thing that distinguishes between them is the identity of the nodes. This information is usually irrelevant, and thus an isomorphism is defined over TFSs as follows: A and B are **alphabetic variants** (denoted $A \sim B$) iff $A \sqsubseteq B$ and $B \sqsubseteq A$. A **strictly subsumes** B ($A \sqsubset B$) iff $A \sqsubseteq B$ and $A \not\sim B$.

If A strictly subsumes B then one of following cases must hold: either B contains paths that A doesn't; or there is a path in B that ends in a node with a type that is greater than its counterpart in A; or B contains 'more reentrancies': paths that lead to the same node in B lead to different nodes in A.

Lemma 2.2 *If $A \sqsubset B$ (through the subsumption morphism h) then at least one of the following conditions holds:*

1. *There exists a path $\pi \in \Pi(B) \setminus \Pi(A)$*
2. *There exists a node $q \in Q_A$ such that $\theta(q) \sqsubset \theta(h(q))$,*
3. *There exist paths $\pi_1, \pi_2 \in \Pi(A)$ such that $\delta_A(\bar{q}_A, \pi_1) \neq \delta_A(\bar{q}_A, \pi_2)$ but $\delta_B(\bar{q}_B, \pi_1) = \delta_B(\bar{q}_B, \pi_2)$.*

3 Well-foundedness

In this section we discuss the well-foundedness of TFS subsumption. A partial order $\succ$ on a set D is **well-founded** iff there does not exist an infinite decreasing sequence $d_0 \succ d_1 \succ d_2 \succ \ldots$ of elements of D. We prove that subsumption of acyclic TFSs is well-founded, and show an example of general (cyclic) TFSs for which subsumption is not well-founded. While these results are not surprising, and in fact might be deduced from works such as, e.g., (Moshier

and Rounds, 1987) or (Shieber, 1992), they were not, to the best of our knowledge, spelled out explicitly before.

Lemma 3.1 *A TFS A is acyclic iff $\Pi(A)$ is finite.*

Proof: If A is cyclic, there exists a node $q \in Q$ and a non-empty path α that $\delta(q,\alpha) = q$. Since q is accessible, let π be the path from the root to q: $\delta(\bar{q},\pi) = q$. The infinite set of paths $\{\pi\alpha^i \mid i \geq 0\}$ is contained in $\Pi(A)$.
If A is acyclic then for every non-empty path $\alpha \in$ PATHS and every $q \in Q$, $\delta(q,\alpha) \neq q$. Q is finite, and so is FEATS, so the out-degree of every node is finite. Therefore the number of different paths leaving $\bar{q}$ is bounded, and hence $\Pi(A)$ is finite. □

Definition 3.2 (Rank) *Let r : TYPES $\rightarrow I\!N$ be a total function such that $r(t) < r(t')$ if $t \sqsubset t'$. For an acyclic TFS A, let $\Delta(A) = |\Pi(A)| - |Q_A|$ and let $\Theta(A) = \sum_{\pi \in \Pi(A)} r(\theta(\delta(\bar{q},\pi)))$. Let rank : ATFSs $\rightarrow I\!N$ be defined by $rank(A) = |\Pi(A)| + \Theta(A) + \Delta(A)$.*

By lemma 3.1, *rank* is well defined for acyclic TFSs. $\Delta(A)$ can be thought of as the 'number of reentrancies' in A: every node $q \in Q_A$ contributes in-degree$(q) - 1$ to $\Delta(A)$. For every acyclic TFS A, $\Delta(A) \geq 0$ (clearly $\Theta(A) \geq 0$ and $|\Pi(A)| \geq 0$) and hence $rank(A) \geq 0$.

Lemma 3.3 *If $A \sqsubset B$ and both are acyclic then $rank(A) < rank(B)$.*

Proof: Since $A \sqsubseteq B$, $\Pi(A) \subseteq \Pi(B)$. Consider the two possible cases:

- If $\Pi(A) = \Pi(B)$, then
  - $|\Pi(A)| = |\Pi(B)|$;
  - $\Theta(A) \leq \Theta(B)$ by the definitions of Θ and subsumption;
  - $\Delta(A) \leq \Delta(B)$ by the definition of subsumption, since every reentrancy in A is a reentrancy in B;
  - By lemma 2.2, either $\Theta(A) < \Theta(B)$ (if case (2) holds), or $\Delta(A) < \Delta(B)$ (if case (3) holds).

  Hence $rank(A) < rank(B)$.
- If $\Pi(A) \subset \Pi(B)$ then

- $|\Pi(A)| < |\Pi(B)|$;
- $\Theta(A) \leq \Theta(B)$ (as above)
- it might be the case that $|Q_A| < |Q_B|$. But for every node $q \in Q_B$ that is not the image of any node in Q_A, there exists a path π such that $\delta_B(\bar{q}_B, \pi) = q$ and $\pi \notin \Pi(A)$. Hence $|\Pi(A)| - |Q_A| \leq |\Pi(B)| - |Q_B|$.

Hence $rank(A) < rank(B)$. □

Theorem 3.4 *Subsumption of acyclic TFSs is well-founded.*

Proof: For every acyclic TFS A, $rank(A) \in I\!N$. By lemma 3.3, if $A \sqsubset B$ then $rank(A) < rank(B)$. If an infinite decreasing sequence of acyclic TFSs existed, $rank$ would have mapped them to an infinite decreasing sequence in $I\!N$, which is a contradiction. Hence subsumption is well-founded for acyclic TFSs. □

Theorem 3.5 *Subsumption of TFSs is not well-founded.*

Proof: Consider the infinite sequence of TFSs $A_0, A_1, \ldots$ depicted graphically in figure 1. For every $i \geq 0$, $A_i \sqsupset A_{i+1}$: to see that consider the morphism h_i that maps $\bar{q}_{i+1}$ to $\bar{q}_i$ and $\delta_{i+1}(q, f)$ to $\delta_i(h(q), f)$ for every $q \in Q_{i+1}$ (i.e., the first $i+1$ nodes of A_{i+1} are mapped to the first $i+1$ nodes of A_i, and the additional node of A_{i+1} is mapped to the last node of A_i). Clearly, for every $i \geq 0$, h_i is a subsumption morphism. Hence, for every $i \geq 0, A_i \sqsupseteq A_{i+1}$.
To show strictness, assume a subsumption morphism $h' : Q_i \rightarrow Q_{i+1}$. By definition, $h'(\bar{q}_i) = \bar{q}_{i+1}$. By the third requirement of subsumption (h commuting with δ), the first $i+1$ nodes in A_i have to be mapped by h' to the first $i+1$ nodes in A_{i+1}. However, if q is the $i+1$-th node of A_i, then $\delta_i(q)$ leads back to q, while $\delta_{i+1}(h'(q))$ leads to the last node of A_{i+1} (the cyclic node), and hence h' does not commute with δ, a contradiction. Hence, $A_i \sqsupset A_{i+1}$.
Thus, there exists a strictly decreasing infinite sequence of cyclic TFSs and therefore subsumption is not well-founded. □

To conclude this section, note that *specification*, which is the inverse relation to subsumption, is not well-founded even when cyclic feature structures are ruled out. This fact can easily be seen by considering the sequence of feature structures $B_0, B_1, \ldots$, where B_i consists

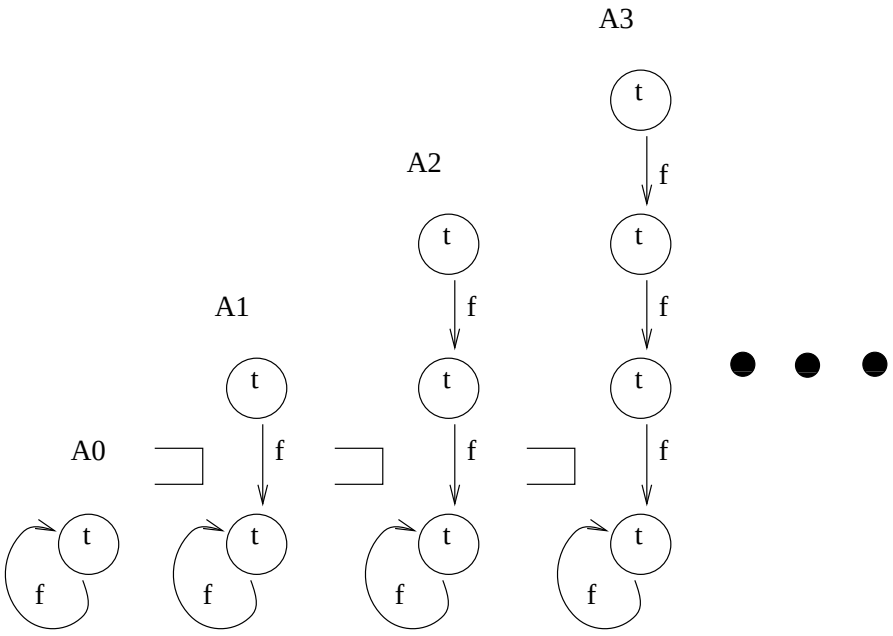

Figure 1: An infinite decreasing sequence of TFSs

of $i+1$ nodes, the first i of which are labeled t and the last – $\perp$, and an f-arc leads from every node to its successor. Clearly, $B_i \sqsubset B_{i+1}$ for every $i > 0$, and the sequence is infinite. This is true whether or not appropriateness constraints are imposed on the feature structures involved.

In the general case, then, given a feature structure A it might be possible to construct, starting from A, both an infinite decreasing sequence of TFSs (by expanding cycles) and an infinite increasing sequence (by adding paths).

4 Parsing

Parsing is the process of determining whether a given string belongs to the language defined by a given grammar, and assigning a structure to the permissible strings. A large variety of parsing algorithms exists for various classes of grammars (for a detailed treatment of the theory of parsing with grammars that are based on feature structures, refer to (Shieber, 1992; Sikkel, 1993; Wintner and Francez, 1995b)). We define below a simple algorithm for grammars that are based on TFSs, but it must be emphasized that the results presented in

this paper are independent of the particular algorithm; they hold for a wide range of different algorithms.

To be able to represent complex linguistic information, such as phrase structure, the notion of feature structures is usually extended. There are two different approaches for representing phrase structure in feature structures: by adding special, designated features to the FSs themselves; or by defining an extended notion of FSs. The first approach is employed by HPSG: special features, such as DTRS (daughters), encode trees in TFSs as lists. This makes it impossible to directly access a particular daughter. (Shieber, 1992) uses a variant of this approach, where a denumerable set of special features, namely $0, 1, \ldots$, are added to encode the order of daughters in a tree. In a typed system such as ours, this method would necessitate the addition of special types as well; in general, no bound can be placed on the number of features and types necessary to state rules (see (Carpenter, 1992b, p. 194)).

We adopt below the other approach: a new notion of *multi-rooted feature structures*, suggested by (Sikkel, 1993) and (Wintner and Francez, 1995b), is being used.

Definition 4.1 (Multi-rooted structures) *A* **multi-rooted feature structure** *(MRS) is a pair $\langle \bar{Q}, G \rangle$ where $G = \langle Q, \delta \rangle$ is a finite, directed, labeled graph consisting of a set $Q \subseteq$ NODES of nodes and a partial function $\delta : Q \times \text{FEATS} \to Q$ specifying the arcs, and $\bar{Q}$ is an ordered (repetition-free) set of distinguished nodes in Q called* **roots**. *G is not necessarily connected, but the union of all the nodes reachable from all the roots in $\bar{Q}$ is required to yield exactly Q. The* **length** *of a MRS is the number of its roots, $|\bar{Q}|$. λ denotes the empty MRS (where $Q = \phi$ since $|\bar{Q}| = 0$). A MRS is* **cyclic** *under the same conditions a TFS is. A MRS is* **reentrant** *if it contains a node that can be reached either from two different roots or through two different paths.*

Meta-variables σ, ρ range over MRSs, and $\delta, Q, \bar{Q}$ over their constituents. If $\sigma = \langle \bar{Q}, G \rangle$ is a MRS and $\bar{q}_i$ is a root in $\bar{Q}$ then $\bar{q}_i$ naturally induces a feature structure $A_i = (Q_i, \bar{q}_i, \delta_i)$, where Q_i is the set of nodes reachable from $\bar{q}_i$ and $\delta_i = \delta|_{Q_i}$. Thus σ can be viewed as an ordered sequence $\langle A_1, \ldots, A_n \rangle$ of (not necessarily disjoint) feature structures. We use the two views of MRSs interchangeably.

Not only can nodes be shared by more than one element of the sequence; paths that start in one root can reach a different root. In particular, cycles can involve more than one root. Still, it is possible to define *sub-structures* of MRSs by considering only the sub-graph that

is accessible from a sub-sequence of the roots.

Definition 4.2 (Sub-structure) *The* **sub-structure** *of $\sigma = \langle A_1, \ldots, A_n \rangle$, induced by the pair $\langle i, j \rangle$ and denoted $\sigma^{i \ldots j}$, is $\langle A_i, \ldots, A_j \rangle$. If $i > j$, $\sigma^{i \ldots j} = \lambda$. If $i = j$, we use σ^i for $\sigma^{i \ldots i}$.*

Definition 4.3 (Subsumption of multi-rooted structures) *A MRS $\sigma = \langle \bar{Q}, G \rangle$* **subsumes** *a MRS $\sigma' = \langle \bar{Q}', G' \rangle$ (denoted by $\sigma \sqsubseteq \sigma'$) if $|\bar{Q}| = |\bar{Q}'|$ and there exists a total function $h : Q \to Q'$ such that:*

- *for every root $\bar{q}_i \in \bar{Q}, h(\bar{q}_i) = \bar{q}'_i$*
- *for every $q \in Q$, $\theta(q) \sqsubseteq \theta(h(q))$*
- *for every $q \in Q$ and $f \in$ FEATS, if $\delta(q, f)\downarrow$ then $h(\delta(q, f)) = \delta'(h(q), f)$*

Many of the properties of TFSs are easily adaptable to MRSs. Let $\Pi(\sigma) = \{(\pi, i) \mid \pi \in$ PATHS, $\bar{q}$ is the i-th root in $\bar{Q}_\sigma$ and $\delta_\sigma(\bar{q}, \pi)\downarrow\}$. Then it is easy to show that if $\sigma \sqsubseteq \rho$ then $\Pi(\sigma) \subseteq \Pi(\rho)$ and every reentrancy in σ is a reentrancy in ρ. Moreover, if $\sigma \sqsubset \rho$ (strictly) then at least one of the three conditions listed in lemma 2.2 holds.

The well-foundedness result of the previous section are easily extended to MRSs as well. Let $\Theta(\sigma) = \sum_{(\pi, \bar{q}) \in \Pi(\sigma)} r(\theta(\delta_\sigma(\bar{q}, \pi)))$ and $\Delta(\sigma) = |\Pi(\sigma)| - |Q_\sigma|$, and the same rank function of TFSs can be used to show the well-foundedness of (acyclic) MRSs. The reverse direction is immediate: in the presence of cycles, duplicate the example of the previous section k times and an infinite decreasing sequence of MRSs of length k is obtained, for any $k > 0$. For a detailed discussion of the properties of MRSs, refer to (Wintner and Francez, 1995b).

Rules and grammars are defined over an additional parameter, a fixed, finite set WORDS of words (in addition to the parameters NODES, FEATS and TYPES). The *lexicon* associates with every word w a feature structure $Cat(w)$, its **category**.[2] The categories are assumed to be disjoint. The input for the parser, therefore, is a sequence of (disjoint) TFSs rather than a string of words.

Definition 4.4 (Pre-terminals) *Let $w = w_1 \ldots w_n \in$ WORDS*. $PT_w(j, k)$ is defined iff $1 \leq j, k \leq n$, in which case it is the MRS $\langle Cat(w_j), Cat(w_{j+1}), \ldots, Cat(w_k) \rangle$. If $j > k$ then $PT_w(j, k) = \lambda$.*

[2] For the sake of simplicity, we assume that lexical entries are not ambiguous. In the case of ambiguity, $Cat(w)$ is a set of TFSs. While the definitions become more cumbersome, all the results still obtain.

If a single word occurs more than once in the input (that is, $w_i = w_j$ for $i \neq j$), its category is copied (with remaned nodes) more than once in PT.

Definition 4.5 (Grammars) *A* **rule** *is an MRS of length greater than or equal to 1 with a designated (first) element, the* **head** *of the rule. The rest of the elements form the rule's* **body** *(which may be empty). A* **grammar** $G = (\mathcal{R}, A_s)$ *is a finite set of rules* $\mathcal{R}$ *and a* **start symbol** A_s *that is a TFS.*

Figure 2 depicts an example grammar (we use AVM notation for this rule; tags such as $\boxed{1}$ denote reentrancy). The type hierarchy on which the grammar is based is omitted here.

Initial symbol:

$$\begin{bmatrix} \textbf{phrase} \\ CAT: \quad \begin{bmatrix} \textbf{s} \end{bmatrix} \end{bmatrix}$$

Lexicon:

John

$$\begin{bmatrix} \textbf{word} \\ CAT: \quad \begin{bmatrix} \textbf{n} \end{bmatrix} \\ AGR: \quad \begin{bmatrix} \textbf{agr} \\ PER: \quad \begin{bmatrix} \textbf{3rd} \end{bmatrix} \\ NUM: \quad \begin{bmatrix} \textbf{sg} \end{bmatrix} \end{bmatrix} \\ SEM: \quad \begin{bmatrix} \textbf{sem} \\ PRED: \quad \begin{bmatrix} \textbf{john} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

her

$$\begin{bmatrix} \textbf{word} \\ CAT: \quad \begin{bmatrix} \textbf{n} \\ CASE: \quad \begin{bmatrix} \textbf{acc} \end{bmatrix} \end{bmatrix} \\ AGR: \quad \begin{bmatrix} \textbf{agr} \\ PER: \quad \begin{bmatrix} \textbf{3rd} \end{bmatrix} \\ NUM: \quad \begin{bmatrix} \textbf{sg} \end{bmatrix} \end{bmatrix} \\ SEM: \quad \begin{bmatrix} \textbf{sem} \\ PRED: \quad \begin{bmatrix} \textbf{she} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

loves

$$\begin{bmatrix} \textbf{word} \\ CAT: \quad \begin{bmatrix} \textbf{v} \end{bmatrix} \\ AGR: \quad \begin{bmatrix} \textbf{agr} \\ PER: \quad \begin{bmatrix} \textbf{3rd} \end{bmatrix} \\ NUM: \quad \begin{bmatrix} \textbf{sg} \end{bmatrix} \end{bmatrix} \\ SEM: \quad \begin{bmatrix} \textbf{sem} \\ PRED: \quad \begin{bmatrix} \textbf{love} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Rules:

$$\begin{bmatrix} \textbf{phrase} \\ CAT: \quad \begin{bmatrix} \textbf{s} \end{bmatrix} \\ AGR: \quad \boxed{1} \\ SEM: \quad \boxed{2} \begin{bmatrix} \textbf{sem} \\ ARG1: \quad \boxed{3} \end{bmatrix} \end{bmatrix} \longrightarrow \begin{bmatrix} \textbf{sign} \\ CAT: \quad \begin{bmatrix} \textbf{n} \\ CASE: \quad \begin{bmatrix} \textbf{nom} \end{bmatrix} \end{bmatrix} \\ AGR: \quad \boxed{1} \\ SEM: \quad \begin{bmatrix} \textbf{sem} \\ PRED: \quad \boxed{3} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \textbf{sign} \\ CAT: \quad \begin{bmatrix} \textbf{v} \end{bmatrix} \\ AGR: \quad \boxed{1} \\ SEM: \quad \boxed{2} \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} \textbf{phrase} \\ CAT: \quad \begin{bmatrix} \textbf{v} \end{bmatrix} \\ AGR: \quad \boxed{1} \\ SEM: \quad \boxed{2} \begin{bmatrix} \textbf{sem} \\ ARG2: \quad \boxed{3} \end{bmatrix} \end{bmatrix} \longrightarrow \begin{bmatrix} \textbf{sign} \\ CAT: \quad \begin{bmatrix} \textbf{v} \end{bmatrix} \\ AGR: \quad \boxed{1} \\ SEM: \quad \boxed{2} \end{bmatrix} \begin{bmatrix} \textbf{sign} \\ CAT: \quad \begin{bmatrix} \textbf{n} \\ CASE: \quad \begin{bmatrix} \textbf{acc} \end{bmatrix} \end{bmatrix} \\ SEM: \quad \begin{bmatrix} \textbf{sem} \\ PRED: \quad \boxed{3} \end{bmatrix} \end{bmatrix} \tag{2}$$

Figure 2: An example grammar

In what follows we define the notion of *derivation* (or *rewriting*) with respect to TFS-based grammars. Informally, this relation (denoted '$\rightsquigarrow$') is defined over MRSs such that $\sigma \rightsquigarrow \rho$ iff

ρ can be obtained from σ by successive application of grammar rules. The reader is referred to, e.g., (Sikkel, 1993; Shieber, Schabes, and Pereira, 1994; Wintner and Francez, 1995b) for a detailed formulation of this concept for a variety of formalisms.

To define derivations we first define *immediate derivation*. Informally, two MRSs A and B are related by immediate derivation if there exists some grammar rule ρ that licenses the derivation. ρ can license a derivation by some MRS R that it subsumes; the head of R must be identified with some element i in A, and the body of R must be identified with a sub-structure of B, starting from i. The parts of A prior to and following i remain intact in B. Note that R might carry reentrancies from A to B: if a path π_2 leaving the i-th element of A is reentrant with some path π_1 leaving the a-th element, and π_2, starting from the head of R, is reentrant with π_3 in some element b in R, then π_1 and π_3 are reentrant in B, starting from the elements in B that correspond to a and b, respectively.

Definition 4.6 *A MRS $A = \langle A_1, \ldots A_k \rangle$* **immediately derives** *a MRS $B = \langle B_1, \ldots B_m \rangle$ (denoted $A \rightarrow B$) iff there exist a rule $\rho \in \mathcal{R}$ of length n and a MRS $R \sqsupseteq \rho$, such that:*

- $m = k + n - 2$
- *R's head is identified with some element i of A: $R^1 = A^i$;*
- *R's body is identified with a sub-structure of B: $R^{2\ldots n} = B^{i\ldots i+n-2}$*
- *The first $i-1$ elements of A and B are identical: $A^{1\ldots i-1} = B^{1\ldots i-1}$;*
- *The last $k-i$ elements of A and B are identical: $A^{i+1\ldots k} = B^{m-(k-i+1)\ldots m}$.*

The reflexive transitive closure of '$\rightarrow$', denoted '$\stackrel{*}{\rightarrow}$', is defined as follows: $A \stackrel{*}{\rightarrow} A''$ if $A = A''$ or if there exists A' such that $A \rightarrow A'$ and $A' \stackrel{*}{\rightarrow} A''$.

Definition 4.7 *A MRS A* **derives** *a MRS B = (denoted $A \rightsquigarrow B$) iff there exist MRSs A', B' such that $A \sqsubseteq A'$, $B \sqsubseteq B'$ and $A' \stackrel{*}{\rightarrow} B'$.*

Immediate derivation is based on the more traditional notion of *substituting* some symbol which constitutes the head of some rule with the body of the rule, leaving the context intact. However, as our rules are based on TFSs, the context of the "symbol" to be substituted might be affected by the substitution. To this end we require identity, and not only unifiability, of

the contexts. MRSs related by derivations should be viewed as being "as specific as needed", i.e., containing all the information that is added by the rule that licenses the derivation. This is also the reason for the weaker conditions on the '$\rightsquigarrow$' relation: it allows an MRS A to derive an MRS B if there is a sequence of immediate derivations that starts with a sepcification of A and ends in a specification of B.

Figure 3 depicts a derivation of the string "John loves her" with respect to the example grammar. The scope of reentrancy tags should be limited to one MRS, but we use the same tags across different MRSs to emphasize the flow of information during derivation.

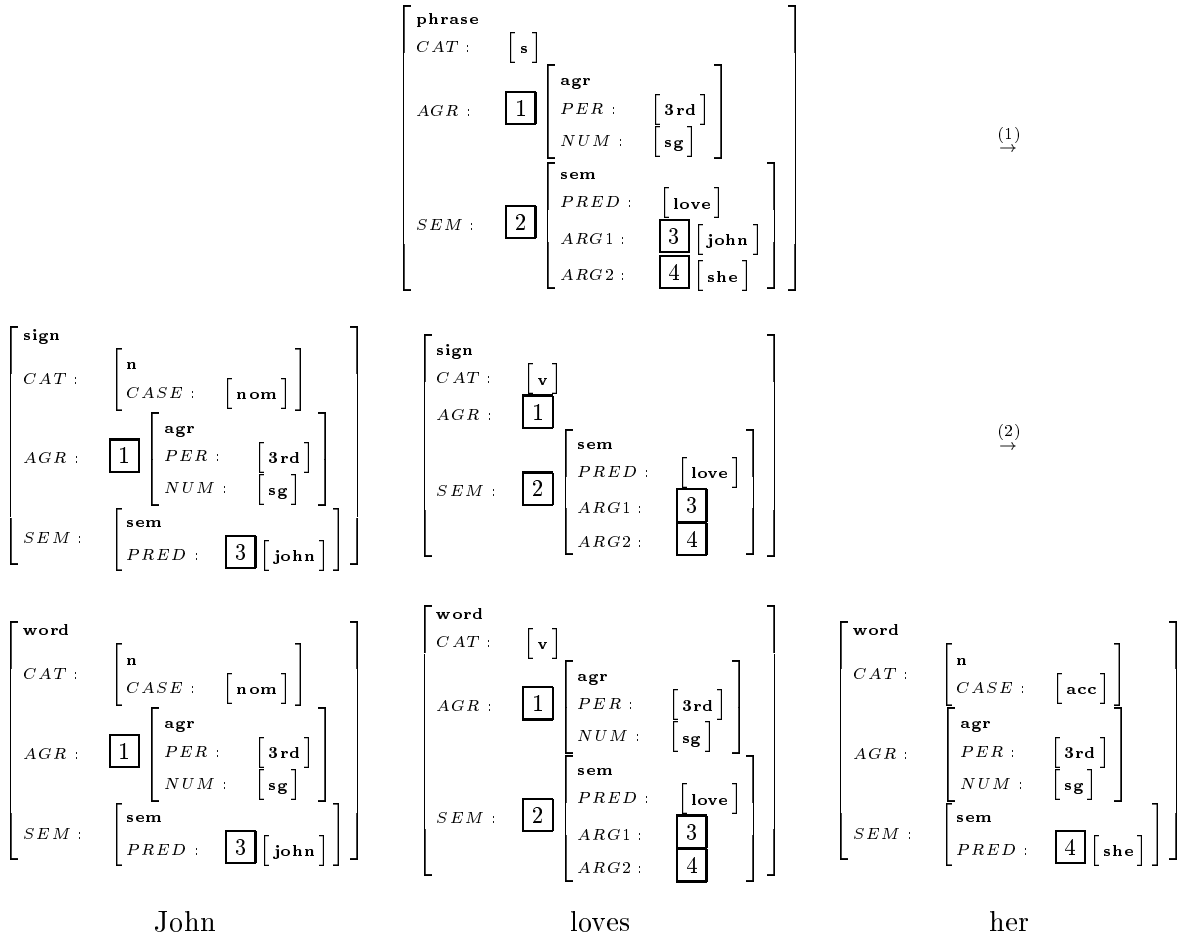

Figure 3: A leftmost derivation

Definition 4.8 (Language) *The* **language** *of a grammar G is* $L(G) = \{w \in \textsc{Words}^* \mid$

$w = w_1 \cdots w_n$ *and* $A_s \rightsquigarrow PT_w(1,n)\}$.

The derivation example of figure 3 shows that the sentence "John loves her" is in the language of the example grammar, since the derivation starts with a TFS that is more specific than the initial symbol and ends in a specification of the lexical entries of the sentences' words.

Parsing is a computational process triggered by some input string of words $w = w_1 \cdots w_n$ of length $n \geq 0$. For the following discussion we fix a particular grammar $G = (\mathcal{R}, A_s)$ and a particular input string w of length n. A *state* of the computation consists of a set of *items*.

Definition 4.9 (Items) *An* **item** *is a tuple* $[i, \sigma, j, k]$*, where* $i, j \in I\!N$*,* $i \leq j$*,* σ *is an MRS and* $0 < k \leq |\sigma|$*. an item is* **active** *if* $k < |\sigma|$*, otherwise it is* **complete**. ITEMS *is the collection of all items.*

If $[i, \sigma, j, k]$ is an item, $\sigma^{1\ldots k}$ is said to *span* the input from position $i+1$ to position j (the parsing invariant below motivates this term). σ and k can be seen as a representation of a *dotted rule*, or *edge*: during parsing all generated items are such that σ is (possibly more specific than) some grammar rule. k is a position in σ, indicating the location of the *dot*. The part of σ prior to the dot was already seen; the part following the dot is still expected. When the entire body of σ is seen, the edge becomes complete.

A *computation* amounts to successively generating items; we assume that item generation is done through a finite set of deterministic *operations* that create an item on the basis of previously generated (zero or more) items. Also, if an item was generated on the basis of some existing items, those items are not used again by the same operation. This assumption is realized by an important class of parsing algorithms known as *chart parsers*. A computation is *terminating* if and when no new items can be generated. A computation is *successful* if, upon termination, a complete item that spans the entire input and contains the initial symbol was generated: the final state of the computation should contain the item $[0, \sigma, n, 1]$, where $A \sim A_s$ and n is the input's length. Different algorithms assign different meanings to items, and generate them in various orders (see, e.g., (Shieber, Schabes, and Pereira, 1994; Sikkel, 1993)). To be as general as possible, we only assume that the following invariant holds:

Parsing invariant *In a computation triggered by* w*, if an item* $[i, \sigma, j, k]$ *is generated then* $\sigma^{1\ldots k} \rightsquigarrow PT_w(i+1, j)$.

One immediate consequence of the invariant is that for all the items $[i, \sigma, j, k]$ generated when parsing w, $0 \leq i \leq j \leq |w|$.

A parsing algorithm is required to be *correct*:

Correctness *A computation triggered by w is successful iff $w \in L(G)$.*
Although (Shieber, 1992) uses a different notation than (Wintner and Francez, 1995b), this property is proven by both.

5 Termination of parsing

It is well-known (see, e.g., (Pereira and Warren, 1983; Johnson, 1988)) that unification-based grammar formalisms are Turing-equivalent, and therefore the parsing problem is undecidable in the general case. However, for grammars that satisfy a certain restriction, termination of the computation can be guaranteed. We make use of the well-foundedness of subsumption (section 3) to prove that parsing is terminating for *off-line parsable* grammars.

To assure efficient computation and avoid maintenance of redundant items, many parsing algorithms employ a mechanism called *subsumption check* (see, e.g., (Shieber, 1992; Sikkel, 1993)) to filter out certain generated items. Define a (partial) order over items: $[i_1, \sigma_1, j_1, k_1] \preceq [i_2, \sigma_2, j_2, k_2]$ iff $i_1 = i_2, j_1 = j_2, k_1 = k_2$ and $\sigma_1 \sqsubseteq \sigma_2$. The subsumption filter is realized by preserving an item x only if no item x' such that $x' \preceq x$ was generated previously. Thus, for all items that span the same substring, only the most general one is maintained. (Shieber, 1992; Wintner and Francez, 1995b) prove that by admitting the subsumption check, the correctness of the computation is preserved.

Off-line parsability was introduced by (Kaplan and Bresnan, 1982) and was adopted by (Pereira and Warren, 1983), according to which "A grammar is off-line parsable if its context-free skeleton is not infinitely ambiguous". As (Johnson, 1988) points out, this restriction (which he defines in slightly different terms) "ensures that the number of constituent structures that have a given string as their yield is bounded by a computable function of the length of that string". The problem with this definition is demonstrated by (Haas, 1989): "Not every natural unification grammar has a context-free backbone".

A context-free backbone is inherent in LFG, due to the separation of c-structure from f-

structure and the explicit demand that the c-structure be context-free. However, this notion is not well-defined in HPSG, where phrase structure is encoded within feature structures (indeed, HPSG itself is not well-defined in the formal language sense). Such a backbone is certainly missing in Categorial Grammar, as there might be infinitely many categories. (Shieber, 1992) generalizes the concept of off-line parsability but doesn't prove that parsing with off-line parsable grammars is terminating. We use an adaptation of his definition below and provide a proof.

Definition 5.1 (Finite-range decreasing functions) *A total function $F : D \to D$, where D is a partially-ordered set, is* **finite-range decreasing** *(FRD) iff the range of F is finite and for every $d \in D, F(d) \preceq d$.*

Definition 5.2 (Strong off-line parsability) *A grammar is* **strongly off-line parsable** *iff there exists an FRD-function F from MRSs to MRSs (partially ordered by subsumption) such that for every string w and different MRSs σ, ρ such that $\sigma \rightsquigarrow \rho$, if $\sigma \rightsquigarrow PT_w(i+1, j)$ and $\rho \rightsquigarrow PT_w(i+1, j)$ then $F(\sigma) \neq F(\rho)$.*

Strong off-line parsability guarantees that any particular sub-string of the input can only be spanned by a finite number of MRSs: if a grammar is strongly off-line parsable, there can not exist an infinite set S of MRSs, such that for some $0 \leq i \leq j \leq |w|$, $s \rightsquigarrow PT_w(i+1, j)$ for every $s \in S$. If such a set existed, F would have mapped its elements to the set $\{F(s) \mid s \in S\}$. This set is infinite since S is infinite and F doesn't map two different items to the same image, and thus the finite range assumption on F is contradicted.

As (Shieber, 1992) points out, "there are non-off-line-parsable grammars for which termination holds". We use below a more general notion of this restriction: we require that F produce a different output on σ and ρ only if they are incomparable with respect to subsumption. We thereby extend the class of grammars for which parsing is guaranteed to terminate (although there still remain decidable grammars for which even the weaker restriction doesn't hold).

Definition 5.3 (Weak off-line parsability) *A grammar G is* **weakly off-line parsable** *iff there exists an FRD-function F from MRSs to MRSs (partially ordered by subsumption)*

such that for every string w and different MRSs σ, ρ such that $\sigma \rightsquigarrow \rho$, if $\sigma \rightsquigarrow PT_w(i+1, j)$, $\rho \rightsquigarrow PT_w(i+1, j)$, $\sigma \not\sqsubseteq \rho$ and $\rho \not\sqsubseteq \sigma$, then $F(\sigma) \neq F(\rho)$.

Clearly, strong off-line parsability implies weak off-line parsability. However, as we show below, the inverse implication does not hold.

We now prove that weakly off-line parsable grammars guarantee termination of parsing in the presence of acyclic MRSs. We prove that if these conditions hold, only a *finite* number of different items can be generated during a computation. The main idea is the following: if an infinite number of different items were generated, then an infinite number of different items must span the same sub-string of the input (since the input is fixed and finite). By the parsing invariant, this would mean that an infinite number of MRSs derive the same sub-string of the input. This, in turn, contradicts the weak off-line parsability constraint.

Theorem 5.4 *If G is weakly off-line parsable and MRSs are acyclic then every computation terminates.*

Proof: Fix a computation triggered by w of length n. By the consequence of the parsing invariant, the indices that determine the span of items are limited ($0 \leq i \leq j \leq n$), as are the dot positions ($0 < k \leq |\sigma|$). It remains to show that for every selection of i, j and k, only a finite number of MRSs are generated. Let $x = [i, \sigma, j, k]$ be a generated item. Suppose another item is generated where only the MRS is different: $x' = [i, \rho, j, k]$ and $\sigma \neq \rho$. If $\sigma \sqsubseteq \rho$, x' will not be preserved because of the subsumption test. If $\rho \sqsubseteq \sigma$, x can be replaced by x'. There is only a finite number of such replacements, since subsumption is well-founded for acyclic MRSs. Now suppose $\sigma \not\sqsubseteq \rho$ and $\rho \not\sqsubseteq \sigma$. By the parsing invariant, $\sigma^{1\ldots k} \rightsquigarrow PT_w(i+1, j)$ and $\rho^{1\ldots k} \rightsquigarrow PT_w(i+1, j)$. Since G is weakly off-line parsable, $F(\sigma^{1\ldots k}) \neq F(\rho^{1\ldots k})$. Since the range of F is finite, there are only finitely many items with equal span that are pairwise incomparable. Since only a finite number of items can be generated and the computation uses a finite number of operations, every computation ends within a finite number of steps. □

The above proof relies on the well-foundedness of subsumption, and indeed termination of parsing is not guaranteed by weak off-line parsability for grammars based on cyclic TFSs. Obviously, cycles can occur during unification even if the unificands are acyclic. However,

it is possible (albeit costly, from a practical point of view) to spot them during parsing. Indeed, many implementations of logic programming languages, as well as of unification-based grammars (e.g., ALE (Carpenter, 1992a)) do not check for cycles. If cyclic TFSs are allowed, the more strict notion of strong off-line parsability is needed. Under the strong condition the above proof is applicable for the case of non-well-founded subsumption as well.

To exemplify the difference between strong and weak off-line parsability, consider a grammar G that contains the following single rule:

$$\begin{bmatrix} \mathbf{t} & \\ f: & \boxed{1} \end{bmatrix} \Rightarrow \boxed{1} \begin{bmatrix} \mathbf{t} & \\ f: & \bot \end{bmatrix}$$

and the single lexical entry, w_1, whose category is:

$$Cat(w_1) = \begin{bmatrix} \mathbf{t} & \\ f: & \bot \end{bmatrix}$$

This lexical entry can be derived by an infinite number of TFSs:

$$\ldots \rightarrow \begin{bmatrix} \mathbf{t} & \\ f: & \begin{bmatrix} \mathbf{t} & \\ f: & \begin{bmatrix} \mathbf{t} & \\ f: & \bot \end{bmatrix} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{t} & \\ f: & \begin{bmatrix} \mathbf{t} & \\ f: & \bot \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{t} & \\ f: & \bot \end{bmatrix} = Cat(w_1)$$

It is easy to see that no FRD-function can distinguish (in pairs) among these TFSs, and hence the grammar is not strongly off-line parsable. The grammar is, however, *weakly* off-line parsable: since the TFSs that derive each lexical entry form a subsumption chain, the antecedent of the implication in the definition for weak off-line parsability never holds; even trivial functions such as the function that returns the empty TFS for every input are appropriate FRD-functions. Thus parsing is guaranteed to terminate with this grammar.

It might be claimed that the example rule is not a part of any grammar for a natural language. It is unclear whether the distinction between weak and strong off-line parsability is relevant when "natural" grammars are concerned. Still, it is important when the formal, mathematical and computational properties of grammars are concerned. We believe that a better understanding of formal properties leads to a better understanding of "natural" grammars as well. Furthermore, what might be seem un-natural today can be common practice in the future.

Acknowledgments

This work is supported by a grant from the Israeli Ministry of Science: "Programming Languages Induced Computational Linguistics". The work of the second author was also partially supported by the Fund for the Promotion of Research in the Technion. We wish to thank the anonymous referees for their enlightening comments.

References

Carpenter, Bob. 1992a. ALE – the attribute logic engine: User's guide. Technical report, Laboratory for Computational Linguistics, Philosophy Department, Carnegie Mellon University, Pittsburgh, PA 15213, December.

Carpenter, Bob. 1992b. *The Logic of Typed Feature Structures.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Haas, Andrew. 1989. A parsing algorithm for unification grammar. *Computational Linguistics*, 15(4):219–232, December.

Haddock, Nicholas, Ewan Klein, and Glyn Morill, editors. 1987. *Categorial Grammar, Unification and Parsing*, volume 1 of *Working Papers in Cognitive Science.* University of Edinburgh, Center for Cognitive Science.

Johnson, Mark. 1988. *Attribute-Value Logic and the Theory of Grammar*, volume 16 of *CSLI Lecture Notes.* CSLI, Stanford, California.

Kaplan, R. and J. Bresnan. 1982. Lexical functional grammar: A formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, Mass., pages 173–281.

Moshier, Drew M. and William C. Rounds. 1987. A logic for partially specified data structures. In *14th Annual ACM Symposium on Principles of Programming Languages*, pages 156–167, January.

Pereira, Fernando C. N. and David H. D. Warren. 1983. Parsing as deduction. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144, June.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar.* University of Chicago Press and CSLI Publications.

Shieber, Stuart M. 1992. *Constraint-Based Grammar Formalisms.* MIT Press, Cambridge, Mass.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1994. Principles and implementation of deductive parsing. Technical Report TR-11-94, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University, April.

Sikkel, Klaas. 1993. *Parsing Schemata.* Klaas Sikkel, Enschede.

Wintner, Shuly and Nissim Francez. 1995a. An abstract machine for typed feature structures. In *Proceedings of the 5th Workshop on Natural Language Understanding and Logic Programming*, pages 205–220, Lisbon, May.

Wintner, Shuly and Nissim Francez. 1995b. Parsing with typed feature structures. Technical Report LCL 95-1, Laboratory for Computational Linguistics, Technion, Israel Institute of Technology, Haifa 32000, Israel, December.

Contents

1 Introduction **1**

2 Theory of feature structures **2**

3 Well-foundedness **4**

4 Parsing **7**

5 Termination of parsing **14**

List of Figures

1 An infinite decreasing sequence of TFSs 7

2 An example grammar 10

3 A leftmost derivation 12