

Feature Structures as Weights in Finite State Morphology

Jan W. Amtrup

Bowne Global Solutions

jan.amtrup@bowneglobal.com

Abstract

We present a finite state morphology system augmented with typed feature structures as weights on transitions. Using a semiring interpretation, the weight of a result represents the possible linguistic interpretations of an input word, while the resulting character string itself represents the lemma of the input. Long-distance phenomena and infixation can be handled in an easy and elegant manner, simultaneously providing a seamless interface to subsequent linguistic processing modules.

1 Introduction

Finite state machines have been used extensively to describe morphological processes and build morphological analyzers and generators, based on the two-level view of morphology (Koskeniemi, 1983). The general view is that a series of characters on the surface (the input in analysis) is converted into a description of the properties of the surface string by employing a finite state transducer. The upper side of transitions (the output) contains characters that build the lemma of the input word, and it also encodes morphological properties of the input word (such as part of speech, case, tense, etc.) by providing markers on the lemma. For instance, the English form *walked* could be described as *walk+Verb+Past*. Newer models of finite state devices assign weights to transitions, which may

function as scores in several applications (see, e.g., (Mohri et al., 2000)).

While the properties of finite state transducers are well understood, and their application for morphology lends itself to clear and highly efficient machines, several areas could still be improved upon. The representation of both the lemma and morphological features in a single string of characters leads to interface complications. Subsequent modules in a larger NLP application have to be aware of the specific structure of the description of a word, and have to analyze the descriptions in order to retrieve the morphological properties. Moreover, using a linear structure as description complicates the formulation of long-distance dependencies, for instance in cases where both prefixes and suffixes are present and influence each other's distributions (Kiraz, 1994). Infixation may result in a discontinuous lemmatization, which also constitutes an interface problem.

Thus, extended models have been proposed that create a third level of description, containing a morphological description of the input word, leaving the other two levels (surface and lexeme) pure (Trost, 1991). In some cases, features are described separately on a separate high level, and later incorporated into a finite state mechanism (Kiraz, 1997). Others (Zajac, 1998) use feature structures as the upper level of a finite state transducer, merging lemma and morphological description in a feature structure instead of a linear string.

This paper provides an integrated view on finite state character transducers and unification of feature structures by interpreting feature structures

as weights on transitions. The accumulation of weights (scores) along a derivation path in a transducer is done by unification, and the weights along different paths through the transducer represent different morphological descriptions for an input word.

We present a formalization of feature structures as weights in section 2. Some architectural issues are discussed in section 3, and section 4 gives some examples of the usage of the analyzer. A description of the use of morphological grammars of this type for generation follows.

2 Formalization

We formalize finite state transducers as letter transducers with weights attached to transitions. The weights are sets of typed feature structures, which form a closed semiring under a specific interpretation. This **Unification Semiring** is the closed semiring $(2^{\mathcal{T}FS}, \cup, \cap, \emptyset, \{\top\})$, where

- $2^{\mathcal{T}FS}$ is the powerset of typed feature structures. We use well-typed feature structures (Carpenter, 1992) to describe morphological properties of input words.
- \cup is the operation of set union,
- \cap is the operation of pairwise unification of typed feature structures, and
- $\top \in \mathcal{T}FS$ is the most general feature structure (the top type).

It is easy to see that $(2^{\mathcal{T}FS}, \cup, \emptyset)$ is a monoid, employing set union with the empty set as identity. To see that $(2^{\mathcal{T}FS}, \cap, \{\top\})$ is a monoid as well, we first assume that there is a special feature structure $\perp \in \mathcal{T}FS$, the inconsistent feature structure. Whenever the unification of two feature structures fails, \perp is the result. Thus, feature structures are closed under unification. The operation \cap used here extends unifications over individual FSs to pairwise unification over the elements of sets of FSs. It is defined as

$$\cap(A, B) = \{f \mid a \in A \wedge b \in B \wedge f = a \cap b\} \quad (1)$$

The powerset of feature structures is closed under \cap . \cap is also associative. The set containing only

\top as feature structure is the identity of \cap , since \top is the identity of unification over feature structures.

Several additional properties have to hold to make a system a closed semiring:

- \cup is commutative, since set union is commutative.
- \emptyset is the annihilator of \cap :

$$\forall A \in 2^{\mathcal{T}FS} : A \cap \emptyset = \emptyset \cap A = \emptyset \quad (2)$$

This follows directly from the definition of \cap .

- \cup is idempotent, since set union is idempotent.
- \cap distributes over \cup , i.e. $\forall A, B, C \in 2^{\mathcal{T}FS} :$

$$\begin{aligned} A \cap (B \cup C) &= (A \cap B) \cup (A \cap C) \\ (A \cup B) \cap C &= (A \cap C) \cup (B \cap C) \end{aligned} \quad (3)$$

This is also easy to see, for instance by assuming that B and C are disjoint, and using induction over the elements of A .

Under the interpretation of sets of typed feature structures as weights on transitions, the output of a finite state machine yields the lemma of an input word. The feature structures encoded as the weights of the result describe the individual linguistic interpretations possible under the grammar. This view has the immediate consequence that we can apply results and algorithms for transducers that use other semirings for weights, such as the tropical semiring $(\mathbb{R}_+, \min, +, \infty, 0)$, e.g. (Mohri, 1997; Mohri et al., 2000; Piskorski, 2002).

3 Architecture

In the implementation using the formalization just described, finite state machines are constructed from regular expressions containing mappings from input characters to characters on the lexical level and feature structures. A morphological grammar consists of a number of named rules, each of which is a regular expression.

A sample rule describing some monosyllabic adjectives that display consonant doubling (like **big**, **bigger**, **biggest** in English) is:

```

Adj2 = Consonant* Vowel+
      (Consonant  Adj[deg: Positive]) |
      (Db1Consonant
       ((er:      Adj[deg: Comparative]) |
        (est:     Adj[deg: Superlative])));

```

Regular expressions can refer to other regular expressions by name, as for instance in `Db1Consonant` above, which could be defined as

```

Db1Consonant =
  AA:A | BB:B | CC:C ... zz:z;

```

The syntax of individual expressions is the familiar mapping using the colon as separator of the surface and lexical side. Typed feature structures are represented using square brackets, the type being written in front of the bracket (e.g. `Adj[deg: Comparative]`). The operators over regular expressions implemented thus far include concatenation, disjunction, Kleene iteration, and optionality.

Feature structures are not attached to every character transition. Instead, they are written by themselves, conceptually belonging to an ϵ transition on characters. Similarly, each character transition is implicitly marked with a neutral $\{\top\}$ as weight. Thus, the content of each transition is a triple (l, u, w) , where $l \in \Sigma$ is some character from the input alphabet (the lower side of the transition), $u \in \Delta$ is some character from the output alphabet (the upper side of the transition), and $w \in 2^{\mathcal{F}S}$ is a set of feature structures, the weight of the transition.

The conversion from regular expressions to transducers is performed according to (Ilie and Yu, 2002), performing only rudimentary ϵ -removal. After this initial step, all character transitions have the form $(l, u, \{\top\})$ with either l or u not ϵ , while all feature structure transitions have the form $(\epsilon, \epsilon, \{f\})$.

The next step ensures that we can perform ϵ -removal by pushing the weights onto transitions that have character content. They are pushed backwards in the direction of the start state of the transducer. In order to preserve the language of the transducer, a separate state has to be created if the start state of a weight-transition has multiple outgoing transitions. Then, the feature structure(s) comprising the weight are pushed onto the incoming transitions by unification. If a weight can not be pushed farther (because the start state of

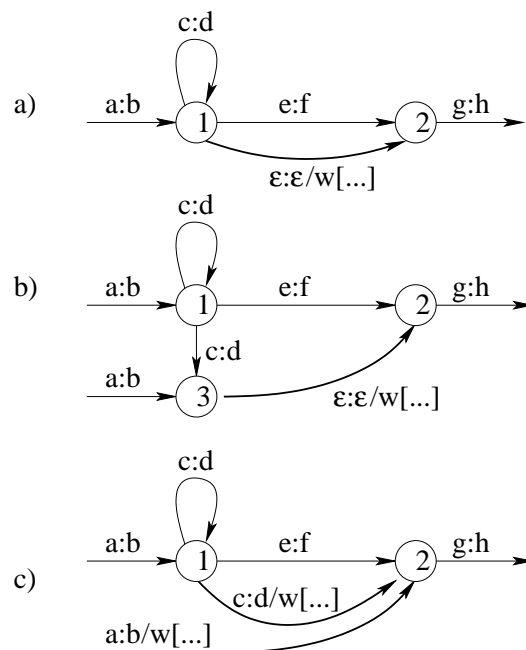


Figure 1: Pushing weights to avoid ϵ transitions

the weight-transition is the start state of the transducer), the annotated feature structures become part of the initial weight. Figure 1 shows an example for this procedure. Part a) shows a weight-bearing ϵ transition from state 1 to state 2. In part b), an additional state has been created, since state 1 had more than one outgoing transition. In part c) finally, the weight has been pushed onto the incoming transitions of state 3.

The next phases of compilation include the replacement of named transitions by the finite state transducers they represent and epsilon removal. The conversion into a deterministic transducer is not yet implemented. The result of the compilation process is a vector of transitions, each of which contain the lower and upper character, and a representative of a set of feature structures used as weight. Specially marked transitions indicate the end of the transitions exiting one state and the presence of a final state. This leads to a simple, linear storage method for the resulting transducers, and simplifies traversal procedures.

One should note that the transducers contain no lexical information, in contrast to other implementations of finite state morphology that include the lexicon of all stems integrated into the

machine. The main drawback of this exclusion is a decrease in efficiency, since all morphologically possible analyses of an input word are initially created, only to be excluded later during lexical lookup. However, this approach has the advantage that the treatment of unknown words can be identical to that of known words. Moreover, keeping the morphological analyzer separate from the lexicon allows for dynamic changes in the lexicon at runtime, no costly recompilation of possibly big analyzers is needed when words are added to or removed from the lexicon, which happens frequently during system development. If needed for a production system, the analysis for a specific rule can be restricted to only those words present in the lexicon by following the trie of lemmata whenever a character is produced on the upper level of the transducer. Alternatively, the lexical trie can be viewed as a finite state automaton and intersected with the morphological analyzer, allowing only known lemmata to be analyzed, as in other implementations.

4 Applications

The implementation of finite state morphology with unification is targeted at the use within machine translation systems employing typed feature structures throughout to represent linguistic objects. Currently, there are a largely complete English morphological grammar and fragments of a Persian grammar. The English grammar serves as development test bed and for evaluation purposes. Each surface form receives around 3.5 different morphological analyses. The speed of analysis is around 300 words per second, measured without optimizations on a very small machine (Pentium II, 400 MHz).

The Persian grammar is used to experiment with more complex morphotactic issues, such as long-distance phenomena within the Persian tense system (Megerdooomian, 2000). For instance, the final tense of verbs is captured by the combination of an optional prefix *my*, indicating imperfectiveness, and tense morphemes in the suffix. Using the type hierarchy underlying the feature structure system, the combinations of both pieces of information can be easily described in a rule:

Indicative =

```
(my: \~:?
  per.Verb[infl.tense: per.Imperfective])
PastStem
PastInfl
per.Verb[infl.tense: per.Past];
```

The two types *per.Imperfective* and *per.Past* interact to result in *per.Preterite* if the imperfective marker is not present, and in *per.Imperfect* if it is.

5 Generation

So far, we have described the incorporation of feature structures and unification from an analysis point of view, constructing the lemma and a morphological description starting out from a surface form. It is obvious that the lower and upper level of transitions in a finite state transducer can be exchanged, and that it can be used for analysis and generation likewise. The same is true for the formalization used here. However, the weights on transitions (feature structures) need to be used as restrictions in the case of generation. Only those surface forms generated are valid that carry with them a weight that is identical to the morphological description of the word to be generated. This check could be carried out once a full surface form has been generated; however, this clearly gives rise to significant overhead. Instead, each time a weight (a feature structure) is encountered on a transition, its compatibility with the incoming description (by unifiability or, stronger, subsumption) should be established.

Another important observation is that the unification of two feature structures is far more expensive than the comparison of two characters. Thus, in the analysis case, unifications should be carried out as late as possible, while they should be carried out as early as possible for generation. Consider the following equivalent rules, describing different forms of Persian Adverbs and Adjectives:

```
AdjAdv1 =
  Stem (per.Adj[] | per.Adv[])
  (per.Cmp[infl.comp: per.Posit] |
  (\~:? tr:
    (per.Cmp[infl.comp: per.Compar] |
    (yn: per.Cmp[infl.comp: per.Super1]))));
```

```
AdjAdv2 =
  (per.Adj[] | per.Adv[]) Stem
  ((per.Cmp[infl.comp: per.Posit]) |
  (per.Cmp[infl.comp: per.Compar] \~:? tr:)) |
```

(per.Cmp[infl.comp: per.Superl] \~:? tryn:the;lexical lemma, and feature structures to treat structure building and morphotactics.

The rule AdjAdv1 performs unification after the relevant morphemes have been recognized by the character level of the transducer, while rule AdjAdv2 carries the weights on transitions that are traversed before the surface evidence has been seen. While the second rule seems unnatural for the analysis of a surface form, it is actually preferable in the case of the generation from a morphological description. For instance, assume that the input to the generator is the feature structure

$$\left[\begin{array}{l} \text{Adj} \\ \text{LEMMA} \quad \text{"bzrg"} \\ \text{INFL} \quad \left[\begin{array}{l} \text{AdjInfl} \\ \text{COMP} \quad \text{Superl} \end{array} \right] \end{array} \right]$$

In this case, the first two alternatives — the positive and comparative forms — can be excluded early as inappropriate, an important factor for more complex morphotactic combinations.

There are at least two possible solutions to this problem, resulting either in different algorithms for the traversal or the compilation of the transducers. In one scenario, one could assume that the author of the grammar writes rules with analysis in mind, and always puts weights as far to the end of a regular expression as possible. In this case, analysis proceeds as usual. The generation of surface forms, however, would start at the end of a transducer, effectively using the reverse form for generation and constructing surface forms beginning at the end of a word.

The second solution involves pushing weights through the transitions of a transducer (Mohri, 1997; Mohri and Riley, 2001). For analysis, feature structures are pushed towards the end states of a transducer; for generation, they have to be pushed towards the start state.

6 Related work

A number of morphological analyzers using unification have been proposed. X2MorF (Trost, 1991) uses two-level rules annotated with feature structures as filters to account for morphotactics and structure-building. X2MorF is the closest relative to the work described here, since it uses three distinct levels of representation: The surface form,

(Zajac, 1998) presents a transducer system comprised of characters as the lower level and feature structures as the upper level. The lexical level is encoded as string variables within the resulting feature structures, which poses certain combinatorial problems with effects such as consonant doubling, especially during morphological generation.

(Kiraz, 1997) incorporates simple feature structures (flat structures with atomic feature values only) directly as distinct symbols on the upper level of a transducer. Feature values might be drawn from finite sets of possible values, which is expressed as disjunction on the formalism level. While this approach is suitable for simple restrictions and the encoding of some part of the structure directly in a transducer, (Kiraz, 1997) notes that "...the morphotactic unification of lexical entries ... is best dealt with at the morphotactic level using a unification based formalism".

7 Conclusion

We have presented a formalization of typed feature structures as weights attached to the transitions of a finite state transducer, applied to the problem of morphological analysis and generation. The formalization uses a semiring representation similar to approaches used to encode scores within analyzers. It retains the formal framework of those transducers, providing a rigid view on the incorporation of feature structures within the finite state paradigm. The target applications are NLP systems using unification-based formalisms to encode linguistic knowledge, which benefit from a rich description language and a simple interface between different components for the analysis of natural language.

References

- Bob Carpenter. 1992. *The Logic of Typed Feature Structures*. Cambridge University Press.
- Lucian Ilie and Sheng Yu. 2002. Constructing NFAs by Optimal Use of Positions in Regular Expressions. In *Proceedings of the 13th CPM*.
- George Anton Kiraz. 1994. Multi-Tape Two-Level Morphology: a Case Study in Semitic Non-linear

- Morphology. In *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, Japan.
- George Anton Kiraz. 1997. Compiling Rule Formalisms with Rule Features into Finite-State Automata. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics*, Madrid, Spain.
- Kimmo Koskenniemi. 1983. Two-level Model for Morphological Analysis. In *Proceedings of the 8th International Conference on Artificial Intelligence, IJCAI'83*, pages 683–685, Karlsruhe, Germany.
- Karine Megerdooian. 2000. Unification-Based Persian Morphology. In Alexander Gelbukh, editor, *Proceedings of CICLing 2000*, Mexico City, Mexico. Centro de Investigacion en Computacion-IPN.
- Mehryar Mohri and Michael Riley. 2001. A Weight Pushing Algorithm for Large Vocabulary Speech Recognition. In *Proceedings of EuroSpeech 2001*, Aalborg, Denmark, September.
- Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The Design Principles of a Weighted Finite-State Transducer Library. *Theoretical Computer Science*, 231:17–32, January.
- Mehryar Mohri. 1997. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23:269–311.
- Jakub Piskorski. 2002. DFKI Finite-State Machine Toolkit. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany.
- Harald Trost. 1991. A morphological component for the recognition and generation of word forms in natural language understanding systems: Integrating two-level morphology and feature unification. *Applied Artificial Intelligence*, 4(4):411–457.
- Rémi Zajac. 1998. Feature Structures, Unification and Finite-State Transducers. In *International Workshop on Finite State Methods in Natural Language Processing*, Ankara, Turkey. Bilkent University.