Efficient Finite State Unification Morphology

Jan W. Amtrup
Kofax Image Products
5120 Shoreham Pl.
San Diego, CA 92122, USA,
Jan_Amtrup@mohomine.com

Finite state transducers are highly efficient means for the representation and processing of morphological knowledge. However, the string representations normally used do not easily provide the rich and detailed linguistic descriptions needed for complex applications in Computational Linguistics, and long-distance phenomena are not easily modeled. This paper describes the use of typed feature structure as weights on transitions in a finite state transducer to represent linguistic objects. This method provides a seamless integration into other linguistic processing modules and facilitates the description of certain morphological phenomena. By using a pre-computation model of unification, we avoid the runtime complexity of unification and achieve a level of efficiency comparable to character-based automata based on other weight structures.

1 Introduction

This paper describes the efficient combination of finite state morphology mechanisms with detailed linguistic descriptions based on a typed feature structure formalism. Finite state systems are the predominant choice for morphological analysis and generation, due to their high run-time efficiency and the inherent simplicity of the mechanism. They implement the two-level view of morphology (Koskenniemi, 1983). The paradigm is usually that a finite state transducer (FST) takes a series of surface characters as input and converts those into a sequence of characters containing the lemma together with characters that describe the morphological properties of an input word. For example, the Persian singular superlative adjective bzrgtryn (biggest) would receive an analysis as in (1).

$$\mathsf{bzrgtryn} \Longrightarrow \mathsf{bzrg{+}Adj{+}Superl{+}Sing} \qquad (1)$$

Unification formalisms based on feature structures, on the other hand, provide a much richer representation mechanism for linguistic knowledge. Instead of a single, linear level as in FST models, feature structures are graphs that can reach considerable complexity. For instance, (2) could be an appropriate analysis for the aforementioned bzrgtryn.

$$\begin{array}{c}\mathsf{bzrgtryn} \Longrightarrow \\ \begin{bmatrix} \textit{Adj} & \\ \text{LEMMA} & \text{"bzrg"} \\ \text{SURF} & \text{"bzrgtryn"} \\ \text{LEX} & \begin{bmatrix} \textit{LexMorph} & \\ \text{REGULAR} & \text{True} \end{bmatrix} \\ \text{INFL} & \begin{bmatrix} \textit{AdjInfl} & \\ \text{COMPARISON} & \text{Superlative} \\ \text{NUMBER} & \text{Singular} \end{bmatrix} \end{bmatrix}\end{array} \qquad (2)$$

The incorporation of typed feature structures into conventional FST models is advantageous for a number of reasons:

- Feature structures provide a richer description of linguistic objects, resulting in an adequate interface to higher-level analysis (e.g. syntax and semantics). This representation is directly available and does not have to be extracted from the result string as in conventional FSTs.
- Surface and lemma characters and morphological properties are no longer treated uniformly. This inability of conventional FSTs to distinguish between characters and features is problematic, for instance in the case of the analysis of infixation. Moreover, the linear representation of morphological features imposes an artificial ordering that might complicate the formulation of morphotactic facts.
- Long-distance dependencies, also a source of complication for FSTs, can easily be accounted for using unification.

This paper presents the combination of finite state transducers with typed feature structure unification formalism based on the concept of weighted FSTs (Mohri, 1997). The interpretation of feature structures as weights on transitions of FSTs allows an integrated, elegant view of the combination of both approaches. The system presented here extends Amtrup (2003) in describing the analysis of sequences, and in providing a constant time interpretation of the unification operation within a morphological analyzer. The basic formalization is described in section 2. In section 3, we outline some aspects of the implementation and the way knowledge sources for this method are constructed. In section 4, we present a solution for the great disparity in run-time efficiency between FSTs and unification formalisms, rendering unification morphology as efficient in principle as conventional FST methods.

2 The Unification Semiring

In order to introduce feature structures into a two-level morphology system, we use weighted finite state transducers (WFSTs). In a WFST, each transition is associated with a certain weight. During the course of the traversal of a machine, the values of the weights are combined using a multiplicative operation. If several paths through a machine yield a result, the accumulated weights are further combined using an additive operation. The elementary weights are organized in the structure of a *semiring* to ensure proper processing (Mohri, 1997). Most commonly, real numbers are used as weights, for instance in applications in speech processing. There, the tropical semiring $(\mathbb{R}_+, \min, +, \infty, 0)$ is used, which is constructed from the positive real numbers, using the minimum operator as multiplication and the addition for real numbers (Mohri et al., 2000).

For the purposes here, we define the *Unification Semiring* (Amtrup, 2003) as follows: It is the structure

$$(2^{\mathcal{TFS}}, \cup, \bigcap, \emptyset, \{\top\}) \tag{3}$$

where

- $2^{\mathcal{TFS}}$ is the power set of typed feature structures. We use well-typed feature structures (Carpenter, 1992) to describe morphological properties of input words. $\top \in \mathcal{TFS}$ is the most general feature structure, called **top**. $\bot \in \mathcal{TFS}$ is the inconsistent feature structure, called **bottom**. We assume that the unification of two incompatible feature structures results in $\bot$.
- $\cup$ is the operation of set union.
- $\bigcap$ is the operation of pairwise unification of typed feature structures, defined as

$$\begin{aligned}\bigcap(A, B) = \{f | a \in A \wedge b \in B \wedge \\ f = a \cap b \wedge f \neq \bot\}\end{aligned} \tag{4}$$

  $\cap$ is the unification operator over typed feature structures. The abovementioned $\top$ is the identity of unification:

$$\forall x \in \mathcal{TFS} \setminus \{\bot\} : x \cap \top = \top \cap x = x. \tag{5}$$

  The computed set contains the results of all successful unifications of any two feature structures from the operand sets.

If this interpretation of weights is used with a finite state transducer modeling the relation between surface strings and lemmas, the result of the traversal of a WFST is the lemma associated with the input word. The feature structures contained in the final weight represent the different linguistic interpretations of the input word. This accounts for the interface problems mentioned in the introduction by separating different kinds of information computed by the analyzer, and provides an immediate interface to other processing components in a complex NLP system. Moreover, work on WFSTs can be immediately integrated, e.g. Piskorski (2002), Mohri and Riley (2001), and Mohri (2002).

The augmentation of finite state transducers with feature structures as weights does not change the explanatory power of the basic regular approach. The weights merely provide an elegant way of conveying linguistic properties of a word. Due to the finite nature of the universe of possible weights and interpretations (see section 4 below), any weight structure could in principle also be realized by lexical description symbols as shown in example (1). Additionally, the unification operation between two weights along the path of an analysis might fail, in which case that particular path is abandoned. This behavior is used to express long distance morphotactic facts. It can be seen as analogous to unsuccessful read operations on the register contents of Finite State Register Automata, for which Cohen-Sygal et al. (2003) have shown that it does not change the equivalence to regular languages, but only reduces the size of the particular automaton.

3 Architecture and Applications

Morphological grammars describe how surface characters are mapped to characters in the lemma of a word, and provide feature structures denotating the morphological features of a word. For instance, the rule in (6) accounts for the distinction between positive, comparative, and superlative adjectives in Persian.

```
Comparative =
  Stem
  ( HasComparisonSign[
      infl.comparison: Null]
  | (\~:? tr:
       HasComparisonSign[                (6)
         infl.comparison: Comparative])
  | (\~:? tryn:
       HasComparisonSign[
         infl.comparison: Superlative])
  );
```

This rule demonstrates some of the properties of morphological grammars in our system:

- Rules are named regular expressions over characters and feature structures. They can refer to the transducers defined by other rules by name. For instance, the rule `Comparative` incorporates the rule `Stem` in (6).
- Regular expressions over characters are written as mappings between surface and lemma. The format is the familiar notation with a colon separating lower and upper side content. For instance, `tryn:` denotes that the string `tryn`, found in the surface, does not generate any characters in the lemma. Internally, all characters are represented in Unicode. We implemented most of the common operators for regular expressions, namely concatenation, disjunction, Kleene iteration, optionality, intersection, and composition. Grouping of expressions is done with parentheses. Negation and difference have not yet been implemented.
- Feature structures describe morphological properties. We use well-typed feature structures (Carpenter, 1992) in this formalization. The type of the feature structure is written in front of the pair of square brackets, which then includes the features.
- Feature structures in the input grammar are conceptually assigned to ϵ-transitions over characters. This makes grammars easier to read and write. During compilation, weights are pushed onto character-bearing transitions to facilitate ϵ-removal.

3.1 Applications

The combination of finite state transducers and typed feature structures presented here is intended to be used mainly within machine translation systems using a unification formalism for other aspects of representation. One of the primary advantages we see is the seamless integration possible by using a common mechanism to describe linguistic objects. The prototype application is a Persian-English Machine Translation system. On the analysis side, two finite state grammars are employed, one for tokenization and one for Persian morphology. English morphological generation will also be handled by a WFST.

The Persian morphological analyzer contains about 60 rules of varying complexity. Consider rule (7) as an example.

```
Indicative =
  ((my: \~:? Verb[tense: Imperfective])
  |          Verb[tense: NonImperfective]
  )
  PastStem
  PastInfl
  Verb[tense: Past];
                                                (7)
```

It demonstrates how unification can be used to simultaneously build structure and handle certain long-distance phenomena. In this case, the optional imperfective prefix `my` interacts with the inflectional suffix morphemes. Due to the presence of feature structures as weights, their underlying type system can be used to express the permitted combinations. Here, the two types `Imperfective` and `Past` interact to result in `Imperfect`, while `NonImperfective` and `Past` result in `Preterite`.

Identical behavior can be expressed in conventional FST morphology systems by applying a filter to an over-generating analyzer. The filter restricts the analyses to cases that adhere to the particular morphotactic rules in question. The disadvantage of this method of treating constraints is that the filtered FST is almost twice as large as the unconstrained one.

The English morphological grammar in our prototype is simple in comparison, and only describes inflectional variation. The morphological system was tested on a corpus of approximately 100,000 Persian words of news items. The analysis speed is around 300 words per second (without transducer minimization) on a small machine (Pentium II, 400MHz). On average, five different morphological analyses are produced for each input token.

Generation of surface forms from stems and associated linguistic descriptions can be performed analogous to morphological analysis by traversing the WFST in reverse order. However, in order to ensure that all morphological features of the word to be generated are taken into account, the weight resulting from the traversal through the reversed transducer must be identical to (or at least subsumed by) the original description. Moreover, for reasons of efficiency, it might be appropriate to check for compatibility with that description during the generation process (cf. (Amtrup, 2003)).

3.2 Sequences of Analyses

It is often necessary to analyze more than one input word at a time during the traversal of a WFST. For instance, the stream of input characters for a contraction like I've should lead to the creation of two conceptual words, one for the pronoun and one for the verb. Moreover, a finite state model might not only be used as a morphological analyzer, but also as a tokenizer. In this application, a continuous stream of characters has to be separated into a sequence of token descriptions.

Conventional finite state models are able to do this without any modification. Since the upper level is a sequence of characters, one particular character (e.g. the space character) can be designated to separate analyses belonging to different words, and processing proceeds as usual. Using feature structures as weights complicates this. They contain morphological properties of words, and features are accumulated by unification. This is beneficial in the case of long-distance dependencies, as already mentioned. However, it precludes the construction of separate analyses for two or more words without modification. For instance, consider the above example I've. The weights along the path through the WFST would initially describe a pronoun. Once the contracted have is recognized, the weights would belong to a verbal interpretation. The combination (by unification) of both is most likely not successful.

To account for this, we introduce a special *eject* transition that instructs the mechanism to finish the analysis of one word and start with a new word. This transition is written as @ in a grammar and contains the Unicode character \uFFFF on both the lower and the upper side. Using this method, a simple tokenizer could be written as in (8).

```
Token = [A-Za-z0-9]: +;
Space = \u0020: | \u0008: |
        \u000a: | \u000d:;
                                        (8)
Sentence =
  (Token Space+ @)* Token Space* @;
```

During compilation, character transitions with this specific runtime semantics can be treated as any other transition for the most part. Only operations that either could separate the lower and upper side of a transition (e.g. ϵ-removal) or that affect the position of weights within the WFST (e.g. weight pushing, cf. Mohri and Riley (2001)) need to receive additional consideration. During compilation, the tokens used to record the progress of the traversal through the WFST have to be augmented to account for the possibility of multiple results.

4 Constant Time Unification

One major obstacle in practical terms for the combination of finite state methods and unification approaches is the large disparity in runtime efficiency. FST machines are well known for their very high speed. Attaching unification operations to a sizeable portion of the transitions in an FST carries the risk of reducing this speed considerably. This is due to the fact that the predominantly used unification algorithm (Huet's algorithm, cf. Knight (1989)) is almost linear in the size of the feature structures involved, and not constant as operations on other semirings. Such a reduction in efficiency might render the combination approach unusable under certain circumstances. In this section, we describe a method to eliminate the linear complexity of feature structure unification during the runtime of a WFST.

We use a pre-computation method that calculates all admissible combinations of weight elements during the compilation of the WFST, and reduce the unification operation to table lookup, which is constant in complexity. Thus, finite state morphology with unification can be performed as efficiently as with other semirings.

The weight elements attached to transitions in a compiled grammar are sets of feature structures. The elements of these sets were either introduced directly in the original source grammar, or are constructed from those during compilation by unification.

Since unification is commutative, associative, and — most importantly — idempotent, the transitive closure of the weights for a particular grammar under pairwise unification exists and is finite.

Formally, this transformation means establishing an isomorphism between the unification semiring and a semiring whose inner monoid consists of a set of natural numbers equipped with an operation $\otimes$ that performs table lookup according to the precomputed unification table. The identity element is the number corresponding to $\{\top\}$ in the original monoid. We decided not to change the additive operation of the outer monoid, which is linear set union. This does not seem to be crucial, since it is only carried out once at the end of an analysis.

In order to assess the runtime ramifications of this modification, we compiled the Persian morphological grammar described above with and without using the table lookup conversion. The original grammar contains 166 distinct weight elements. Constructing the transitive closure increases this number significantly, namely to 31536. With 9.1 million active cells, the unification table is only lightly populated (density 0.916%). For space reasons, we decided to use a sparse vector representation in one dimension of the table, which renders lookup logarithmic in the number of active cells in a row (there are 289 such cells on average). We tested the grammar on the corpus of 100,000 words of Persian news articles mentioned above, and achieved an efficiency improvement of 87%.

5 Related Work

Several extended models have been proposed in the literature that incorporate feature structures and unification within a finite state morphology model. The approach closest to the one presented here (but without the formalization of feature structure weights as a semiring) is Trost (1991), who describes feature structures as filters on transitions in finite state transducers. Those filters describe morphotactic restrictions and are responsible for building structural descriptions.

Other models restrict the complexity of feature structures allowed to a level at which they can be easily integrated into the finite state paradigm. Kiraz (1997) incorporates simple feature structures (pairs of feature names and atomic values) directly as distinct symbols on the upper level of an FST. Beesley and Karttunen (2003) use *flag diacritics* as a mechanism to introduce some unification capabilities on atomic symbols into an FST. They provide two implementations for this model. One uses explicit symbols and operators, using special transitions carrying the symbols and instructions. Alternatively, the feature operations can be compiled directly into the machine, since they are equivalent to conventional FST operations. Both of these methods target the formulation of morphotactic restrictions that are difficult or awkward to express in terms of finite state operators and are not concerned with structure building. A formal definition of a similar extension to FSTs with additional operators to facilitate the use of atomic registers appears in Cohen-Sygal et al. (2003). The two last approaches allow destructive write operations on the additional memory of the transducer, which a unification operation as described here does not permit. The effects of sequences of destructive operations on generation using the same automaton are not immediately clear.

Zajac (1998) modifies the architecture of an FST to use feature structures instead of characters as the upper level in an FST. Instead of concatenation, unification is used as sequencing operation on that level. The surface characters that take part in building the lemma are initially collected in string variables, and are concatenated within a feature structure to form the final lemma. This concatenation leads to problems during generation, when all possible segmentations of a lemma have to be explored.

6 Conclusion

We have presented a formalization of the integration of typed feature structures and unification into a finite state model of morphology. This enables us to create a uniform representational level across all components of a natural language processing system. The formulation of sets of feature structures as elements of a weight system forming a semiring allows for a rigid view of the resulting weighted finite state transducer, allowing us to adopt existing results and algorithms.

We showed how the current implementation is used in a Persian-English machine translation system, and described a technical extension to account for multiple analyses. The run-time speed is acceptable with around 300 words per second.

Finally, we described the reformulation of unification as table lookup during the runtime of a morphological analyzer, which results in an efficiency equal to that of other weighted finite state transducers. In experiments, this modification resulted in a speed improvement of 87%.

References

Jan W. Amtrup. 2003. Feature Structures as Weights in Finite State Morphology. In *Proceedings of the Workshop on Finite State Methods in Natural Language Processing*, Budapest, Hungary, April.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology.* CSLI Publications.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures.* Cambridge University Press.

Yael Cohen-Sygal, Dale Gerdemann, and Shuly Wintner. 2003. Computational Implementation of Non-Concatenative Morphology. In *Proceedings of the Workshop on Finite State Methods in Natural Language Processing*, pages 59–66, Budapest, Hungary, April.

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.

Kevin Knight. 1989. Unification: A Multi-Disciplinary Survey. *ACM Computer Surveys*, 21:98–124.

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.

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.

Mehryar Mohri. 2002. Generic ϵ-Removal and Input ϵ-Normalization for Weighted Transducers. *International Journal of Foundations of Computer Science*, 13(1):129–143.

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.