Using Mazurkiewicz Trace Languages for Partition-Based Morphology

François Barthélemy
CNAM Cedric, 292 rue Saint-Martin, 75003 Paris (France)
INRIA Atoll, domaine de Voluceau, 78153 Le Chesnay cedex (France)
barthe@cnam.fr

Abstract

Partition-based morphology is an approach of finite-state morphology where a grammar describes a special kind of regular relations, which split all the strings of a given tuple into the same number of substrings. They are compiled in finite-state machines. In this paper, we address the question of merging grammars using different partitionings into a single finite-state machine. A morphological description may then be obtained by parallel or sequential application of constraints expressed on different partition notions (e.g. morpheme, phoneme, grapheme). The theory of Mazurkiewicz Trace Languages, a well known semantics of parallel systems, provides a way of representing and compiling such a description.

1 Partition-Based Morphology

Finite-State Morphology is based on the idea that regular relations are an appropriate formalism to describe the morphology of a natural language. Such a relation is a set of pairs, the first component being an actual form called *surface form*, the second component being an abstract description of this form called *lexical form*. It is usually implemented by a finite-state transducer. Relations are not oriented, so the same transducer may be used both for analysis and generation. They may be non-deterministic, when the same form belongs to several pairs. Furthermore, finite state machines have interesting properties, they are composable and efficient.

There are two main trends in Finite-State Morphology: rewrite-rule systems and two-level rule systems. Rewrite-rule systems describe the morphology of languages using contextual rewrite rules which are easily applied in cascade. Rules are compiled into finite-state transducers and merged using transducer composition (Kaplan and Kay, 1994).

The other important trend of Finite-State Morphology is Two-Level Morphology (Koskenniemi, 1983). In this approach, not only pairs of lexical and surface strings are related, but there is a one-to-one correspondence between their symbols. It means that the two strings of a given pair must have the same length. Whenever a symbol of one side does not have an actual counterpart in the other string, a special symbol 0 is inserted at the relevant position in order to fulfill the same-length constraint. For example, the correspondence between the surface form `spies` and the morpheme concatenation `spy+s` is given as follows:

s	p	y	0	+	s
s	p	i	e	0	s

Same-length relations are closed under intersection, so two-level grammars describe a system as the simultaneous application of local constraints.

A third approach, Partition-Based Morphology, consists in splitting the strings of a pair into the same number of substrings. The same-length constraint does not hold on symbols but on substrings. For example, `spies` and `spy+s` may be partitioned as follows:

s	p	y	+	s
s	p	ie	ϵ	s

The partition-based approach was first proposed by (Black et al., 1987) and further improved by (Pulman and Hepple, 1993) and (Grimley-Evans et al.,

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 928–935,
Prague, Czech Republic, June 2007.

1996). It has been used to describe the morphology of Syriac (Kiraz, 2000), Akkadian (Barthélemy, 2006) and Arabic Dialects (Habash et al., 2005). These works use multi-tape transducers instead of usual two tape transducers, describing a special case of n-ary relations instead of binary relations.

Definition 1 *Partitioned n-relation*
A partitioned n-relation is a set of finite sequences of string n-tuples.

For instance, the n-tuple sequence of the example $(spy, spies)$ given above is $(s, s)(p, p)(y, ie)(+, \epsilon)(s, s)$. Of course, all the partitioned n-relations are not recognizable using a finite-state machine. Grimley-Evans and al. propose a partition-based formalism with a strong restriction: the string n-tuples used in the sequences belong to a finite set of such n-tuples (the centers of context-restriction rules). They describe an algorithm which compiles a set of contextual rules describing a partitioned n-relation into an epsilon-free letter transducer. (Barthélemy, 2005) proposed a more powerful framework, where the relations are defined by concatenating tuples of independent regular expressions and operations on partitioned n-relations such as intersection and complementation are considered.

In this paper, we propose to use Mazurkiewicz Trace Languages instead of partitioned relation as the semantics of partition-based morphological formalisms. The benefits are twofold: firstly, there is an extension of the formal power which allows the combination of morphological description using different partitionings of forms. Secondly, the compilation of such languages into finite-state machines has been exhaustively studied. Their closure properties provide operations useful for morphological purposes.

They include the concatenation (for instance for compound words), the intersection used to merge local constraints, the union (modular lexicon), the composition (cascading descriptions, form recognition and generation), the projection (to extract one level of the relation), the complementation and set difference, used to compile contextual rules following the algorithms in (Kaplan and Kay, 1994), (Grimley-Evans et al., 1996) and (Yli-Jyrä and Koskenniemi, 2004).

The use of the new semantics does not imply any change of the user-level formalisms, thanks to a straightforward homomorphism from partitioned n-relations to Mazurkiewicz Trace Languages.

2 Mazurkiewicz Trace Languages

Within a given n-tuple, there is no meaningful order between symbols of the different levels. Mazurkiewicz trace languages is a theory which expresses partial ordering between symbols. They have been defined and studied in the realm of parallel computing. In this section, we recall their definition and some classical results. (Diekert and Métivier, 1997) gives an exhaustive presentation on the subject with a detailed bibliography. It contains all the results mentioned here and refers to their original publication.

2.1 Definitions

A Partially Commutative Monoid is defined on an alphabet Σ with an independence binary relation I over $\Sigma \times \Sigma$ which is symmetric and irreflexive. Two independent symbols commute freely whereas non-independent symbols do not. I defines an equivalence relation $\sim_I$ on Σ^*: two words are equivalent if one is the result of a series of commutation of pairs of successive symbols which belong to I. The notation $[x]$ is used to denote the equivalence class of a string x with respect to $\sim_I$.

The Partially Commutative Monoid $M(\Sigma, I)$ is the quotient of the free monoid Σ^* by the equivalence relation $\sim_I$.

The binary relation $D = (\Sigma \times \Sigma) - I$ is called the dependence relation. It is reflexive and symmetric.

φ is the canonical homomorphism defined by:

$$\begin{array}{cccc} \varphi: & \Sigma^* & \rightarrow & M(\Sigma, I) \\ & x & \mapsto & [x] \end{array}$$

A Mazurkiewicz trace language (abbreviation: trace language) is a subset of a partially commutative monoid $M(\Sigma, I)$.

2.2 Recognizable Trace Languages

A trace language T is said *recognizable* if there exists an homomorphism ν from $M(\Sigma, I)$ to a finite monoid S such that $T = \nu^{-1}(F)$ for some $F \subseteq S$. A recognizable Trace Language may be implemented by a Finite-State Automaton.

A trace $[x]$ is said to be connected if the dependence relation restricted to the alphabet of $[x]$ is a connected graph. A trace language is connected if all its traces are connected.

A string x is said to be in lexicographic normal form if x is the smallest string of its equivalence class $[x]$ with respect to the lexicographic ordering induced by an ordering on Σ. The set of strings in lexicographic normal form is written $LexNF$. This set is a regular language which is described by the following regular expression:
$LexNF = \Sigma^* - \bigcup_{(a,b)\in I, a<b} \Sigma^* b(I(a))^* a \Sigma^*$
where $I(a)$ denotes the set of symbols independent from a.

Property 1 *Let $T \subseteq M(\Sigma, I)$ be a trace language. The following assertions are equivalent:*

- *T is recognizable*
- *T is expressible as a rational expression where the Kleene star is used only on connected languages.*
- *The set $Min(T) = \{x \in LexNF | [x] \in T\}$ is a regular language over Σ^*.*

Recognizability is closely related to the notion of iterative factor, which is the language-level equivalent of a loop in a finite-state machine. If two symbols a and b such that $a < b$ belong to a loop, and if the loop is traversed several times, then occurrences of a and b are interlaced. For such a string to be in lexicographic normal form, a dependent symbol must appear in the loop between b and a.

2.3 Operations and closure properties

Recognizable trace languages are closed under intersection and union. Furthermore, $\text{Min}(T_1) \cup \text{Min}(T_2) = \text{Min}(T_1 \cup T_2)$ and $\text{Min}(T_1) \cap \text{Min}(T_2) = \text{Min}(T_1 \cap T_2)$. It comes from the fact that intersection and union do not create new iterative factor. The property on lexicographic normal form comes from the fact that all the traces in the result of the operation belong to at least one of the operands which are in normal form.

Recognizable trace language are closed under concatenation. Concatenation do not create new iterative factors. The concatenation $\text{Min}(T_1)\text{Min}(T_2)$ is not necessarily in lexicographic normal form. For instance, suppose that $a > b$. Then $\{[a]\}.\{[b]\} = \{[ab]\}$, but $\text{Min}(\{[a]\}) = a, \text{Min}(\{[b]\}) = b$, and $\text{Min}(\{[ab]\}) = ba$.

Recognizable trace languages are closed under complementation.

Recognizable Trace Languages are not closed under Kleene star. For instance, $a < b$, $\text{Min}([ab]^*) = a^n b^n$ which is known not to be regular.

The projection on a subset S of Σ is the operation written π_S, which deletes all the occurrences of symbols in $\Sigma - S$ from the traces. Recognizable trace languages are not closed under projection. The reason is that the projection may delete symbols which makes the languages of loops connected.

3 Partitioned relations and trace languages

It is possible to convert a partitioned relation into a trace language as follows:

- represent the partition boundaries using a symbol ω not in Σ.
- distinguish the symbols according to the component (tape) of the n-tuple they belong to. For this purpose, we will use a subscript.
- define the dependence relation D by:
  - ω is dependent from all the other symbols
  - symbols in Σ sharing the same subscript are mutually dependent whereas symbols having different subscript are mutually independent.

For instance, the *spy* n-tuple sequence $(s,s)(p,p)(y,ie)(+,\epsilon)(s,s)$ is translated into the trace $\omega s_1 s_2 \omega p_1 p_2 \omega y_1 i_2 e_2 \omega +_1 \omega s_1 s_2 \omega$. The figure 1 gives the partial order between symbols of this trace.

The dependence relation is intuitively sound. For instance, in the third n-tuple, there is a dependency between i and e which cannot be permuted, but there is no dependency between i (resp. e) and y: i is neither before nor after y. There are three equivalent permutations: $y_1 i_2 e_2$, $i_2 y_1 e_2$ and $i_2 e_2 y_1$. In an implementation, one canonical representation must be chosen, in order to ensure that set operations, such as intersection, are correct. The notion of lexicographic normal form, based on any arbitrary but fixed order on symbols, gives such a canonical form.

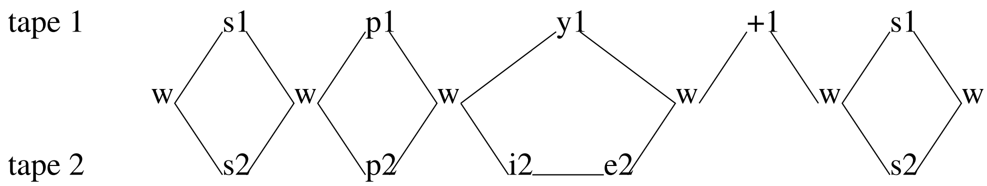

Figure 1: Partially ordered symbols

The compilation of the trace language into a finite-state automaton has been studied through the notion of recognizability. This automaton is very similar to an n-tape transducer. The Trace Language theory gives properties such as closure under intersection and soundness of the lexicographic normal form, which do not hold for usual transducers classes. It also provides a criterion to restrict the description of languages through regular expressions. This restriction is that the closure operator (Kleene star) must occur on connected languages only. In the translation of a partition-based regular expression, a star may appear either on a string of symbols of a given tape or on a string with at least one occurrence of ω.

Another benefit of Mazurkiewicz trace languages with respect to partitioned relations is their ability to represent the segmentation of the same form using two different partitionings. The example of figure 2 uses two partitionings of the form `spy+s`, one based on the notion of morpheme, the other on the notion of phoneme. The notation `<pos=noun>` and `<number=pl>` stands for two single symbols. Flat feature structures over (small) finite domains are easily represented by a string of such symbols. N-tuples are not very convenient to represent such a system.

Partition-based formalism are especially adapted to express relations between different representation such as feature structures and affixes, with respect to two-level morphology which imposes an artificial symbol-to-symbol mapping.

A multi-partitioned relation may be obtained by merging the translation of two partition-based grammars which share one or more common tapes. Such a merging is performed by the join operator of the relational algebra. Using a partition-based grammar for recognition or generation implies such an operation: the grammar is joined with a 1-tape machine without partitioning representing the form to be recognized (surface level) or generated (lexical level).

4 Multi-Tape Trace Languages

In this section, we define a subclass of Mazurkiewicz Trace Languages especially adapted to partition-based morphology, thanks to an explicit notion of tape partially synchronized by partition boundaries.

Definition 2 *A multi-tape partially commutative monoid is defined by a tuple* $(\Sigma, \Theta, \Omega, \mu)$ *where*

- Σ *is a finite set of symbols called the alphabet.*
- Θ *is a finite set of symbols called the tapes.*
- Ω *is a finite set of symbols which do not belong to* Σ*, called the partition boundaries.*
- μ *is a mapping from* $\Sigma \cup \Omega$ *to* 2^{θ} *such that* $\mu(x)$ *is a singleton for any* $x \in \Sigma$.

It is the Partially Commutative Monoid $M(\Sigma \cup \Omega, I_\mu)$ *where the independence relation is defined by* $I_\mu = \{(x,y) \in \Sigma \cup \Omega \times \Sigma \cup \Omega | \mu(x) \cap \mu(y) = \emptyset\}$. *Notation:* $MPM(\Sigma, \Theta, \Omega, \mu)$.

A Multi-Tape Trace Language is a subset of a Multi-Tape partially commutative monoid.

We now address the problem of relational operations over Recognizable Multi-Tape Trace Languages. Recognizable languages may be implemented by finite-state automata in lexicographic normal form, using the morphism φ^{-1}. Operations on trace languages are implemented by operations on finite-state automata. We are looking for implementations preserving the normal form property, because changing the order in regular languages is not a standard operation.

Some set operations are very simple to implement, namely union, intersection and difference.

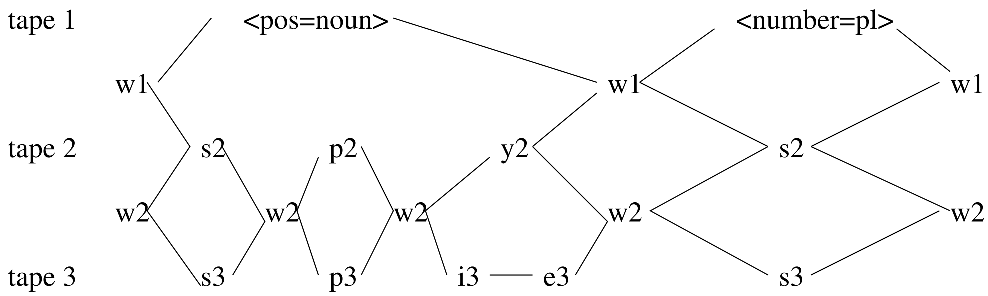

Figure 2: Two partitions of the same tape

The elements of the result of such an operation belongs to one or both operands, and are therefore in lexicographic normal form. If we write $Min(T)$ the set $Min(T) = \{x \in LexNF | [x] \in T\}$, where T is a Multi-Tape Trace Language, we have trivially the properties:

- $Min(T_1 \cup T_2) = Min(T_1) \cup Min(T_2)$
- $Min(T_1 \cap T_2) = Min(T_1) \cap Min(T_2)$
- $Min(T_1 - T_2) = Min(T_1) - Min(T_2)$

Implementing the complementation is not so straightforward because $Min(\overline{T})$ is usually not equal to $\overline{Min(T)}$. The later set contains strings not in lexical normal forms which may belong to the equivalence class of a member of T with respect to $\sim_I$. The complementation must not be computed with respect to regular languages but to LexNF.
$Min(\overline{T}) = LexNF - Min(T)$

As already mentioned, the concatenation of two regular languages in lexicographic normal form is not necessarily in normal form. We do not have a general solution to the problem but two partial solutions. Firstly, it is easy to test whether the result is actually in normal form or not. Secondly, the result is in normal form whenever a synchronization point belonging to all the levels is inserted between the strings of the two languages. Let $\omega_u \in \Omega, \mu(\omega_u) = \Theta$. Then, $Min(T_1.\{\omega_u\}.T_2) = Min(T_1).Min(\omega_u).Min(T_2)$.

The closure (Kleene star) operation creates a new iterative factor and therefore, the result may be a non recognizable trace language. Here again, concatenating a global synchronization point at the end of the language gives a trace language closed under Kleene star. By definition, such a language is connected. Furthermore, the result is in normal form.

So far, operations have operands and the result belonging to the same Multi-tape Monoid. It is not the case of the last two operations: projection and join.

We use the the operators Dom, Range, and the relations Id and Insert as defined in (Kaplan and Kay, 1994):

- Dom$(R) = \{x | \exists y, (x, y) \in R\}$
- Range$(R) = \{y | \exists x, (x, y) \in R\}$
- Id$(L) = \{(x, x) | x \in L\}$
- Insert$(S) = (\text{Id}(\Sigma) \cup (\{\epsilon\} \times S))^*$. It is used to insert freely symbols from S in a string from Σ^*. Conversely, Insert$(S)^{-1}$ removes all the occurrences of symbols from S, if $S \cap \Sigma = \emptyset$.

The result of a projection operation may not be recognizable if it deletes symbols making iterative factors connected. Furthermore, when the result is recognizable, the projection on $Min(T)$ is not necessarily in normal form. Both phenomena come from the deletion of synchronization points. Therefore, a projection which deletes only symbols from Σ is safe. The deletion of synchronization points is also possible whenever they do not synchronize anything more in the result of the projection because all but possibly one of its tapes have been deleted.

In the tape-oriented computation system, we are mainly interested in the projection which deletes some tapes and possibly some related synchronization points.

Property 2 *Projection*
Let T be a trace language over the MTM $M = (\Sigma, \Theta, w, \mu)$. Let $\Omega_1 \subset \Omega$ and $\Theta_1 \subset \Theta$. If

$\forall \omega \in \Omega - \Omega_1, |\mu(\omega) \cap \Theta_1| \leq 1,$ *then* $Min(\pi_{\Theta_1,\Omega_1}(T)) = Range(Insert(\{x \in \Sigma | \mu(x) \notin \Theta_1\} \cup \Omega - \Omega_1)^{-1} \circ Min(T))$

The join operation is named by analogy with the operator of the relational algebra. It has been defined on finite-state transducers (Kempe et al., 2004).

Definition 3 *Multi-tape join*
Let $T_1 \subset MTM(\Sigma_1, \Theta_1, \Omega_1, \mu_1)$ *and* $T_2 \subset TM(\Sigma_2, \Theta_2, \Omega_2, \mu_2)$ *be two multi-tape trace languages.* $T_1 \bowtie T_2$ *is defined if and only if*

- $\forall \sigma \in \Sigma_1 \cap \Sigma_2, \mu_1(\sigma) \cap \Theta_2 = \mu_2(\sigma) \cap \Theta_1$
- $\forall \omega \in \Omega_1 \cap \Omega_2, \mu_1(\omega) \cap \Theta_2 = \mu_2(\omega) \cap \Theta_1$

The Multi-tape Trace Language $T_1 \bowtie T_2$ *is defined on the Multi-tape Partially Commutative Monoid* $MTM(\Sigma_1 \cup \Sigma_2, \Theta_1 \cup \Theta_2, \Omega_1 \cup \Omega_2, \mu)$ *where* $\mu(x) = \mu_1(x) \cup \mu_2(x)$*. It is defined by* $\pi_{\Sigma_1 \cup \Theta_1 \cup \Omega_1}(T_1 \bowtie T_2) = T_1$ *and* $\pi_{\Sigma_2 \cup \Theta_2 \cup \Omega_2}(T_1 \bowtie T_2) = T_2$*.*

If the two operands T_1 and T_2 belong to the same MTM, then $T_1 \bowtie T_2 = T_1 \cap T_2$. If the operands belong to disjoint monoids (which do not share any symbol), then the join is a Cartesian product.

The implementation of the join relies on the finite-state intersection algorithm. This algorithm works whenever the common symbols of the two languages appear in the same order in the two operands. The normal form does not ensure this property, because symbols in the common part of the join may be synchronized by tapes not in the common part, by transitivity, like in the example of the figure 3. In this example, `c` on tape 3 and `f` on tape 1 are ordered $c < f$ by transitivity using tape 2.

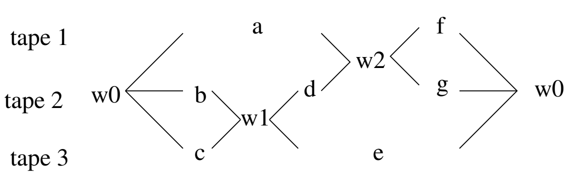

Figure 3: indirect tape synchronization

Let $T \subseteq MPM(\Sigma, \Theta, \Omega, \mu)$ a multi-partition trace language. Let G_T be the labeled graph where the nodes are the tape symbols from Θ and the edges are the set $\{(x, \omega, y) \in \Theta \times \Omega \times \Theta | x \in \mu(\omega)$ and $y \in \mu(\omega)\}$. Let Sync(Θ) be the set defined by Sync(Θ) $= \{\omega \in \Omega | \omega$ appears in G_T on a path between two tapes of $\Theta\}$.

The G_T graph for example of the figure 3 is given in figure 4 and Sync($\{1,3\}$) $= \{\omega_0, \omega_1, \omega_2\}$.

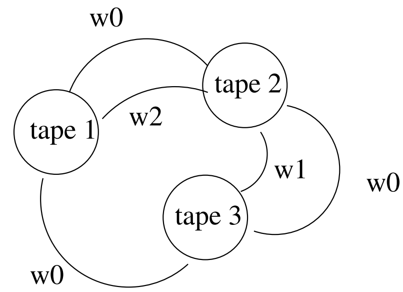

Figure 4: the G_T graph

Sync(Θ) is different from $\mu^{-1}(\Theta) \cap \Omega$ because some synchronization points may induce an order between two tapes by transitivity, using other tapes.

Property 3 *Let* $T_1 \subseteq MPM(\Sigma_1, \Theta_1, \Omega_1, \mu_1)$ *and* $T_2 \subseteq MPM(\Sigma_2, \Theta_2, \Omega_2, \mu_2)$ *be two multi-partition trace languages. Let* $\Sigma = \Sigma_1 \cap \Sigma_2$ *and* $\Omega = \Omega_1 \cap \Omega_2$*. If Sync*$(\Theta_1 \cap \Theta_2) \subseteq \Omega$*, then* $\pi_{\Sigma \cup \Omega}(Min(T_1)) \cap \pi_{\Sigma \cup \Omega}(Min(T_2)) = Min(\pi_{\Sigma \cup \Omega}(T_1) \cap \pi_{\Sigma \cup \Omega}(T_2)$

This property expresses the fact that symbols belonging to both languages appear in the same order in lexicographic normal forms whenever all the direct and indirect synchronization symbols belong to the two languages too.

Property 4 *Let* $T_1 \subseteq MPM(\Sigma_1, \Theta_1, \Omega_1, \mu_1)$ *and* $T_2 \subseteq MPM(\Sigma_2, \Theta_2, \Omega_2, \mu_2)$ *be two multi-partition trace languages. If* $\Theta_1 \cap \Theta_2$ *is a singleton* $\{\theta\}$ *and if* $\forall \omega \in \Omega_1 \cap \Omega_2, \theta \in \mu(\omega)$*, then* $\pi_{\Sigma \cup \Omega}(Min(T_1)) \cap \pi_{\Sigma \cup \Omega}(Min(T_2)) = Min(\pi_{\Sigma \cup \Omega}(T_1) \cap \pi_{\Sigma \cup \Omega}(T_2)$

This second property expresses the fact that symbols appear necessarily in the same order in the two operands if the intersection of the two languages is restricted to symbols of a single tape. This property is straightforward since symbols of a given tape are mutually dependent.

We now define a computation over $(\Sigma \cup \Omega)^*$ which computes $Min(T_1 \bowtie T_2)$.

Let $T_1 \subset MTM(\Sigma_1, \Theta_1, \omega_1, \mu_1)$ and $T_2 \subset MTM(\Sigma_2, \Theta_2, \Omega_2, \mu_2)$ be two recognizable multi-tape trace languages.
If Sync($\Theta_1 \cap \Theta_2$) $\subseteq \Omega$, then $Min(T_1 \bowtie T_2) =$ Range($Min(T_1 \circ$ Insert($\Sigma_2 - \Sigma_1$) $\circ$ Id(LexNF)) $\cap$ Range($Min(T_2) \circ$ Insert($\Sigma_1 - \Sigma_2$) $\circ$ Id(LexNF)).

5 A short example

We have written a morphological description of Turkish verbal morphology using two different partitionings. The first one corresponds to the notion of affix (morpheme). It is used to describe the morphotactics of the language using rules such as the following context-restriction rule:

$\texttt{(y}^?\texttt{I}^4\texttt{m,1 sing)} \Rightarrow$
$\quad \texttt{(I}^?\texttt{yor,prog)}|\texttt{(y}^?\texttt{E}^2\texttt{cE}^2\texttt{k,future)}$ __

In this rule, $y^?$ stands for an optional y, I^4 and E^2 for abstract vowels which realizations are subject to vowel harmony and $I^?$ is an optional occurrence of the first vowel. The rule may be read: the suffix $y^?I^4m$ denoting a first person singular may appear only after the suffix of progressive or the suffix of future[1]. Such rules describe simply affix order in verbal forms.

The second partitioning is a symbol-to-symbol correspondence similar to the one used in standard two-level morphology. This partitioning is more convenient to express the constraints of vowel harmony which occurs anywhere in the affixes and does not depend on affix boundaries.

Here are two of the rules implementing vowel harmony:

$\texttt{(I}^4\texttt{,i)} \Rightarrow \texttt{(Vow,e|i) (Cons,Cons)*}$ __
$\texttt{(I}^4\texttt{,u)} \Rightarrow \texttt{(Vow,o|u) (Cons,Cons)*}$ __

Vow and Cons denote respectively the sets of vowels and consonants. These rules may be read: *a symbol I^4 is realized as i (resp. u) whenever the closest preceding vowel is realized as e or i (resp. o or u).*

The realization or not of an optional letter may be expressed using one or the other partitioning. These optional letters always appear in the first position of an affix and depends only on the last letter of the preceding affix.

$\texttt{(y}^?\texttt{,y)} \Rightarrow \texttt{(Vow,Vow)}$ __

Here is an example of a verbal form given as a 3-tape relation partitioned using the two partitionings.

verbal root			prog				1 sing		
g	e	l	$I^?$	y	o	r	$Y^?$	I^4	m
g	e	l	i	y	o	r	ϵ	u	m

The translation of each rule into a Multi-tape Trace Language involves two tasks: introducing partition boundary symbols at each frontier between partitions. A different symbol is used for each kind of partitioning. Distinguishing symbols from different tapes in order to ensure that $\mu(x)$ is a singleton for each $x \in \Sigma$. Symbols of Σ are therefore pairs with the symbol appearing in the rule as first component and the tape identifier, a number, as second component.

Any complete order between symbols would define a lexicographic normal form. The order used by our system orders symbol with respect to tapes: symbols of the first tape are smaller than the symbols of tape 2, and so on. The order between symbols of a same tape is not important because these symbols are mutually dependent. The translation of a tuple $(a_1 \dots a_n, b_1 \dots b_m)$ is $(a_1, 1) \dots (a_n, 1)(b_1, 2) \dots (b_m, 2)\omega_1$. Such a string is in lexicographic normal form. Furthermore, this expression is connected, thanks to the partition boundary which synchronizes all the tapes, so its closure is recognizable. The concatenation too is safe.

All contextual rules are compiled following the algorithm in (Yli-Jyrä and Koskenniemi, 2004) [2]. Then all the rules describing affixes are intersected in an automaton, and all the rules describing surface transformation are intersected in another automaton. Then a join is performed to obtain the final machine. This join is possible because the intersection of the two languages consists in one tape (cf. property 4). Using it either for recognition or generation is also done by a join, possibly followed by a projection.

For instance, to recognize a surface form $geliyorum$, first compile it in the multi-tape trace language $(g, 3)(e, 3)(l, 3) \dots (m, 3)$, join it with the morphological description, and then project the result on tape 1 to obtain an abstract form `(verbal root,1)(prog,1)(1 sing,1)`. Finally extract the first component of each pair.

6 Conclusion

Partition-oriented rules are a convenient way to describe some of the constraints involved in the morphology of the language, but not all the constraints refer to the same partition notion. Describing a rule

[1]The actual rule has 5 other alternative tenses. It has been shortened for clarity.

[2]Two other compilation algorithm also work on the rules of this example (Kaplan and Kay, 1994), (Grimley-Evans et al., 1996). (Yli-Jyrä and Koskenniemi, 2004) is more general.

with an irrelevant one is sometimes difficult and inelegant. For instance, describing vowel harmony using a partitioning based on morphemes takes necessarily several rules corresponding to the cases where the harmony is within a morpheme or across several morphemes.

Previous partition-based formalisms use a unique partitioning which is used in all the contextual rules. Our proposition is to use several partitionings in order to express constraints with the proper granularity. Typically, these partitionings correspond to the notions of morphemes, phonemes and graphemes.

Partition-based grammars have the same theoretical power as two-level morphology, which is the power of regular languages. It was designed to remain finite-state and closed under intersection. It is compiled in finite-state automata which are formally equivalent to the epsilon-free letter transducers used by two-level morphology. It is simply more easy to use in some cases, just like two-level rules are more convenient than simple regular expressions for some applications.

Partition-Based morphology is convenient whenever the different levels use very different representations, like feature structures and strings, or different writing systems (e.g. Japanese hiragana and transcription). Two-level rules on the other hand are convenient whenever the related strings are variants of the same representation like in the example (spy+s,spies). Note that multi-partition morphology may use a one-to-one correspondence as one of its partitionings, and therefore is compatible with usual two-level morphology.

With respect to rewrite rule systems, partition-based morphology gives better support to parallel rule application and context definition may involve several levels. The counterpart is a risk of conflicts between contextual rules.

Acknowledgement

We would like to thank an anonymous referee of this paper for his/her helpful comments.

References

François Barthélemy. 2005. Partitioning multitape transducers. In *International Workshop on Finite State Methods in Natural Language Processing (FSMNLP)*, Helsinki, Finland.

François Barthélemy. 2006. Un analyseur morphologique utilisant la jointure. In *Traitement Automatique de la Langue Naturelle (TALN)*, Leuven, Belgium.

Alan Black, Graeme Ritchie, Steve Pulman, and Graham Russell. 1987. Formalisms for morphographemic description. In *Proceedings of the third conference on European chapter of the Association for Computational Linguistics (EACL)*, pages 11–18.

Volker Diekert and Yves Métivier. 1997. Partial commutation and traces. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. 3*, pages 457–534. Springer-Verlag, Berlin.

Edmund Grimley-Evans, George Kiraz, and Stephen Pulman. 1996. Compiling a partition-based two-level formalism. In *COLING*, pages 454–459, Copenhagen, Denmark.

Nizar Habash, Owen Rambow, and George Kiraz. 2005. Morphological analysis and generation for arabic dialects. In *Proceedings of the ACL Workshop on Semitic Languages*, Ann Harbour, Michigan.

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20:3:331–378.

André Kempe, Jean-Marc Champarnaud, and Jason Eisner. 2004. A note on join and auto-intersection of *n*-ary rational relations. In B. Watson and L. Cleophas, editors, *Proc. Eindhoven FASTAR Days*, Eindhoven, Netherlands.

George Anton Kiraz. 2000. Multitiered nonlinear morphology using multitape finite automata: a case study on syriac and arabic. *Comput. Linguist.*, 26(1):77–105.

Kimmo Koskenniemi. 1983. Two-level model for morphological analysis. In *IJCAI-83*, pages 683–685, Karlsruhe, Germany.

Stephen G. Pulman and Mark R. Hepple. 1993. A feature-based formalism for two-level phonology. *Computer Speech and Language*, 7:333–358.

Anssi Yli-Jyrä and Kimmo Koskenniemi. 2004. Compiling contextual restrictions on strings into finite-state automata. In B. Watson and L. Cleophas, editors, *Proc. Eindhoven FASTAR Days*, Eindhoven, Netherlands.