TR-IT-0049

Texts and Structures Pattern-matching and Distances

Yves Lepage

March 1994

Abstract

Basic data structures of natural language processing, strings and trees, are generalised in a data structure called *wood*. On this data structure, pattern-matching is formalised as *identification*. A neutral element is built, which enables a formal interpretation of variables in patterns. Also, a *distance* on this data structure can be defined as an extension and a generalisation of well-known metrics on strings and trees. The links between pattern-matching and distance are presented as well as results in formal language theory.

Keywords

String, tree, wood, matching, identification, distance, metric.

Contents

List of Figures

Introduction

This report gives a theoretical justification of the work we have done in ATR from the 1st of November 1991 to the 31st of March 1994.

Our research has concentrated on the theoretical links between a pattern-matching operation and distances on one hand and on the implementation of the theoretical results on the other hand. This report is dedicated to the first aspect. As for implementation and results, the reader may refer to other technical reports and articles.

- [Lepage 92b] for a range of basic objects and generic functions which we used in our various programs;
- [Lepage 92a] and [Lepage 92c] for special useful C functions and macros (grammar writing and dynamic programming);
- [Lepage et al. 92] on the establishment of a relation between distances and pattern-matching;
- [Lepage 93a] for a report on our first experiments with a non-directional system making use of genetic algorithms. [Lepage 93c] and [Lepage 93b] are conference articles on the same topic;
- [Lepage and Fais 94a] for a description of the linguistic structures (drawn by hand) we used in the experiments reported in the previous articles;

1 Texts and structures

A sketch of nowadays machine translation landscape would show two main tendencies: the use of symbolic calculation on structures and that of distance on strings. Our purpose is to pave the way for a possible mixing of these approaches.

They could be reconciled if the relation between some matching operation and some distance could be clearly set. Our purpose is to show that *identification* is linked with the *Wagner and Fischer distance* on strings and the *Selkow distance* on trees.

This demonstration is facilitated by the definition of a general data structure, *woods*. It has been introduced to unify the data structures contained in the *board*, a non-directional object based on an analogy with a basic linguistic notion: the *signe*.

1.1 Structures from rule-based systems

Until now, it seems that NLP people have been concentrating on the structures that parsers are supposed to deliver. Their usual shape is that of a labeled or decorated tree, a logical formula, or a feature structure, *i.e.* a direct acyclic graph, often encoding a syntactic CF-skeleton, a logical formula and other attributes. The goal of an analysis is to produce such a structure. Actual parsers are based on the fundamental idea that parsing is the task of building a structure for a given input text.

Real-size machine translation systems use pattern-matching on trees and tree-transduction to produce such structures. For instance Q-systems [Colmerauer 70] or GramR [Chandioux et Guérard 81] for the TAUM-METEO system, or ROBRA [Boitet 82] for the GETA system, where complex tree patterns are described with the help of variables.

Prototype systems, based on NLP formalisms, reflect the concern about the constructing operation of the output structures. In those prototypes, unification or adjunction, for instance, are the central operations which dictate the complexity and the rapidity of the system.

In all those systems, which follow a *rule-based* approach, the computation is essentially symbolic and no approximation has its place.

1.2 Texts in example-based systems

There has been a reemergence of the corpus-based approach to machine translation in the recent years [Nagao 84]. Its most illustrative realisations fall under the name of *example-based* machine translation. The basic idea is to adapt already translated texts (the corpus of examples) to new inputs [Sato 90]. The closer the input to the examples, the less adaptation to the stored translation is required.

As a matter of fact, the computation of distances allows one to choose between several candidates in problems which are not yet well formalised linguistically ([Furuse and Iida 92a], [Sumita and Iida 92]). Tree-editing functions may also be used to adapt patterns to the data at hand in a minimal sense, which implies some distance computation [Sadler and Vendelmans 90], [Sato and Nagao 90].

In this approach, the *example-based* one, the translation process relies on the computation of a numerical function: a distance between the input text and stored texts.

1.3 Blending

Usually, the output of a parser is a structure, not the relationship between the structure output and the input string. In no actual system is this relationship an explicit result of the analysis. We would need to see directly which part of the string corresponds to which part in the structure.

Reciprocally, whereas it is relatively easy to define a distance between words, taking account of this distance in structures built by a parser has only been defined in *ad hoc* ways. We need the definition of a general distance applying on string patterns as well as on tree patterns.

Our proposal is a sort of *blending*. It aims at defining a general data structure subsuming strings, trees, string patterns and tree patterns. On this data structure, pattern-matching and distance are defined so that these two operations have some relationship.

2 Boards

2.1 A structuralist notion: the "signe"

The founder of modern linguistics, Ferdinand de Saussure, defined the task of linguistics as the description of what he called *signe*: the association of two things (the *signifiant* or acoustical image, and the *signifié* or concept) [Saussure 15]. He insisted on the fact that both the *signifiant* and the *signifié* have their own *structure*[1].

If we draw a parallel between the *signe* and the objects NLP researchers work with, why not consider the texts as *signifiant* and the linguistic representations as *signifié*? Of course, it is just an analogy, and we by no means pretend that we show how to implement a *signe*.

2.2 A proposal: the board

Our proposal is to view the pair constituted by a text and its linguistic structure as the basic object in a grammar. Such a pair we call a *board*. The types of the two objects in this pair are commonly used: a text is a string and a structure is a tree.

In current grammars, a grammar "element" is either a chunk of structure ($S \rightarrow NP\ VP$ or $det \rightarrow the$ are the same as *S(NP,VP)* and *det(the)* respectively) or a string pattern (in grammars inspired by Harris). In our proposal, both aspects are present. In this way, the structural aspect of linguistic works can be mixed with examples. Figure 1 shows a sentence and its associated representation in a board.

Figure 2 shows a general pattern. In this board, variables stand for substrings in the string and forests in the tree part.

2.3 The association in the "signe"

The *signe* is not only the two sides of the coins, it is also their *association*. The master of Geneva insisted that this association is arbitrary, which does not mean that it is left to the laws of fate, but that there

[1]In a structure, in the structuralist meaning of the word, elements have no absolute existence. They exist thanks to their relationships or *oppositions* to other elements. The concept of structure gives rise to the distinction between phonology (the study of functional or opposition features of the sounds of a given language) and phonetics (the study of the articulatory or perceptive aspects of the sounds of one or many languages).

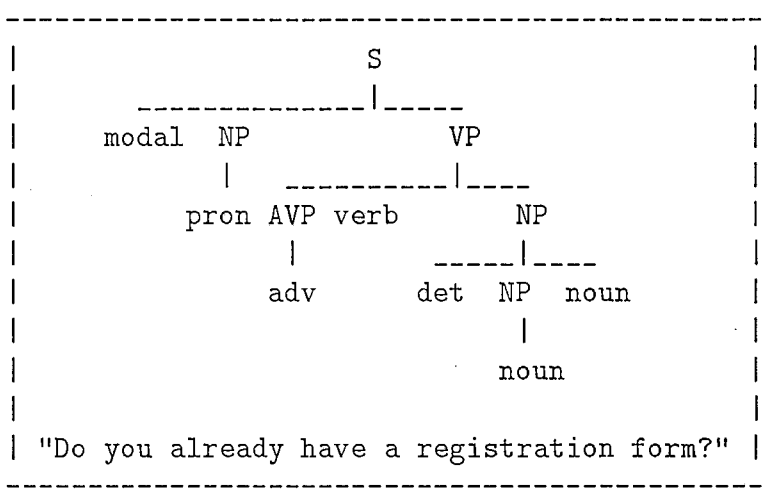


Figure 1: A board: text and associated representation

is no necessary connection between some *signifiant* and some *signifié*. If we carry on our daring analogy with the *signe*, and have a look at the different representations NLP people give to the same sentence (constituent structure, dependency structure, logical form, *etc.*), then a board has to account for various kinds of associations between a text chunk and a structure.

2.4 A proposal: correspondences

When projectivity is assumed, this association is implicit. But, projectivity is an English-oriented hypothesis which has never been considered valid for the description of, for instance, slavic languages in general[2]. Fashion linguistic theories reject the problems posed by this too strong assumption in a "linear-precedence module", which no actual implementation has ever been able to realise. As a consequence, trying to get the fine relationship between a text and its representation output by a parser almost always means tracing the parsing process.

In our proposal, the relationship between the text and its structural representation is called *correspondences* [Boitet and Zaharin 88]. It is explicit and made out of two kinds of links:

[2]As a matter of fact, I doubt any Slav linguist could have ever imagined such a restrictive hypothesis!

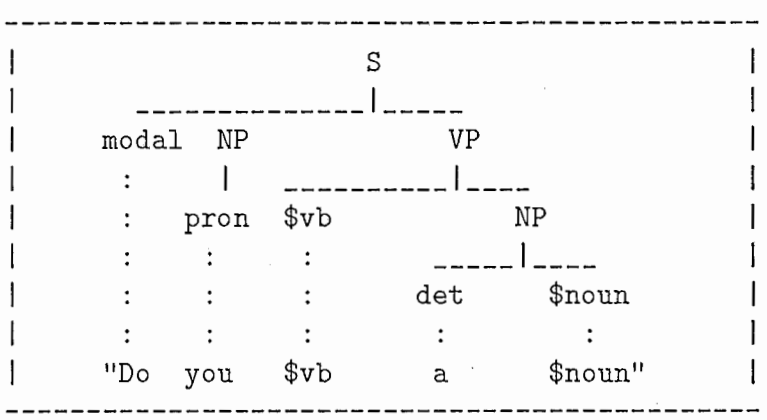


Figure 2: A board with variables

- between (possibly a list of) words and nodes;
- between (possibly non-connex) substrings and complete subtrees (designated by their root).

This is illustrated in Figure 3. On each node, two intervals stand for the word-node and the substring-subtree correspondences.

We have shown [Lepage 89] that only three constraints may govern correspondences in order to be able to describe standard linguistic structural representations.

- **global correspondence**: the entire tree in a board corresponds to the entire string in that board;
- **inclusion**: if a subtree is included in another subtree, then the substring in correspondence with the included subtree is included in the substring in correspondence with the including subtree;
- **membership** if a node is member of a subtree, then the words in correspondence with the node are in the substring in correspondence with the subtree.

From the language theoretical point of view, [Lepage 91] shows how a simple grammar of four boards can analyse and generate the context-sensitive language $a^n b^n c^n$ with perspicuous and natural structures (grouping all a's, all b's, all c's together, which no standard formalism is able to do).

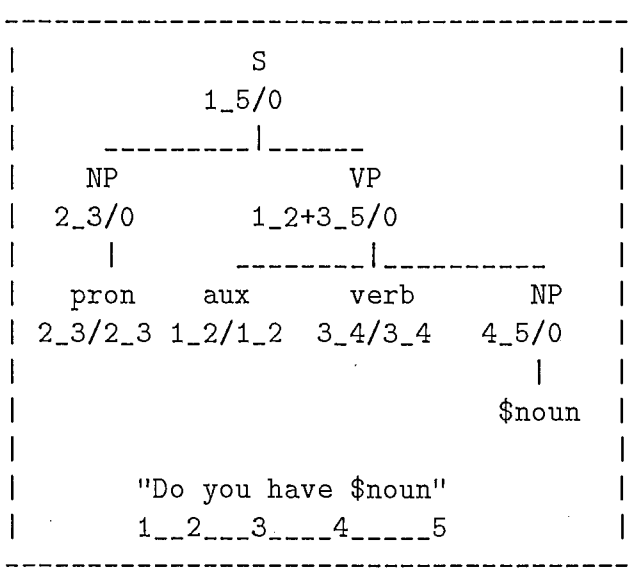


Figure 3: Correspondences in a board

From the linguistic point of view, [Zaharin and Lepage 92] describes how crossed dependencies (non-projectivity and discontinuity) are economically handled in the analysis of the famous Dutch subordinate clause *dat Jan Marie de hond de krant probeert to leren laten halen* ("that John tries to teach Mary to make the dog fetch the newspaper").

3 Matching

3.1 Identification

Identification is a formalisation of pattern-matching on a data structure which generalises strings and trees. It was first introduced in [Zaharin 86] and then completely defined in [Lepage 89]. Appendix B gives a formalisation and [Lepage & Zaharin 91] some linguistic justifications.

3.1.1 Strings

On strings, identification is a standard pattern-matching.

$$\mathrm{idn}(his.uncle.'s.friend, his.uncle.'s.\$1) = his.uncle.'s.friend$$

with $\$1 = \{friend\}$.

As a particular case, a variable may instantiate with an empty string.

$$\mathrm{idn}(his.uncle.'s.son, his.uncle.\$1.'s.son) = his.uncle.'s.son$$

with $\$1 = \{\varepsilon\}$.

In some cases, Colmerauer's unification fails because variables, for that operation, only instantiate with string elements. But identification does not fail because here variables stand for substrings of any length.

$$\mathrm{idn}(his.uncle.'s.friend.'s.son, his.\$1.'s.\$2) = his.uncle.'s.friend.'s.son$$

with $\$1 = \{uncle, uncle.'s.friend\}$ and $\$2 = \{friend.'s.son, son\}$.

As a consequence, identification would be non-deterministic if considered strictly on the string data structure.

3.1.2 Trees

On trees, variables may instantiate with possibly empty forests, as illustrated in Figure 4.

3.1.3 Woods

The data structure of *wood* generalises strings and trees. Woods may be seen as forests with a depth dimension. So, woods have three dimensions.

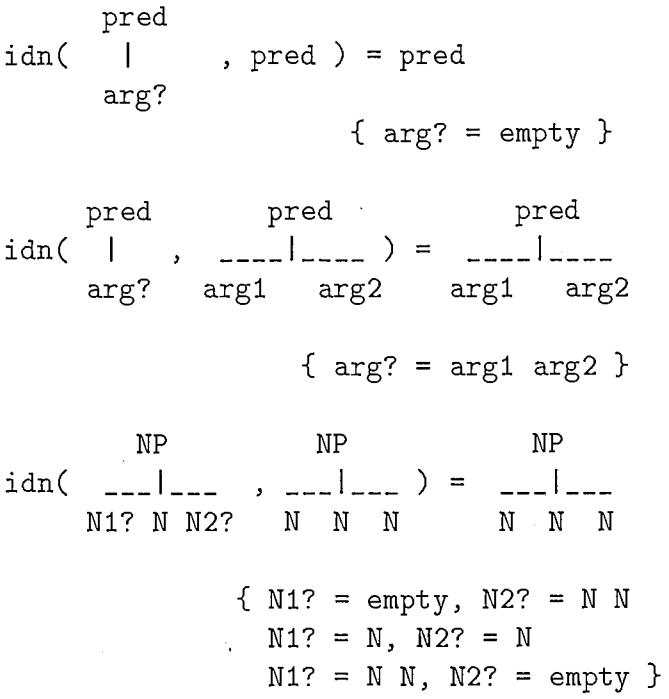


Figure 4: Examples of identification

- The first one is that of string concatenation;
- the second one is that of dominance in trees;
- the third one is the way by which we capture different solutions during pattern matching.

A string appears to be a forest with no second nor third dimension. It is a "flat" and "thin" forest. Trees appear to be "thin" forests with only one element on the higher level. Hence, as announced, strings and trees are particular cases of woods.

Thanks to their third dimension, woods can capture ambiguities as illustrated in Figure 5 for the sentence *He writes a manual for the beginner* which may have two interpretations depending on the PP attachment of *for the beginner*.

As a consequence, woods can encapsulate the non-determinism of pattern-matching and identification becomes a deterministic operation on woods.

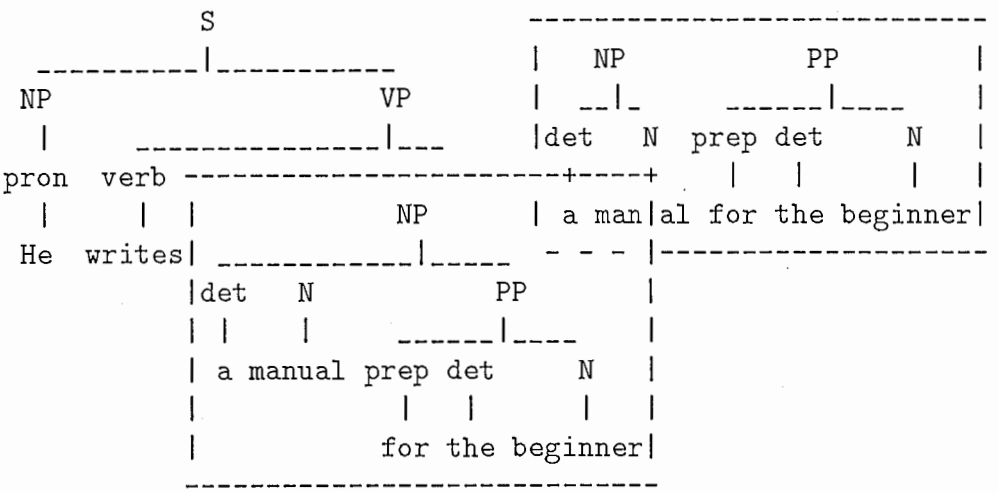


Figure 5: Representation of ambiguities

Our main motivation for the wood data structure was to give a tentative answer to a tickling problem in NLP [Boitet 91]: how can we do linguistic programming on ambiguous structures without the burden of explicitly handling such structures, in the same way as people do recursive programming in modern programming languages without the burden of programming the handling of a stack for the variable context?

3.2 A possible engine

Figure 6 gives a possible algorithm for a system using grammars composed of boards. With this system, interpreting exactly the same set of boards, analysis and generation turn out to be only particular cases of a more general case, depending on the type of input only. The general task of the system is non-direction completion (see 3.3.3).

The *explore* procedure tries to identify its input with each board in the grammar. The fundamental operation for the *explore* procedure is *identification*. Performing identification entails instantiation of variables, stored in a substitution (a set of variables, with their associated value). The *propagate* procedure realises the propagation of substitutions to those boards containing one (or more) of the variables present in the substitution. Those new boards are explored in turn. A significant difference with usual systems, is that variable names are not local to one board, but global to a grammar.

```
explore(BOARD *candidate)
{
   for ( each board != candidate )
      if ( idn(candidate,board) and
           substitution != empty )
         propagate(substitution) ;
}

propagate(SUBSTITUTION *substitution)
{
   for ( each board containing a variable
         present in substitution )
      substitution(board/substitution) ;
}
```

Figure 6: Simplified interpretation algorithm for a board grammar

The algorithm halts when no more identification is possible. The system answers by a failure if no identification has been performed and a success otherwise. In this latter case, the output is the input board with variables replaced by their values. Several solutions are possible due to the non-determinism of identification.

3.3 Declarativity and non-directionality

Our proposal is not only more declarative than fashion NLP formalisms, it also has an original property which is more interesting than bi-directionality.

3.3.1 Declarativity

An advantage of the previous proposal is that the presentation of a linguistic fact by just stating the correspondences between a string and a tree appears to be more declarative than many formalisms. For instance, HPSG can be criticised from the viewpoint of declarativity as follows.

If the output is the complete feature structure, then, the declarativity of the syntactic part is questionable, because special mother and daughter features encode the mother-daughter relation in the syntactic tree. A declarative formalism should allow the syntactic description to be drawn exactly as it is.

Now, if the output is only the semantic part of the feature structure, then the relationship between the text and its representation is obtained through the composition of two functions: the first one is a unification-augmented context-free parsing, and the second one is a Montague-style mapping between a syntactical tree and a logical formula. The composition of these two functions is not stated declaratively.

Boards allow one to write two separate grammars one for syntax, the other one for semantics. In HPSG-like formalisms, the relationship between the two structures is kept through the feature structure variables. With boards, it is the role of correspondences to ensure these links, and they go necessarily through the text.

3.3.2 Bi-directionality

The term *non-directionality* was used by [Winograd 83] to describe context-free grammars, because rules can be applied in both directions: analysis and generation. In fact, as [Zaharin 90] observed, the reverse operation of analysis would imply starting from the start symbol only. But, in actual NLP systems, it starts with a complete structure. So generation is not the exact reverse operation of analysis. Nowadays, the term *non-directionality* has been replaced by *bi-directionality*.

Many formalisms are said to be bi-directional, but often, some extensions or tricks in programming make grammars suitable only for analysis (or only for generation). In other systems, only the formalism (or the source code to adopt a programmer's point of view) is bidirectional, and a compilation delivers two different executable codes, one for analysis, and one for generation. This means that the engine (or the executable code, to follow our comparison) is not bidirectional.

We are in favour of a system where not only the formalism, but the engine too is bidirectional. Not only for aesthetical reasons, but because the advantage of a good formalism lies in two things: it allows reformulation from older formalisms and it has new interesting features. This is the case for our proposal.

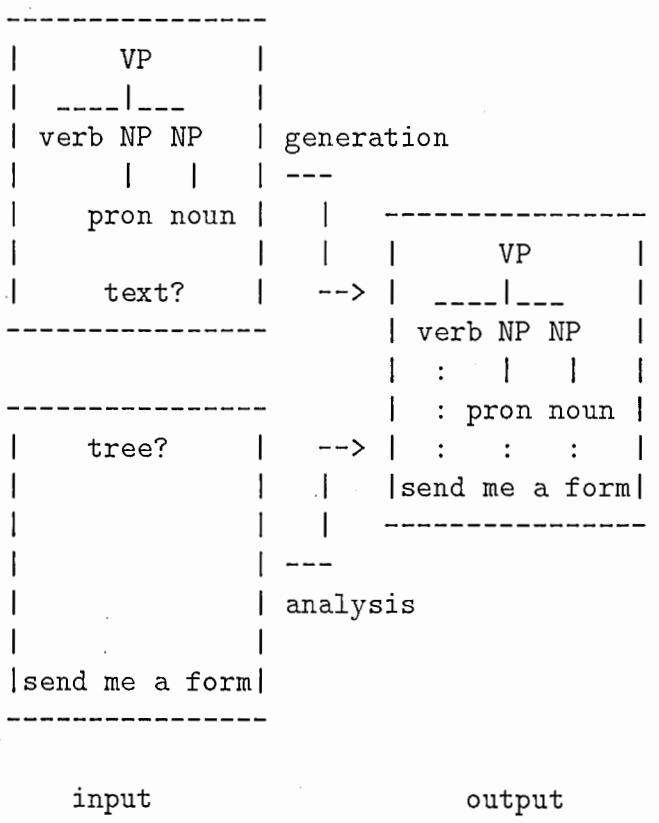


Figure 7: Analysis and generation with boards

3.3.3 Non-directionality

In our proposal, boards are not only the basic objects in the grammar, they are also input and output objects for analysis and generation as shown in Figure 7 where `text?` and `tree?` denote variables, the respective values of which have to be determined.

This view of analysis and generation gives birth to a more general operation which we call non-directional completion. Bi-directionality appears to be a particular case of what we call *non-directionality*, to revive the term [Lepage 91].

Non-directional completion consists in proposing an uncomplete string and an associated uncomplete tree to the system. The system has to deliver complete strings associated with complete trees which match the uncomplete input and which come from the grammar.

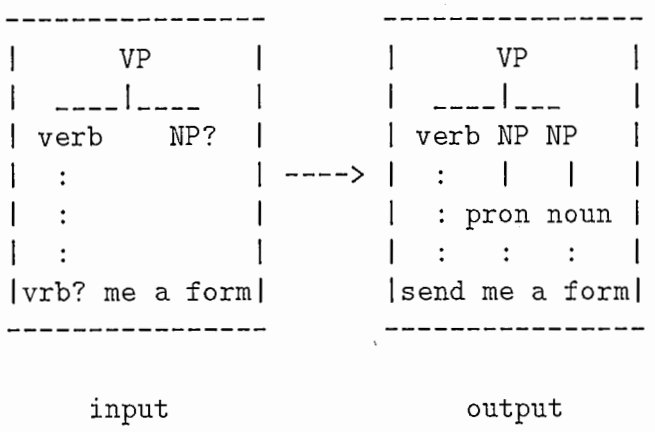


Figure 8: Non-direction completion

This is impossible with a compilation of the grammar delivering two different specialised modules, one for analysis and one for generation. This is possible for our proposal because the basic object is not only bi-directional but also non-directional. As a matter of fact, **the general function of a system is to deliver complete correspondences for partially specified correspondences.**

4 Distances

4.1 Metrics and distances

4.1.1 Strings

Consider the spell-checker of a word-processor. Words in a text are successively compared to words in a dictionary. If a word of the text is not present in the dictionary, the closest word in the dictionary must be proposed to the user as a candidate for correction. So, we suppose that the user must have performed some typing errors and, by indulgence, a minimal number of them. In this sense, the closest word in the dictionary is that word from which the string at hand can be obtained by performing on it the minimal number of typing errors.

Here is a list of possible typing errors illustrated on the word *uncle*.

- insertion: *uxncle*;
- deletion: *unle*;
- replacement: *unxle*.

Whereas [Wagner & Fischer 74] considered these three typing errors only, [Lowrance & Wagner 75] considered a fourth one.

- interchange of two adjacent characters: *ucnle*.

But they say they do not know of a general algorithm that can find a minimal cost trace in cases where the weight of interchange is less than half the sum of the weights of insertion and deletion. So the Lowrance and Wagner distance value is always greater than the Wagner and Fischer's one and is also always more time-consuming. This is why we consider the latter one, which is simpler, for our work.

The recursive definition of a string says that it is an element of the vocabulary (be they charaters or words, or even trees) followed by a string. Hence, the comparison between two strings can be performed by comparing the first elements together. If they are equal, then we carry on the comparison. If not, we suppose that it is a replacement typing error. So we have

$$\mathrm{dist}(a.u, b.v) = \mathrm{dist}(a, b) + \mathrm{dist}(u, v)$$

But this is not always the case, as the a in the first string may be the result of an insertion operation:

$$\text{dist}(a.u, b.v) = \text{dist}(a, \varepsilon) + \text{dist}(u, b.v)$$

or could have been deleted in the second string:

$$\text{dist}(a.u, b.v) = \text{dist}(\varepsilon, b) + \text{dist}(a.u, v)$$

The string-to-string problem being a minimisation problem, we combine the three previous formulae.

$$\text{dist}(a.u, b.v) = min(\begin{array}{l}\text{dist}(a, \varepsilon) + \text{dist}(u, b.v), \\ \text{dist}(a, b) + \text{dist}(u, v), \\ \text{dist}(\varepsilon, b) + \text{dist}(a.u, v)\end{array})$$

The distances between elements of the vocabulary and the empty string have to be defined. One can assign a different weight for each editing operation. For simplification, we may posit:

$$\begin{array}{lllll}\text{dist}(a, b) = & 0 & \text{if} & a = b \\ & 1 & \text{else} & \text{(replacement)} \\ \text{dist}(\varepsilon, b) = & 1 & & \text{(insertion)} \\ \text{dist}(a, \varepsilon) = & 1 & & \text{(deletion)}\end{array}$$

Wagner and Fischer have proved that the function defined in this way yields the minimum number of edit operations one has to perform in order to transform one string into another one.

In the general case, the three edit operations do not have to have the same weight for the Wagner and Fischer distance to be a mathematical metric. We have proved that, if the distances on the vocabulary (augmented with the empty string) is a metric, then the Wagner and Fischer distance is a metric on the set of strings built on that vocabulary (see Appendix C).

4.1.2 Trees

There exists a similar work on distances on trees. The simplest form of distance is the Selkow one [Selkow 77], the more complex one being Tai's one [Tai 77]. The Selkow distance accounts for insertion, deletion and replacement of nodes in trees. With that distance, the results given in Figure 9 are obtained.

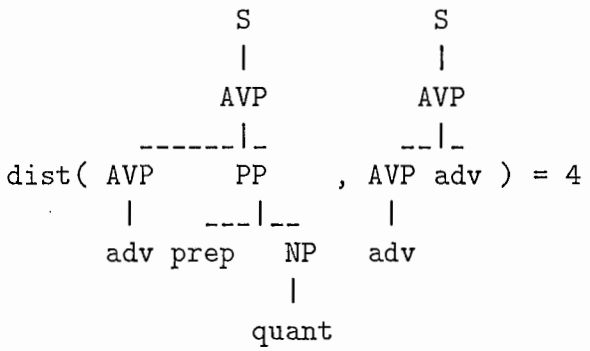


Figure 9: Selkow distance between two trees

4.1.3 Woods

On forests, a general function may be defined in a very similar way as the Wagner and Fischer distance on strings.

$$\mathrm{dist}(a(u').u, b(v').v) = min\left(\begin{array}{l}\mathrm{dist}(a(u'),\varepsilon) + \mathrm{dist}(u, b(v').v),\\ \mathrm{dist}(a(u'), b(v')) + \mathrm{dist}(u, v),\\ \mathrm{dist}(\varepsilon, b(v')) + \mathrm{dist}(a(u').u, v)\end{array}\right)$$

$$\begin{aligned}\mathrm{dist}(a(u'),\varepsilon) &= \mathrm{dist}(a,\varepsilon) + \mathrm{dist}(u',\varepsilon)\\ \mathrm{dist}(a(u'), b(v')) &= \mathrm{dist}(a,b) + \mathrm{dist}(u',v')\end{aligned}$$

$$\begin{aligned}\mathrm{dist}(a,b) &= 0 \text{ if } a = b\\ &\quad 1 \text{ else (replacement)}\\ \mathrm{dist}(\varepsilon,b) &= 1 \quad \text{(insertion)}\\ \mathrm{dist}(a,\varepsilon) &= 1 \quad \text{(deletion)}\end{aligned}$$

This distance appears to be a generalisation of the Wagner and Fischer distance on strings and the Selkow distance on trees, simply because strings are forests with no subtrees, and trees are forests with only one node on the higher level.

This distance can be extended to wood patterns, *i.e.* wood with variables. This extension is made possible by considering patterns as denumerable sets of constant objects. A distance may be extended in a natural way from elements to subsets by considering the minimum on

all possible distances between elements of the subsets. From the mathematical point of view, this extension is no more a metric (if the distance on elements was one). For instance, the first axiom does not hold in the general case and has to be replaced by the following proposition.

$$\text{dist}(A, B) = 0 \Leftrightarrow A \cap B \neq \emptyset$$

4.2 A possible engine

Genetic algorithms constitute a possible answer to build an NLP system using distances. They are a collection of techniques for approaching the solution of optimisation problems.

On the contrary to usual programming techniques which handle only one object at a time, genetic algorithms deal with a collection of individuals, called a *population*. For each individual, one can compute a function, called the *fitness function*. Those individuals for which the fitness function is optimum are the *best individuals*. From two individuals, one can derive two new individuals by cutting them into two pieces and gluing the pieces back. This is *crossover*.

In our system, individuals are boards, and the starting population a data base of example boards. The fitness of an element in a population (set of boards) is the distance to a given input (a board) to the system. We defined a distance between boards by taking the sum of the distances between the strings on the one hand, and the trees on the other hand.

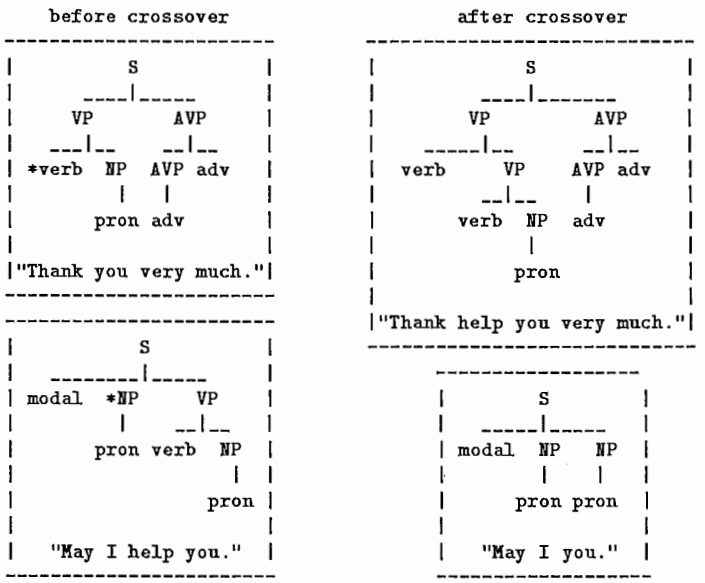


Figure 10: Crossover on projective boards

As for crossover, we insisted on keeping the unity of data structure between strings and trees. So, we translated the common string

crossover into forest terms: it is the exchange of the sister forests of the crossover points. This can be applied directly to trees. By keeping projectivity during crossover, only corresponding parts of strings and trees are exchanged. As a consequence, string crossover allows exchange of inner substrings. Hence, a board obtained by crossover gives a partially valid description of a possibly ungrammatical sentence (see Figure 10).

The system built implements a simple genetic algorithm. The starting population is a set of example boards, *i.e.* complete sentences with their complete associated linguistic structures.

If an input board is given to the system, each board in the data base of examples is assigned a fitness score: its distance to the input board. The output is a board, built from pieces of the data base boards which minimises the distance to the input. It is important to stress the point that the input never enters the data base of boards. It is only used to compute the fitness of each board in the data base in each generation. Figure 11 summarises the system and its functioning.

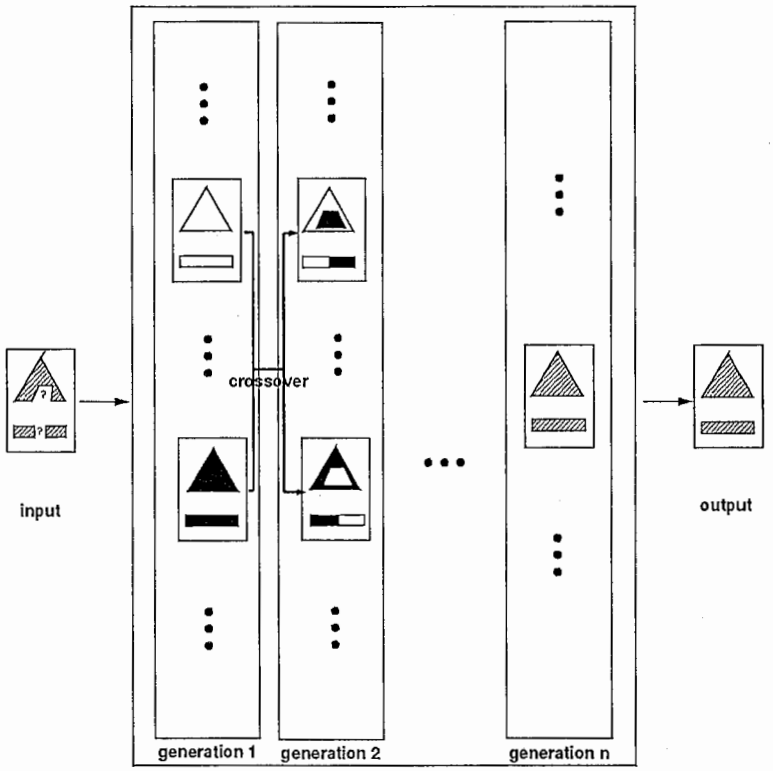


Figure 11: A scheme of the system

4.3 Assessment and generality

4.3.1 Self-assessment

A flaw of rule-based systems using context-free parsers, is that they often fail to deliver a solution for trivial reasons such as a word missing in a dictionary. On the contrary, the previous system always delivers an output for any input, would it be "bad". Of course, this would be of no meaning if the quality of outputs would not be evaluated. Hence, when delivering a solution, the system scores it. This score is the measure of the distance between the input and the output. It is independent of the inner knowledge of the system.

4.3.2 Generality

From a general point of view, natural language processing seriously lacks methods to assess its results. Some machine translation systems, viewed as expert systems, return an evaluation of their work in terms of their knowledge (grammar) [Tong 89], some other systems evaluate the result according to thesaurus classification and statistical frequencies [Furuse and Iida 92b], [Sato 91]. All these methods are specific.

The previous system, on the contrary, delivers a score which is a formal distance between the input and the output. Thus, it is independent of the linguistic representation chosen (dependency or constituency). This score gives an intrinsic information. It is not the case with a proposal such as [Harrison *et al.* 91], as outputs from different systems must adopt the same kind of representation, and because only part of the output is taken into account. Our score could lead to reliable comparisons between systems.

4.3.3 The thesaurus distance

In [Sumita and Iida 92], translation is basically done by applying example patterns to input chunks of texts. This is done by using a distance defined according to a thesaurus. This thesaurus is a semantic hierarchy, *i.e.* a tree. As an example, Figure 12 shows a fragment of the hierarchy under the *actions* item.

To calculate the distance between two words, one simply considers the height of the smallest subtree dominating both words. By convention, a tree reduced to one node is of height zero. Hence, for instance,

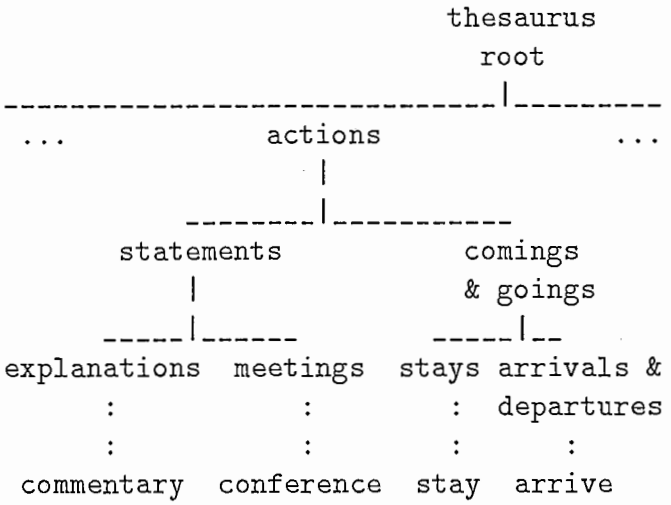


Figure 12: A fragment of the thesaurus

$$\begin{aligned} \text{dist}(commentary, conference) &= 1 \\ \text{dist}(commentary, stay) &= 2 \end{aligned}$$

For their application, [Sumita and Iida 92] divide this number by the maximal height, 3 to obtain a number between 0 and 1.

For each word, it is possible to write down a board for which the tree is the path in the thesaurus corresponding to this word, and the string is the word itself. This is illustrated in Figure 13. Applying the Selkow distance on these trees delivers the results given in Figure 14, which are the same as those obtained directly in the hierarchical tree.

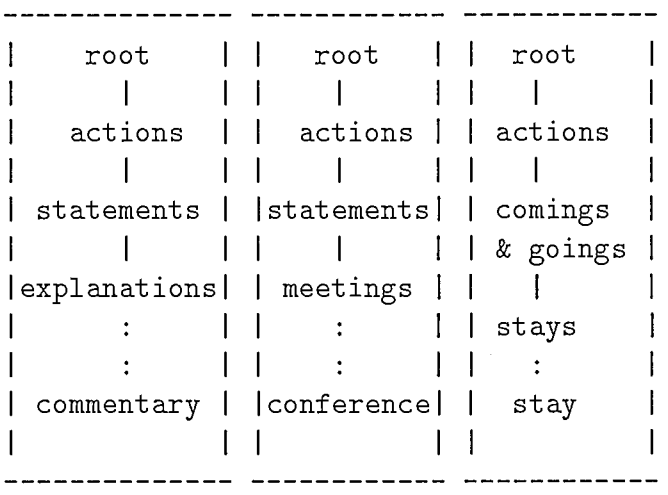


Figure 13: Reformulating the thesaurus with boards

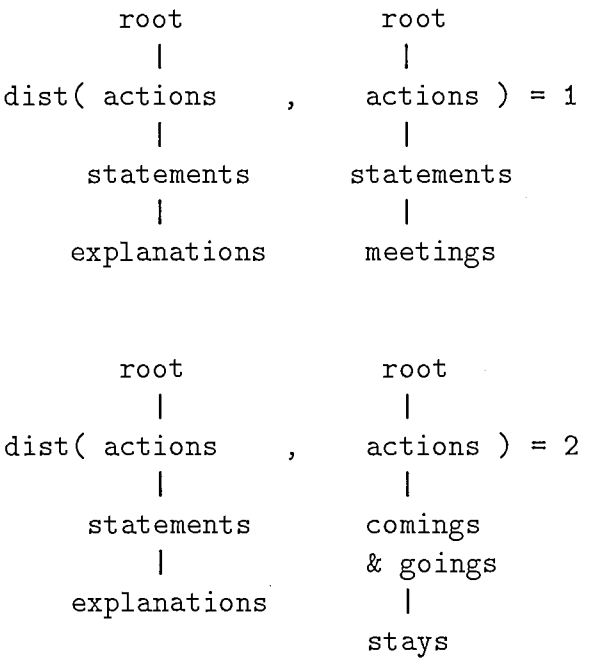


Figure 14: Computing distances

5 Blending

Some link can be established between distance and pattern-matching (Appendix D). Although our pattern-matching operation, identification, can be criticised for its non-determinism it is preferable over an operation where variables would instantiate with only one element in a string. Such an operation is of poor consequences when compared with a string distance. On the contrary, as the variables in identification can take their value from an empty forest to a forest of any length, forests factor a sequence of adjacent links between two strings in traces defined by Wagner and Fischer (see [Lepage et al. 92] for more technical details).

Thanks to that, an interesting result has been established between identification and distance on forest patterns. Given two forest patterns, one can find at least one forest with the following properties:

- **its identification with either of the given forests yields this given forest back**, and,
- **summing the maximum lengths of variable instantiations resulting from the previous identification yields the distance between the two given forest patterns.**

We recall that

- identification is a generalisation of pattern-matching on strings and trees;
- the distance we defined is also a generalisation and an extension of the Wagner and Fischer distance on strings and the Selkow distance on trees.

A system which would use identification and the distance on pattern woods could have two original interesting properties:

- **non-directionality** which is more than bi-directionality. Analysis and generation are particular cases of non-directional completion;
- **self-assessment** independently of its inner knowledge.

Conclusion

This report, we believe, opens some new ideas toward a blending of NLP approaches.

We proposed to separate a text from its linguistic structure and to make their association explicit. A unique data structure underlies strings and trees. A unique pattern-matching operation and a unique distance apply on this data structure. They are generalisations and extensions of pattern-matching and distances on strings and trees.

Appendices

A Data structures

This section presents the basic data structure we work with and the notion of patterns as a set notion.

A.1 Vocabulary

In the following, we consider a finite non-empty set. We call it the vocabulary. We note it $\mathcal{V}$. We sometimes will adjoin a special element noted ε which we call the empty element (not the same thing as the empty set).

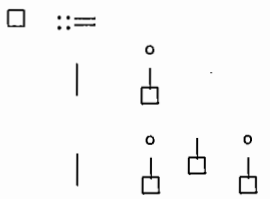
A.2 Forests

The set of forests we consider is the classical one, a list of trees. Formally, they are oriented, ordered and labeled trees. The set of labels V is denumerable. We call F the set of forests.

A grammar for forests can be given in the following graphical form which extends the Backus Normal Form. This grammar does not favour a particular traversal, from left to right nor right to left.

A node is represented by $\circ$. A forest is one of the three following

- empty;
- a tree (a node dominating a forest);
- a forest with trees on both sides, in the general case.



The labelling function associates a word to each node. According to this grammar, a forest can be recursively decomposed starting from both trees on the sides and repeatedly until reaching an empty tree or a simple tree in the middle.

Clearly, a string is a particular case of a forest. It is a forest with only one element on the vertical axis. Its vertical dimension is 1. We call $\mathcal{S}$ the set of strings labelled on $\mathcal{V}$.

A tree is a pair of a node, called root, and a forest, dominated by the root. It is a degenerated case of a forest on the horizontal axis. Its horizontal dimension is 1. We call $\mathcal{T}$ the set of oriented, ordered trees labelled on $\mathcal{V}$.

The intersection of the set of trees (degenerated forests on the horizontal axis) and the set of strings (degenerated forests on the vertical axis) is the set of forests with only one node (or no node at all for the empty string and tree). This set is in bijection with $\mathcal{V} \cup \{\varepsilon\}$.

$$\mathcal{V} \cup \{\varepsilon\} = \mathcal{S} \cap \mathcal{T}$$

A.3 Patterns

We define a variable as a (possibly denumerable) subset of structured elements.

On $\mathcal{V}$, we define a pattern μ which extension is $\overline{\mu} = \mathcal{V} \cup \{\varepsilon\}$. This element is such that it verifies:

$$\forall a \in \mathcal{V} \cup \{\varepsilon\}, \begin{cases} \mathrm{dist}(a,\mu) = \mathrm{dist}(\mu,a) = 0 \\ \mathrm{idn}(a,\mu) = \mathrm{idn}(\mu,a) = a \end{cases}$$

with dist being the set extension of the Wagner and Fischer distance and idn identification on the vocabulary.

For forests, we define a special pattern φ with the extension:

$$\overline{\varphi} = \{\varepsilon, \mu\} \cup \overline{\mu(\varphi).\varphi.\mu(\varphi)}$$

This definition is similar to the formal definition of the neutral element of identification given in [Lepage 89]. The extension of a forest pattern is defined by the following propositions:

$$\overline{u, v} = \overline{u}, \overline{v}$$

$$\overline{a(u_1, \ldots, u_n)} = a(\overline{u_1}, \ldots, \overline{u_n})$$

General forest patterns are obtained from constant forests on the leaves of which the pattern φ may appear. As a particular case, if u is a constant object (be it a string or a tree), then $\overline{u} = \{u\}$.

B Identification

Identification has been introduced informally in [Zaharin 86]. It has been formalised in [Lepage 89]. In a first step, identification is formalised on constant forests, which cover constant strings and trees. Then, the extension of forests to *woods*, allows the definition of a neutral element for identification. Thanks to this neutral element, an interpretation of string variables and forest variables in terms of woods is possible. Identification on patterns will simply be identification on woods resulting from this interpretation of forest variables.

B.1 Identification on constant objects

We recall that the set of forests is a superset of strings and trees. Defining identification on forests is hence sufficient. Forests are labeled on words. Hence, we first define an operation on the set of words $\mathcal{V}$, which we call idn.

$$\forall (v, v') \in \mathcal{V} \times \mathcal{V}, \begin{array}{lll} \mathrm{idn}(v, v') = v & \Leftrightarrow & v = v' \\ \mathrm{idn}(v, v') = \mathrm{fail} & \Leftrightarrow & v \neq v' \end{array}$$

So as to construct a neutral element, we introduce a new element μ outside of $\mathcal{V}$. We note $\mathcal{V}'$ the set $\mathcal{V} \cup \{\mu\}$. With this new element,

$$\mathrm{idn}(\mu, \mu) = \mu$$

$$\forall v \in \mathcal{V}, \mathrm{idn}(\mu, v) = \mathrm{idn}(v, \mu) = v$$

With this, identification is a commutative law from $\mathcal{V}'$ to $\mathcal{V}' \cup \{\mathrm{fail}\}$ with μ as the neutral element.

This law can be represented in the following hollow array, where only edges and the diagonal are not fail. All blank cells stand for fail.

idn	.	$\cdots$	v	$\cdots$
.	.	$\cdots$	v	$\cdots$
$\vdots$	$\vdots$	$\ddots$		
v	v		v	
$\vdots$	$\vdots$			$\ddots$

This law allows us to formulate the equality between two words thanks to the following property:

$$\forall (v, v') \in \mathcal{V} \times \mathcal{V}', v = v' \Leftrightarrow \mathrm{idn}(v, v') \neq \mathrm{fail}$$

Let us now generalise idn to forests through the labelling function. Identification is generalised on forests according to the following array, based on the formal definition of forests.

idn	$\emptyset$	$\circ$	$\circ\ \ \circ$
$\emptyset$	$\emptyset$	fail	fail
$\circ$	fail	tree case	fail
$\circ\ \ \circ\ \ \circ$	fail	fail	forest case

This array is hollow, and symmetrical with respect to the diagonal. For the tree case, identification on the roots is performed according to identification on nodes, and the identification on forests is recursively applied. In a similar way, we distribute identification on nodes and

$$\mathrm{idn}(\ \circ\ ,\ \circ\ ) = \begin{array}{c}\mathrm{idn}(\circ,\circ)\\ |\\ \mathrm{idn}(\Box,\Box)\end{array}$$

Figure 15: Case of constant trees

forests in the forest case. The order in which identification is distributed

$$\mathrm{idn}(\ \circ\ \ \circ\ ,\ \circ\ \ \circ\ ) = \begin{array}{c}\mathrm{idn}(\circ,\circ)\\ |\\ \mathrm{idn}(\Box,\Box)\end{array} \quad | \quad \mathrm{idn}(\Box,\Box) \quad \begin{array}{c}\mathrm{idn}(\circ,\circ)\\ |\\ \mathrm{idn}(\Box,\Box)\end{array}$$

Figure 16: Case of constant forests

in both cases has no importance on the result. A failure at a particular point implies a failure for the whole identification at hand.

Equality on forests may be defined from identification in the following way:

$$\forall (f, f') \in \mathcal{F} \times \mathcal{F}, f = f' \Leftrightarrow \mathrm{idn}(f, f') \neq \mathrm{fail}$$

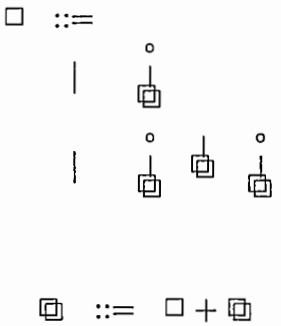
This proposition holds for strings and trees, as they are subsets of forests.

B.2 Identification with variables

B.2.1 Woods

Woods *Multi-arborescences à tranches* (which we translate by sliced forests) have been introduced in [Verastegui 82] as a suitable data structure for parallel programming of tree-transformational systems. Three types of special nodes were introduced: sets, lists and traces. We use a restriction of sliced forests, that with the set special node. We call them *woods* (a translation of the French *bois*).

Definition of woods They can be defined in a semi-graphical grammar as follows:



The labelling function is extended to $\mathcal{V}'$. It assigns an element of $\mathcal{V}'$ to a node, represented by a dot. Slices are noted ⧉. If a slice has only one element, it is this element. The order in which forests are enumerated in slices is not relevant,

This grammar contains the forest grammar given above. Hence, the set of woods, noted $\mathcal{F}_s$ includes the set of forests. $\mathcal{F} \subset \mathcal{F}_s$.

A spatial interpretation of slices can be given by considering slices as a depth dimension in addition to the horizontal one (strings) and the vertical one (trees).

B.2.2 Identification on woods

General array for identification The external form of identification on woods is the same as for constant forests. Only the diagonal is non-empty. Anywhere else, identification fails.

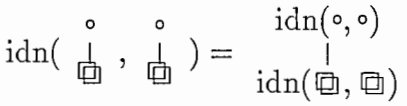


Figure 17: Case of sliced trees

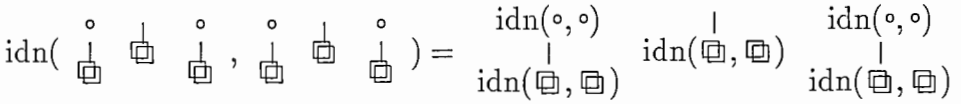


Figure 18: Case of woods

idn	$\emptyset$	∘ | ▣	∘ | ▣ ∘ | ▣ ∘
$\emptyset$	$\emptyset$	fail	fail
∘ | ▣	fail	tree case	fail
∘ | ▣ ∘ | ▣ ∘	fail	fail	forest case

The tree and forest cases are a simple copy of those cases for constant trees and forests.

Identification between slices To complete the previous definition, identification between slices must be defined. This is done by positing that identification is distributed over all possible pairs of forests from the first and second argument. A failure is similar to $\emptyset$ for the construction of slices. In a simple manner, one may consider that a slice is either a forest or a forest plus a slice. Hence the following array.

idn	$\square$	$\square + \blacksquare$
$\square$	$\mathrm{idn}(\square, \square)$	$\mathrm{idn}(\square, \square) + \mathrm{idn}(\square, \blacksquare)$
$\square + \blacksquare$	$\mathrm{idn}(\square, \square) + \mathrm{idn}(\blacksquare, \square)$	$\mathrm{idn}(\square, \square) + \mathrm{idn}(\blacksquare, \blacksquare)$

B.2.3 A neutral element for identification

Identification is a relation from $\mathcal{F}_s$ to $\mathcal{F}_s \cup \{\texttt{fail}\}$. We now look for a neutral element for identification. Let us call this element φ. It must verify the following property:

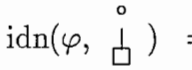
$$\forall f \in \mathcal{F}_s, \mathrm{idn}(f, \varphi) = \mathrm{idn}(\varphi, f) = f$$

which implies, as $\mathcal{F} \subset \mathcal{F}_s$,

$$\forall f \in \mathcal{F}, \mathrm{idn}(f, \varphi) = \mathrm{idn}(\varphi, f) = f$$

According to the forest grammar, this neutral element must verify the following propositions:

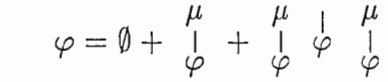
$$\mathrm{idn}(\varphi, \emptyset) = \emptyset$$



Now, φ may be defined as a three-slice sliced forest in which

- the first slice is empty;
- the second one is a neutral tree;
- the third one is a non-trivial neutral forest.

Because the neutral element for words is μ, the nodes in φ must bear μ. But identification is recursive, so the inner forests in φ must equal φ. In summary, φ is defined in the following way.



By construction, this element is a neutral element for identification on woods.

Interpretation of variables In a forest with variables, each variable is simply replaced by the neutral element for identification, φ. Hence, the notion of string variable and forest variable is replaced by only one notion, that of neutral element for identification, which is valid on the set of woods.

Identification with variables Identification on forests with variables is simply performed in two steps. First, variables are replaced by the neutral element φ and then identification is performed on the woods obtained. The result of identification of two forests with variables is a sliced forest in the general case.

To summarise, we formalised identification on a general data structure, woods, and introduced a neutral element for it. As a result, the result of identification on two woods is unique. The non-determinism of the pattern-matching operation has been shifted into the data structure, wood.

B.3 Grammars of boards and system

B.3.1 Boards

A board is a pair of a string and a tree.

$$\forall s \in \mathcal{S}, \forall t \in \mathcal{T}, (s,t) \text{ is a board}$$

B.3.2 Grammar of boards

A grammar is a set of boards. Let φ be a symbol outside of $\mathcal{V}$. Then we note $X' = X \cup \{\varphi\}$ with X being any of the sets $\mathcal{V}$, $\mathcal{S}$, $\mathcal{T}$ or $\mathcal{F}$.

B.3.3 System

Formulation with equality Let G be a grammar, the system built on the grammar G realises the following function M:

- The domain of M is $\mathcal{S}' \times \mathcal{T}'$;
- The co-domain of M is $\mathcal{P}(G)$, the power set of G;
- $M(\varphi,\varphi) = G$;
- $\forall s \in \mathcal{S}, M((s,\varphi)) = \{(s,t) \in G / t \in \mathcal{T}\} = (\{s\} \times \mathcal{T}) \cap G$;
- $\forall t \in \mathcal{T}, M((\varphi,t)) = \{(s,t) \in G / s \in \mathcal{S}\} = (\mathcal{S} \times \{t\}) \cap G$;
- $\forall s \in \mathcal{S}, \forall t \in \mathcal{T}, M((s,t)) = \{(s,t)\} \cap G$

This definition seems rather complicated for the signification it stands for. This comes from the fact that this first definition applies on constant objects. When we extend it to more complex objects, it will get simpler.

This function can also be seen as an operation which aims at equalising φ with a string or a forest so that the formed pair belongs to the grammar. Hence, the general form of the function will be:

$$M((s,t)) = \{(s',t') \in G / s' = s \in \mathcal{S} \wedge t' = t \in \mathcal{T}\}$$

Formulation with identification Equality may be defined from identification, the neutral element of which is φ.

Let $\mathcal{S}_s$ be the set of sliced strings labeled on $\mathcal{V}'$ obtained by interpretation of strings with variables and $\mathcal{T}_s$ be the set of sliced trees labeled on $\mathcal{V}'$ obtained by interpretation of trees with variables. A board is an element of $\mathcal{S}_s \times \mathcal{T}_s$. Let $\mathcal{S}'_s = \mathcal{S}_s \cup \{\varphi\}$ and $\mathcal{T}'_s = \mathcal{T}_s \cup \{\varphi\}$ Let us recall that fail $\notin \mathcal{F}_s$.

A grammar is a subset of $\mathcal{S}_s \times \mathcal{T}_s$. The function realised by a system built on a grammar G is a function M from $\mathcal{S}'_s \times \mathcal{T}'_s$ to $\mathcal{P}(\mathcal{S}_s \times \mathcal{T}_s)$, such that $\forall (s', t') \in \mathcal{S}'_s \times \mathcal{T}'_s$,

$$M((s', t')) = \{(\mathrm{idn}(s', s), \mathrm{idn}(t', t)) \in \mathcal{S}_s \times \mathcal{T}_s \ / \ (s, t) \in G\}$$

C Metrics

In this section, we recall the mathematical definition of a metric. A natural generalisation of a metric to subsets is also presented. This can apply for patterns defined as sets. We show that the Wagner and Fischer distance is a metric over strings if it is defined from a metric over the vocabulary elements. Some properties are also proved.

C.1 Metrics

Definition 1 (Metric) *Let $\mathcal{V}$ be a set,* dist *a function from $\mathcal{V} \times \mathcal{V}$ to $\mathbb{R}^+$, the set of non-negative real numbers,* dist *is a metric on $\mathcal{V}$ if and only if*

- *(equality axiom)* $\forall (a, b) \in \mathcal{V}^2,\ \text{dist}(a, b) = 0 \Leftrightarrow a = b$
- *(commutativity)* $\forall (a, b) \in \mathcal{V}^2,\ \text{dist}(a, b) = \text{dist}(b, a)$
- *(triangle inequality)* $\forall (a, b, c) \in \mathcal{V}^3, \text{dist}(a, c) \leq \text{dist}(a, b) + \text{dist}(b, c)$

C.2 Generalisation to sets

A distance on $\mathcal{V}$ can be generalised in a natural way on the set of non-empty subsets of $\mathcal{V}$.

$$\forall (A, B) \in (2^{\mathcal{V}} \setminus \{\emptyset\}),\ \text{dist}(A, B) = \min_{(a,b) \in A \times B} \text{dist}(a, b)$$

The function obtained in this way is no more a metric. Commutativity is still verified but the first and third properties do not hold in the general case.

The loss of the first property is proved for $card(\mathcal{S}) \geq 3$. Then, there exist two subsets $A = \{a, b\}$ and $B = \{a, c\}$. We have $b \neq c$, which implies that $A \neq B$, and $\text{dist}(A, B) = \text{dist}(a, a) = 0$. As a matter of fact, the equality axiom has to be replaced by the following proposition.

$$\forall (A, B) \in (2^{\mathcal{V}} \setminus \{\emptyset\}), \text{dist}(A, B) = 0 \Leftrightarrow A \cap B \neq \emptyset$$

The loss of the third property is illustrated on the following example. Let $A = \{a\}$, $B = \{a, c\}$ and $C = \{c\}$ with $a \neq c$. Then, $\text{dist}(A, C) > 0$ and $\text{dist}(A, B) = \text{dist}(B, C) = 0$. The triangle equality is not verified.

C.3 The Wagner and Fischer distance

C.3.1 Definition

Definition 2 (Edit operation) *Let $(a,b) \in (\mathcal{V} \cup \{\varepsilon\})^2$, it is called an edit operation and is noted $a \to b$ if and only if $a \neq b$. We call $\mathcal{E}$ the set of edit operations.*

An edit operation $a \to b$ is called an insertion iff $a = \varepsilon$; it is a deletion iff $b = \varepsilon$ and it is called a replacement otherwise.

A string $u \in \mathcal{V}^*$ directly derives the string $v \in \mathcal{V}^*$ if and only if

$$\exists\, a \to b \in \mathcal{E},\ \exists\, (x,y) \in (\mathcal{V}^*)^2 \ /\ u = x.a.y \wedge v = x.b.y$$

Lemma 1 (Unic edit operation) *If $u \in \mathcal{V}^*$ directly derives $v \in \mathcal{V}^*$ then, $\exists!\, a \to b \in \mathcal{E},\ \exists\, (x,y) \in (\mathcal{V}^*)^2 \ /\ u = x.a.y \wedge v = x.b.y$*

This lemma means that the edit operation involved in a direct derivation is unique. But, where it takes place may not be unique. For instance, consider *aaa* directly deriving *aa* by the edit operation $a \to \varepsilon$ which can take place on any of the three a's of the first string.

Proof Suppose there are two edit operations $a \to b$ and $a' \to b'$. Then, there exist x, y, x', y' in $\mathcal{V}^*$ such that

$$\begin{cases} u &= x.a.y &= x'.a'.y' \\ v &= x.b.y &= x'.b'.y' \end{cases}$$

There are four cases, which can be factored in two, by considering that x is a prefix of x', and either y is a prefix of y' or y' is a prefix of y. This leads to the following system of equations:

$$\begin{cases} a.y'' &= x''.a' \\ b.y'' &= x''.b' \end{cases} \vee \begin{cases} a.y &= x''.a'.y'' \\ b.y &= x''.b'.y'' \end{cases}$$

which can only be satisfied if $a = b$ and $a' = b'$ (impossible by the definition of an edit operation), or if $a = a'$ and $b = b'$. □

Lemma 2 *If $u \in \mathcal{V}^*$ directly derives $v \in \mathcal{V}^*$ with $b \to c$ as the edit operation then, for any $a \in \mathcal{V}$, $a.u$ directly derives $a.v$ with the same edit operation.*

Proof Let $b' \rightarrow c'$ be the edit operation involved in the direct derivation from $a.u$ to $a.v$.

$$\exists (x', y') \in (\mathcal{V}^*)^2 \; / \; \begin{array}{rcl} a.u & = & x'.b'.y' \\ a.v & = & x'.c'.y' \end{array}$$

Now, $x' = \varepsilon$ is impossible because $b' \neq c'$ by definition of an edit operation. Hence, $x' = a.x''$ and

$$\left\{ \begin{array}{rcl} a.u & = & a.x''.b'.y' \\ a.v & = & a.x''.c'.y' \end{array} \right.$$

which implies that $u = x''.b'.y' \wedge v = x''.c'.y'$. By the unicity of edit operation in direct derivation, necessarily, $b = b' \wedge c = c'$. □

A string $u \in \mathcal{V}^*$ derives a string $v \in \mathcal{V}^*$ if and only if

$$\exists\, u_1, u_2, \ldots, u_n \in \mathcal{V}^* / \; \begin{array}{l} u = u_1 \wedge \\ v = u_n \wedge \\ \forall i \,/\, 1 \leq i < n,\ u_i \text{ directly derives } u_{i+1} \end{array}$$

Let dist be a metric on the set $\mathcal{V} \cup \{\varepsilon\}$. Each editing operation $a \rightarrow b$ can be assigned the value of $\text{dist}(a, b)$. Each given direct derivation can also be assigned the value of the edit operation involved. Thus we can compute the minimal number $\sum_{i=1}^{i=n-1} \text{dist}(u_i, u_{i+1})$ over all possible derivations from u_1 to u_n. This defines the Wagner and Fischer distance.

Definition 3 (Wagner and Fischer distance) *Let $\mathcal{V}$ be a vocabulary,* dist *a metric on $\mathcal{V} \cup \{\varepsilon\}$,* dist *can be extended to $\mathcal{V}^*$ in the following way: $\forall u, v \in \mathcal{V}^*$,*

$$\begin{array}{rcl} u = v & \Leftrightarrow & \text{dist}(u, v) = 0 \\ u \neq v & \Leftrightarrow & \text{dist}(u, v) = \min(\displaystyle\sum_{i=1}^{i=n-1} \text{dist}(u_i, u_{i+1})) \end{array}$$

over all $(u_1, \ldots, u_n)$ derivations from u to v. Then, dist *is a metric on $\mathcal{V}^*$.*

We must prove the existence of $\text{dist}(u, v)$ for all $(u, v) \in (\mathcal{V}^*)^2 \; / \; u \neq v$. Let $u = a_1 \ldots a_m$ and $v = b_1 \ldots b_n$ with $a_i, b_i \in \mathcal{V}$. Clearly, the derivation

$$(a_1 \ldots a_m,\ a_2 \ldots a_m,\ \ldots,\ a_m,\ \varepsilon,\ b_1,\ b_1.b_2,\ \ldots,\ b_1 \ldots b_n)$$

exists. □

The Wagner and Fischer distance can be defined in a second way.

Definition 4 (Wagner and Fischer distance) *Let $\mathcal{V}$ be a vocabulary,* dist *a metric on $\mathcal{V} \cup \{\varepsilon\}$,* dist *can be extended to $\mathcal{V}^*$ in the following way: $\forall (a,b) \in (\mathcal{V})^2$, $\forall (u,v) \in (\mathcal{V}^*)^2$,*

$$\begin{aligned} \text{dist}(a.u, \varepsilon) &= \text{dist}(a, \varepsilon) + \text{dist}(u, \varepsilon) \\ \text{dist}(a.u, b.v) &= min(\begin{array}{l} \text{dist}(a, \varepsilon) + \text{dist}(u, b.v), \\ \text{dist}(a, b) + \text{dist}(u, v), \\ \text{dist}(\varepsilon, b) + \text{dist}(a.u, v) \end{array}) \end{aligned}$$

dist *is the Wagner and Fischer distance on $\mathcal{V}^*$.*

We will not prove the equivalence of the two definitions.

C.3.2 Metric

Lemma 3 *Let $\mathcal{V}$ be a vocabulary and* dist *a metric on $\mathcal{V} \cup \{\varepsilon\}$, then the Wagner and Fischer distance over $\mathcal{V}^*$ is a metric.*

Proof We have to verify that the values of the Wagner and Fischer distance are all non-negative, and that it verifies the equality axiom, commutativity and the triangle inequality.

Non-negative values As dist is a metric over $\mathcal{V} \cup \{\varepsilon\}$, its values are non-negative. According to its first definition the Wagner and Fischer distance involves only additions of such values or zero, hence it yields only non-negative values.

Equality axiom If there would be $u, v \in \mathcal{V}^*$ such that $u \neq v \wedge \text{dist}(u,v) = 0$, then there would exist a derivation from u to v with cost zero. So there would exist an edit operation $a \rightarrow b$ $(a \neq b)$ with cost zero. But this is impossible because dist is a metric on $\mathcal{V} \cup \{\varepsilon\}$.

Commutativity Obvious by construction of the Wagner and Fischer distance and commutativity of dist on $\mathcal{V} \cup \{\varepsilon\}$.

Triangle inequality Consider a derivation $(v_1, v_2, \ldots, v_n)$ from u to w, such that there exists $k \ / \ 1 \leq k \leq n$ for which $v_k = v$. Any such derivation is a derivation from u to w and also a derivation from u to v followed by a derivation form v to w. The Wagner and Fischer distance being the minimum over all possible derivations, the triangle inequality follows.

$$\text{dist}(u,w) \leq \text{dist}(u,v) + \text{dist}(v,w)$$

□

In the sequel, we suppose that $\mathcal{V}$ is finite and has at least two elements. Because $\mathcal{V}$ has at least one element and is finite, there exists a least non-zero value for dist on $\mathcal{V} \cup \{\varepsilon\}$ which we call δ.

C.3.3 Properties

Lemma 4 $\forall (a, b, c) \in (\mathcal{V} \cup \{\varepsilon\})^3$, $\text{dist}(a.b, a.c) = \text{dist}(b, c)$

Proof If $a = \varepsilon$, this is trivial. We suppose $a \neq \varepsilon$. If $b = c$, the equality becomes trivial: $\text{dist}(a.b, a.c) = 0 = \text{dist}(b, c)$. We suppose that $b \neq c$. Consider the derivation $(u_1 = a.b, u_2 = a.c)$ involving the only edit operation $b \rightarrow c$. Because the Wagner and Fischer distance is the minimum over all possible derivations, we have:

$$\text{dist}(a.b, a.c) \leq \text{dist}(b, c)$$

Reciprocally, suppose $(u_1 = a.b, u_2, \ldots, u_{n-1}, u_n = a.c)$ is the derivation for which $\text{dist}(a.b, a.c)$ is reached. If $b \rightarrow c$ is involved then

$$\text{dist}(a.b, a.c) \geq \text{dist}(b, c)$$

else there must exist two edit operations $b \rightarrow e$ and $f \rightarrow c$ with $e, f \in \mathcal{V} \cup \{\varepsilon\}$ rendering account for the deletion/replacement of b and the insertion/replacement of c. Hence, $\text{dist}(a.b, a.c) \geq \text{dist}(b, e) + \text{dist}(f, c)$ Necessarily we must render account of the deletion of e and the insertion of f too, so the same rationale must repeat. As a derivation is finite, there eventually exists a derivation from b to c such that

$$\text{dist}(a.b, a.c) \geq \text{dist}(b, e_1) + \text{dist}(e_1, e_2) + \cdots + \text{dist}(f_2, f_1) + \text{dist}(f_1, c)$$

where either $f_n = e_n$ or $\text{dist}(e_n, f_n)$ is involved. The triangle inequality holds for dist on $\mathcal{V} \cup \{\varepsilon\}$, hence

$$\begin{aligned} \text{dist}(a.b, a.c) &\geq \text{dist}(b, e_1) + \text{dist}(e_1, e_2) + \\ &\quad \cdots + \text{dist}(f_2, f_1) + \text{dist}(f_1, c) \\ &\geq \text{dist}(b, c) \end{aligned}$$

By combining both inequalities, we have $\text{dist}(a.b, a.c) = \text{dist}(b, c)$ □

Lemma 5 $\forall a \in \mathcal{V}$, $(v, w) \in (\mathcal{V}^*)^2$, $\text{dist}(a.v, a.w) = \text{dist}(v, w)$

Proof Let $(a.u = v_1, \ldots, v_i, \ldots, v_n = a.v)$ be the derivation for which $\text{dist}(a.u, a.v)$ is reached. Consider the first v_i for which a is not a prefix. Necessarily, the previous direct derivation was from $a.v_i$ to v_i. If we delete this direct derivation, we get a derivation from u to $a.v$. By definition of dist as a minimum over all derivations, we have:

$$\text{dist}(u, a.v) \leq \text{dist}(a.u, a.v) - \text{dist}(a, \varepsilon)$$

Now, if we add a direct derivation from $a.v$ to v at the end of the previously built derivation, we get a derivation from u to v. Because dist is a minimum over all derivations, we have:

$$\text{dist}(u, v) \leq \text{dist}(a.u, a.v) - \text{dist}(a, \varepsilon) + \text{dist}(\varepsilon, a)$$

that is $\text{dist}(u, v) \leq \text{dist}(a.u, a.v)$ Reciprocally, consider a derivation $(u = v'_1, \ldots, v'_n = v)$ for which $\text{dist}(u, v)$ is reached. We can build the derivation $(a.u = v'_1, \ldots, a.v'_n = a.v)$ from $a.u$ to $a.v$. By the Unic edit operation lemma, this derivation involves the same edit operations. Hence its cost is the same as the derivation from u to v, that is $\text{dist}(u, v)$. By definition of dist as a minimum over all derivations, we have

$$\text{dist}(a.u, a.v) \leq \text{dist}(u, v)$$

Both inequalities give the expected result. □

Lemma 6 (Prefix)

$$\forall (u, v, w) \in (\mathcal{V}^*)^3,\ \text{dist}(u.v, u.w) = \text{dist}(v, w)$$

Proof Trivial given the previous lemma. Let $u = a_1 \ldots a_n$. Then,

$$\text{dist}(v, w) = \text{dist}(a_n.v, a_n.w) = \cdots = \text{dist}(a_1 \ldots a_n.v, a_1 \ldots a_n.w)$$

Obviously, the lemma is also true if u is a suffix instead of a prefix, *i.e.* $\text{dist}(v.u, w.u) = \text{dist}(v, w)$. □

Lemma 7 (Splitting)

$$\forall (u, v, u', v') \in (\mathcal{V}^*)^4,\ \text{dist}(u.v, u'.v') \leq \text{dist}(u, u') + \text{dist}(v, v')$$

Proof By application of the Prefix lemma,

$$\text{dist}(u, u') + \text{dist}(v, v') = \text{dist}(u.v, u'.v) + \text{dist}(u'.v, u'.v')$$

which gives by triangle inequality

$$\text{dist}(u.v, u'.v') \leq \text{dist}(u, u') + \text{dist}(v, v')$$

□

C.4 Generalisation to patterns

As the generalisation of a string pattern is a denumerable set, so is the cartesian product of the extension of two string patterns. Moreover, the values of the Wagner and Fischer distance have a lower bound, 0. These two propositions imply that there exists a minimal value for the Wagner and Fischer distance on all possible pairs of constant strings in the generalisation of two string patterns, and that there exists at least one pair for which the minimal value is reached.

Hence a distance between two string patterns can be defined in the following way:

$$\text{dist}(\overline{u}, \overline{v}) = \min_{u_i \in \overline{u}, v_j \in \overline{v}} \text{dist}(u_i, v_j)$$

As the generalisation of u is $\{u\}$ if u is a constant string, the following proposition holds: $\forall (u, v) \in \mathcal{V}^*$,

$$\begin{aligned} \text{dist}(\overline{u}, \overline{v}) &= \min_{u_i \in \overline{u}, v_j \in \overline{v}} \text{dist}(u_i, v_j) \\ &= \min_{u_i \in \overline{u}, v_j \in \overline{v}} \text{dist}(u_i, v_j) \\ &= \min_{u_i \in \{u\}, v_j \in \{v\}} \text{dist}(u_i, v_j) \\ &= \text{dist}(u, v) \end{aligned}$$

which shows that dist is a generalisation of the Wagner and Fischer distance on a larger set of objects.

This generalisation is no more a metric, as was already mentioned. For instance, the first axiom must read:

$$\text{dist}(\overline{u}, \overline{v}) = 0 \Leftrightarrow \overline{u} \cap \overline{v} \neq \emptyset$$

D Pattern-matching and distances

D.1 The string side

D.1.1 Traces and minimal woods

Wagner and Fischer defined the *trace* between two strings as a graphical representation in which a link between elements alignes the stringsa nd indicates a replacement if the elements are different. An element with no link is the result of an insertion (or a deletion).

his . *uncle* . *'s* . *friend* . *'s* . *son*

his . *aunt* . *'s* . *son*

Based upon such a trace, we can derive a string pattern containing variables:

his . *uncle* . *'s* . *friend* . *'s* . *son*

his . *aunt* . *'s* . *son*

→ *his*.$1.*'s*.$2.*son*

We proceed as follows:

- replace each element involved in an editing operation by a variable;
- merge two adjacent variables into one variable (recall that the variables in identification stand for strings).

The Wagner and Fischer distance value is the minimum number of elements inserted, replaced or deleted. This value is calculated on the set of all possible traces between the two strings. As there may be several minimum traces, there may be several minimum patterns. These patterns are factored into one wood structure, which we call the *minimum wood.*

D.1.2 Minimal woods and identification

A direct consequence of the previous definition of the minimal wood is that identification of the minimal wood with one of the strings from which it was derived yields this latter string.

$$\text{idn}(his.uncle.'s.friend.'s.son, his.1.'\text{s}.2.son) = his.uncle.'s.friend.'s.son$$

with $\$1 = \{uncle\}$ and $\$2 = \{friend.'s\}$.

$$\text{idn}(his.aunt.'s.son, his.1.'\text{s}.2.son) = his.aunt.'s.son$$

with $\$1 = \{aunt\}$ and $\$2 = \{\varepsilon\}$.

D.1.3 Minimal woods and distance

Another consequence is that, for two strings, given any string pattern in the minimal wood, the distance between the string is the sum of the maximal lengths of the strings which instantiate the variables during identification.

$$\begin{aligned}
\max(\text{length}(\text{inst}(\$1))) \; &+ \; \max(\text{length}(\text{inst}(\$2))) \\
&= \max(\text{length}(uncle), \text{length}(aunt)) \\
&+ \max(\text{length}(friend.'s), \text{length}(\varepsilon)) \\
&= \max(1,1) + \max(0,2) \\
&= 1 + 2 \\
&= 3 \\
&= \text{dist}(his.uncle.'s.friend, his.aunt.'s.son)
\end{aligned}$$

This result comes from the definition of a string pattern variable. A variable stands for a sequence of editing operations; hence, the maximum lengths of the instantiations yields the number of editing operations.

If the minimal wood does not contain any variables, then the value of the summation is zero, and, necessarily, $w = u = v$.

The previous result can be summarised as follows:

$$\forall (u,v) \in S,\ \exists w \in F_s \,/\, \begin{cases} \text{idn}(u,w) = u \\ \text{idn}(v,w) = v \\ \text{dist}(u,v) = \displaystyle\sum_{\$i \text{ var in } w} \max_{s_i \in \{\text{inst}(\$i)\}} \text{length}(s_i) \end{cases}$$

where s_i is an instantiation of $\$i$ during $\text{idn}(u,w)$ or $\text{idn}(u,w)$.

D.1.4 Identification and distance on strings

As identification operates on string patterns, and we wish to compare identification as a pattern-matching operation with the Wagner and Fischer distance, we have now to consider string patterns.

We consider the distance on string patterns obtained by the set generalisation of the Wagner and Fischer distance. Let $(u.v)$ be a pair of string patterns. There exists at least one pair of constant strings in their extension for which the distance value is reached. Let us call P the set of all such elements. Let $(u_i, v_j) \in P$. Hence, there exists a string pattern w for which the three following propositions hold.

$$\begin{cases} \text{idn}(u_i, w) &= u_i \\ \text{idn}(v_i, w) &= v_i \\ \text{dist}(u_i, v_i) &= \sum\limits_{\$i \text{ var in } w} \max\limits_{s_i \in \{\text{inst}(\$i)\}} \text{length}(s_i) \end{cases}$$

The set of all such string patterns can be factored into one wood, which verifies the previous propositions. Hence, this wood is the minimal wood for the two string patterns at hand.

D.2 The tree side

Similar results may be found on trees. As shown in Section 4.1.3, the Wagner and Fischer distance is extended to forests and this extension appears to be an extension of the Selkow distance on trees. By using the definition of forest patterns given in Section A.3, a general result is obtained on forests:

$$\forall (u, v) \in F_s,\ \exists w \in F_s \ / \begin{cases} \text{idn}(u, w) &= u \\ \text{idn}(v, w) &= v \\ \text{dist}(u, v) &= \sum\limits_{\$i \text{ var in } w} \max\limits_{s_i \in \{\text{inst}(\$i)\}} \text{length}(s_i) \end{cases}$$

where length(u) is the weight of a forest, that is the number of its nodes.

E Languages and distances

For notions on language theory, we refer the reader to [Salomaa 73] and [Lewis and Papadimitriou 81].

We first define *coronas*. They stand for sets of words at a given distance of a given language.

Definition 5 (Coronas) *Let $\mathcal{L}$ be a language over $\mathcal{V}$, for a given x in $\mathbb{R}^+$, the following sets are called coronas:*

$\mathcal{L}_{=x} = \{w \in \mathcal{V}^* / \exists w' \in \mathcal{L}, \text{dist}(w', w) = x\}$
$\mathcal{L}_{\leq x} = \{w \in \mathcal{V}^* / \exists w' \in \mathcal{L}, \text{dist}(w', w) \leq x\}$
$\mathcal{L}_{>x} = \{w \in \mathcal{V}^* / \exists w' \in \mathcal{L}, \text{dist}(w', w) > x\}$

A natural question concerns the nature of coronas given the family of languages they are defined from.

Problem 1 (Regular (resp. context-free) coronas) *Let $\mathcal{L}$ be a regular (resp. context-free) language over $\mathcal{V}$, then $\forall x \in \mathbb{R}^+$, are $\mathcal{L}_{\leq x}$, $\mathcal{L}_{>x}$ and $\mathcal{L}_{=x}$ regular (resp. context-free) languages?*

Some partial results can be proved. We give the ones we could establish.

First, we may consider the set of words at a given distance of a given word.

Lemma 8 (Singletons' coronas) *Let $u \in \mathcal{V}^*$ and $x \in \mathbb{R}$, then the set $\{v \in \mathcal{V}^*, \text{dist}(u, v) \leq x\}$ is a regular language.*

The following lemma is necessary to establish the previous result. We give it without a proof.

Lemma 9 (Finite number of sequences) *Let $\mathcal{R}$ be a finite subset of $\mathbb{R}^+$, let $x \in \mathbb{R}^+$, there exists only a finite number of sequences $(x_1, x_2, \ldots, x_n)$ of any length in $\mathcal{R}$, such that*

$$\sum_{i=1}^{n} x_i \leq x$$

Proof for the Singletons' coronas lemma The number of values taken by dist on $\mathcal{V}$ is finite. So, the Finite number of sequences lemma applies, and hence there is only a finite number of sequences $(x_1, \ldots, x_n)$ such that $\sum_{i=1}^{n} x_i \leq x$. This proves that the set $\{v \in \mathcal{V}^*, \text{dist}(u, v) \leq x\}$ is finite, and hence trivially a regular language. □

A second result may be obtained on languages over a vocabulary of one symbol.

Lemma 10 (One-symbol coronas) *Let $\mathcal{L}$ be a regular language over $\mathcal{V} = \{a\}$, then $\forall x \in \mathbb{R}^+$, $\mathcal{L}_{=x}$, $\mathcal{L}_{\leq x}$ and $\mathcal{L}_{>x}$ are regular languages.*

Firstly, the problem can be "normalised" by setting $\text{dist}(a, \varepsilon) = 1$. With this hypothesis, $\text{dist}(a^n, a^p) = \text{dist}(a^{|n-p|}, \varepsilon) = |n - p|$. So, $\text{dist}(a^n, a^p) = x$ is equivalent to $n = p - x \vee n = p + x$.

Secondly, let us recall that any regular language on $\{a\}$ is of the form $\{a^p / p \in P\}$ with P an ultimately periodic set in $\mathbb{N}$, and reciprocally.

Proof for $\mathcal{L}_{=x}$ If $x \notin \mathbb{N}$ (an integral multiple of $\text{dist}(a, \varepsilon)$ in the general case), $\mathcal{L}_{=x}$ is empty. Suppose $x \in \mathbb{N}$. Let P be the ultimately periodic set associated with $\mathcal{L}$, so that $\mathcal{L} = \{a^p / p \in P\}$. So we can write

$$\mathcal{L}_{=x} = \{a^n / \exists p \in P / n = p + x \vee n = p - x\}$$

or, more clearly,

$$\mathcal{L}_{=x} = \{a^{p+x} / p \in P\} \cup \{a^{p-x} / p \in P \wedge p - x \in \mathbb{N}\}$$

Both sets $\{p + x / p \in P\}$ and $\{p - x / p \in P \wedge p - x \in \mathbb{N}\}$ are ultimately periodic sets, hence, $\mathcal{L}_{=x}$ is the union of two regular languages. The class of regular languages being closed under union, $\mathcal{L}_{=x}$ is a regular language. □

Proof for $\mathcal{L}_{\leq x}$ Let $E(x)$ be the integer part of x.

$$\mathcal{L}_{\leq x} = \bigcup_{i=0}^{E(x)} \mathcal{L}_{=i}$$

$\mathcal{L}_{\leq x}$ is a finite union of regular languages, hence it is a regular language. □

Proof for $\mathcal{L}_{>x}$ $\text{dist}(n, p) > x$ is equivalent to $x < p - n \vee x < n - p$ also equivalent to $n < p - x \vee p + x < n$. So we write

$$\begin{aligned}
\mathcal{L}_{>x} &= \{a^n \in \mathcal{V}^* / \exists p \in P, n < p - x \vee p + x < n\} \\
&= \mathcal{V}^* \setminus \{a^n \in \mathcal{V}^* / \forall p \in P, p - x \leq n \leq p + x\} \\
&= \mathcal{V}^* \setminus (\{a^n \in \mathcal{V}^* / \forall p \in P, p - x \leq n\} \\
&\qquad \cap \{a^n \in \mathcal{V}^* / \forall p \in P, n \leq p + x\})
\end{aligned}$$

P has a least element. So the set of a^n for which n is less than or equal to this least element plus a constant is finite. We know that any finite set in $\mathcal{V}^*$ is a regular language, that the intersection of a finite set with any set is finite and that the class of regular language is closed under complementation. This implies that $\mathcal{L}_{>x}$ is a regular language. □

The One-symbol coronas lemma can also be formulated for context-free languages as the classes of regular and context-free languages over $\mathcal{V} = \{a\}$ are equal.

In the general case, the Regular (resp. Context-free) corona problem is still open. Intermediate results can however be obtained. The first one is an implication.

Lemma 11 *Let $\mathcal{L}$ be a regular language, if $\forall x \in \mathbb{R}, \mathcal{L}_{=x}$ is a regular language, then, for any given x', $\mathcal{L}_{\leq x'}$ is also a regular language.*

Proof By the Finite number of sequences lemma, there exists only a finite number of $x_i \in \mathbb{R}$, such that x_i is sum of distance values on $\mathcal{V}$ and $0 \leq x_i \leq x$. Hence,

$$\mathcal{L}_{\leq x} = \bigcup_{finite} \mathcal{L}_{=x_i}$$

If all $\mathcal{L}_{=x_i}$ are regular languages, $\mathcal{L}_{\leq x}$ being a finite union of such languages is a regular language. □

Reciprocally, the following lemma holds.

Lemma 12 *Let $\mathcal{L}$ be a regular language, if $\forall x \in \mathbb{R}, \mathcal{L}_{\leq x}$ is a regular language, then, for any given x', $\mathcal{L}_{=x'}$ is also a regular language.*

Proof By the Finite number of sequences lemma, we can take the greatest x_i and the smallest $x_j \in \mathbb{R}$, such that $x_i < x \leq x_j$ and x_i and x_j are sums of distance values on $\mathcal{V}$. If $x = x_j$, then

$$\mathcal{L}_{=x} = \mathcal{L}_{\leq x_j} \setminus \mathcal{L}_{\leq x_i}$$

The family of regular languages being closed under set complementation, $\mathcal{L}_{=x}$ is a regular language. If $x \neq x_j$, $\mathcal{L}_{=x}$ is empty and is trivially a regular language. □

Regular coronas: tentative proof for $\mathcal{L}_{\leq x}$ We cannot use the theorem which tells that the family of regular languages is closed under any arbitrary homomorphism, as $\text{dist}(.,w) \leq x$ does not define a homomorphism. But we can use the equivalence between regular languages and regular expressions. Let us recall that a regular expression is defined as follows:

- $\forall e \in \mathcal{V} \cup \{\varepsilon\}$, e is a regular expression;
- let e_1, e_2 be regular expressions, then $e = e_1.e_2$ and $e = e_1 + e_2$ and $e = e_1^*$ are regular expressions with ., + and * standing for catenation, union and catenation closure (or iteration).
- a regular expression is nothing else than the previous.

Case $e \in \mathcal{V} \cup \{\varepsilon\}$ The Singletons' coronas lemma proves that $e_{\leq x}$ is a regular language for any $e \in \mathcal{V} \cup \{\varepsilon\}$.

Case $e = e_1.e_2$. We show that

$$e_{\leq x} \subset \sum_{x_1+x_2 \leq x} (e_1)_{\leq x_1}.(e_2)_{\leq x_2}, x_1, x_2 \in \mathbb{R}^+$$

By the Finite combinations lemma, there exists only a finite number of x_1, x_2 such that $x_1 + x_2 \leq x$ and x_1 and x_2 are sums of distance values on $\mathcal{V} \cup \{\varepsilon\}$. Let $u \in (e_1)_{\leq x_1}.(e_2)_{\leq x_2}$ with $x_1 + x_2 \leq x$, then

$$\exists u_1 \in (e_1)_{\leq x_1}, \exists u_2 \in (e_2)_{\leq x_2} / u = u_1.u_2$$

By the Splitting lemma, $\forall v_1 \in e_1, v_2 \in e_2$,

$$\text{dist}(u, v_1.v_2) \leq \text{dist}(u_1, v_1) + \text{dist}(u_2, v_2) \leq x_1 + x_2 \leq x$$

which means that $u \in (e_1.e_2)_{\leq x}$.

The reverse inclusion seems to be wrong and is left unproved.

Case $e = e_1 + e_2$. Then, trivially, $e_{\leq x} = (e_1)_{\leq x} + (e_2)_{\leq x}$.

Case $e = e_1^*$. Then, trivially, $e_{\leq x} = e_1^*.(e_1)_{\leq x}.e_1^*$.

References

[Boitet 82] Christian Boitet, rédacteur
Le point sur Ariane-78, début 1982
(Volume 1, Partie 1 : le logiciel)
Convention ADI no 81/423 Cap Sogeti Logiciel – GETA-Champollion, Grenoble, avril 1982.

[Boitet 91] Christian Boitet
Twelve Problems for Machine Translation
Proceedings of the International Conference on Current Issues in Computational Linguistics, Penang, June 1991, pp. 45-56.

[Boitet and Zaharin 88] Christian Boitet and Zaharin Yusoff
Representation trees and string-tree correspondences
Proceedings of COLING-88, pp 59-64, Budapest, 1988.

[Chandioux et Guérard 81] John Chandioux et M.F. Guérard
METEO : un système à l'épreuve du temps
Meta, **26**(1), 1981, pp 17-22.

[Colmerauer 70] Alain Colmerauer
Les systèmes-Q, un formalisme pour analyser et synthétiser des phrases sur ordinateur.
TAUM, Université de Montréal, 1970.

[Furuse and Iida 92a] Furuse Osamu and Iida Hitoshi
Cooperation between Transfer and Analysis in Example-based Framework
Proceedings of COLING-92, vol II, pp 645-651, Nantes, 1992.

[Furuse and Iida 92b] Furuse Osamu and Iida Hitoshi
An Example-based Method for Transfer-driven Machine Translation
Proceedings of the fourth International Conference on Theoretical and Methodological Issues in Machine Translation TMI-92, pp 139-150, Montréal, 1992.

[Harrison *et al.* 91] P. Harrison, S. Abney, E. Black, D. Flickenger, C. Gdaniec, R. Grishman, D. Hindle, R. Ingria, M. Marcus, B. Santorini, T. Strzalkowski

Evaluating Syntax Performance of Parser / Grammars of English
Proceedings of the Workshop on Evaluating Natural Language Processing Systems, ACL, 1991, pp. ?-?.

[Lepage 89] Yves Lepage
Un système de grammaires correspondancielles d'identification
Thèse, Université de Grenoble, juin 1989.

[Lepage 91] Yves Lepage
Parsing and Generating Context-Sensitive Languages with Correspondence Identification Grammars
Proceedings of the Pacific Rim Symposium on Natural Language Processing, Singapore, November 1991, pp. 256-263.

[Lepage 92a] Yves Lepage
Easier C programming
Input/output facilities
ATR report TR-I-0293, Kyoto, November 1992.

[Lepage 92b] Yves Lepage
Easier C programming
Some useful objects
ATR report TR-I-0294, Kyoto, November 1992.

[Lepage 92c] Yves Lepage
Easier C programming
Dynamic programming
ATR report TR-I-0295, Kyoto, November 1992.

[Lepage 93a] Yves Lepage
Analysis, generation and more by means of genetic algorithms
ATR report TR-IT-0002, Kyoto, June 1993.

[Lepage 93b] Yves Lepage
Analysis, generation and more in natural language processing by means of genetic algorithms
Proceedings of the First Meeting of Special Interest Group on Parallel Processing for Artificial Intelligence, pp 13-18, Tokyo, September 1993.

[Lepage 93c] Yves Lepage
Analysis, generation and more using genetic algorithms
Technical Report of IEICE, NLC 93–42-47, October 1993, pp. 1-8.

[Lepage & Zaharin 91] Yves Lepage and Zaharin Yusoff
Identification: a unification-like operation with variables instantiating to forests
Proceedings of the International Conference on Current Issues in Computational Linguistics, Penang, June 1991, pp. 262-275.

[Lepage et al. 92] Yves Lepage, Furuse Osamu and Iida Hitoshi
Relation between a pattern-matching operation and a distance: On the path to reconcile two approaches in Natural Language Proceessing
Proceedings of the First Singapore International Conference on Intelligent Systems, Singapore, November 1992, pp. 513-518.

[Lepage and Fais 94a] Yves Lepage and Laurel Fais
Syntactic trees for the sentences of ten dialogues
ATR report TR-IT-0036, Kyoto, January 1994.

[Lewis and Papadimitriou 81] Harry R. Lewis and Christos H. Papadimitriou
Elements of the theory of Computation
Prentice Hall Inc., 1981.

[Lowrance & Wagner 75] Roy Lowrance and Robert A. Wagner
An Extension of the String-to-String Correction Problem
Journal for the Association of Computing Machinery, Vol. 22, No. 2, April 1975, pp. 177-183.

[Nagao 84] Nagao Makoto
A framework of a Mechanical Translation between Japanese and English by Analogy Principle
in *Artificial intelligence and Human Intelligence*,
Elithorn A. and Banerji R. eds., Elsevier Science Publishers, 1984.

[Sadler and Vendelmans 90] Victor Sadler and Ronald Vendelmans
Pilot implementation of a bilingual knowledge bank
Proceedings of Coling-90, Helsinki, 1990, vol 3, pp. 449-451.

[Salomaa 73] Arto Salomaa
Formal Languages
Academic Press Inc., 1973.

[Saussure 15] Ferdinand de Saussure
Cours de linguistique générale
Payot, Paris, 1915.

[Sato 91] Sato Satoshi
Example-based Machine Translation
Ph. D. thesis, Kyoto University, September 1991.

[Sato and Nagao 90] Sato Satoshi and Nagao Makoto
Toward Memory-based Translation
Proceedings of Coling-90, Helsinki, 1990, vol 2, pp. 247-252.

[Sato 90] Sato Satoshi and Nagao Makoto
Example-Based Translation of Technical Terms
Proceedings of the fifth International Conference on Theoretical and Methodological Issues in Machine Translation TMI-93, pp 58-68, Kyoto, 1993.

[Selkow 77] Stanley M. Selkow
The Tree-to-Tree Editing Problem
Information Processing Letters, Vol. 6, No. 6, December 1977, pp. 184-186.

[Sumita and Iida 92] Sumita Eiichiro and Iida Hitoshi
Example-Based Transfer of Japanese Adnominal Particles into English
IEICE Trans. Inf. & Syst., vol E-75-D, No 4, July 1992, pp 585-594.

[Tai 77] Kuo-Chung Tai
The Tree-to-Tree Correction Problem
Journal for the Association of Computing Machinery, Vol. 26, No. 3, July 1979, pp. 422-433.

[Tong 89] Tong Loong Cheong
A data-driven control strategy for grammar writing systems
Machine Translation, 4(4), December 1989, pp. 177-193.

[Verastegui 82] Nelson Verastegui-Carvajal
Etude du parallélisme appliqué à la traduction automatisé par ordinateur.
Star-pale: un système parallèle
Thèse de Docteur-Ingénieur, INPG, Grenoble, mai 1982.

[Wagner & Fischer 74] Robert A. Wagner and Michael J. Fischer
The String-to-String Correction Problem
Journal for the Association of Computing Machinery, Vol. 21, No. 1, January 1974, pp. 168-173.

[Winograd 83] Terry Winograd
Language as a cognitive process
vol.1 Syntax
Addison Wesley, 1983.

[Zaharin 86] Zaharin Yusoff
Strategies and Heuristics in the analysis of natural language in Machine Translation
Ph.D.Thesis, Universiti Sains Malaysia, Penang, 1986.

[Zaharin 90] Zaharin Yusoff
Generation of synthesis programs in ROBRA (ARIANE) from String-Tree Correspondence Grammars (or a strategy for synthesis in machine translation)
Proceedings of COLING-90, vol 2, pp 425-430, Helsinki, 1990.

[Zaharin and Lepage 92] Zaharin Yusoff and Yves Lepage
On the specification of abstract linguistic structures in formalisms for Machine Translation
Proceedings of the International Symposium on Natural Language Understanding and AI, pp 145-153, Iizuka, July 1992.

Index