Tree bimorphisms and their relevance in the theory of translations

Cătălin-Ionuț Tîrnăucă

Rovira i Virgili University, Research Group on Mathematical Linguistics

Plaça Imperial Tàrraco 1, 43005, Tarragona, Spain

catalinionut.tirnauca@estudiants.urv.cat

Abstract

In the past few years, it was argued that in natural language processing (and especially in machine translation) researchers should focus more on formalisms that can model trees and tree transformations since they can capture syntax-sensitive transformations and execute certain reorderings of parts of sentences. Also, the closure under composition of such classes of tree transformations was stated as a necessary condition for improving translations: it is always easier to decompose a translation into smaller pieces which are easier to train, test and understand, and then obtain the original translation as a composition of them. We offer a quick and rather informal survey of formalisms that define classes of tree transformations: tree homomorphisms, tree transducers and synchronous grammars. Also, we present another approach towards the achieving of closure under composition property of such classes, the tree bimorphism, and we revisit some of the most recently obtained results.

Key words: machine translation, tree transformation, tree bimorphism, synchronous grammars.

Resum

Durant els darrers anys s'ha defensat que en processament del llenguatge natural (i especialment en traducció automàtica), els investigadors s'haurien de centrar especialment en formalismes capaços de modelar arbres i tranformacions d'arbres. La raó principal es que es creu que aquests formalismes poden ser un bon instrument per explicar las transformacions sensibles a la sintaxi i executar certes reordenacions de parts de l'oració. D'altra banda, la clausura respecte a la composició d'algunes classes de transformacions d'arbres ha estat considerada com una condició necessària per millorar les traduccions: sempre és més facil descomposar una traducció en peces més petites, que es poden optimitzar, comprovar i comprendre més fàcilment, i després obtenir una traducció original tot composant-les. En aquest article oferim una visió general i informal dels formalismes que defineixen les classes de transformacions d'arbres: homomorfismes d'arbres, transductors d'arbres i gramàtiques síncrones. A més, presentem un enfocament relativament nou per aconseguir la clausura respecte a la composició en aquestes classes, els bimorphismes d'arbres, y repassem alguns dels resultats obtinguts recentment.

Mots clau: traducció automàtica, transformacions d'arbres, bimorfismes d'arbres, gramàtiques síncrones.

Resumen

En los últimos años se ha discutido que en procesamiento del lenguaje natural (y especialmente en traducción automática), los investigadores deberían centrase especialmente en formalismos capaces de modelar árboles y transformaciones de árboles, puesto que estos mecanismos pueden detectar las transformaciones sensibles a la sintaxis y ejecutar ciertos reordenamientos de partes de la oración. Por otra parte, la clausura bajo la composición de estas clases de transformaciones de árboles se ha considerado como una condición necesaria para mejorar las traducciones: siempre es mejor descomponer una traducción en piezas pequeñas, que son más fáciles de comprobar, mejorar y comprender, y después obtener la traducción original como una composición de ellas. En este artículo se ofrece un panorama rápido e informal de los formalismos que definen las clases de trasformaciones de árboles: homomorfismos de árboles, transductores de árboles, y gramáticas síncronas. Además, presentamos un enfoque relativamente nuevo para conseguir la clausura bajo la composición de estas clases, el bimorfismo de árboles, y repasamos algunos de los resultados recientemente obtenidos.

Palabras clave: traducción automática, transformaciones de árboles, bimorfismos de árboles, gramáticas síncronas.

Table of contents

1. Introduction

Automatic translation between natural languages has become a true necessity for present-day society since international communication is increasing very fast (for example, due to the Internet), and this has stimulated the development of appropriate mathematical models and tools. Because of their attractive properties and well-developed theory, finite state machines, usually with some probability or weight features, were used with considerable success: (Brown et al 1993), (Mohri 1997), (Knight and Al-Onaizan 1998).

But in natural language translations there are many particularities and ambiguities that have to be considered. For example in English, the sentence “I have painted walls” has at least two meanings depending on how “painted” is interpreted: as an adjective or as a part of the verb of the sentence. Therefore, it is difficult to perform always a correct translation if the sentence is treated outside of the context in which it appears. Also, the structure of the English sentences is different from Arabic or German sentences. So, in most cases both the syntax (specifies the structure of an input sentence), and the semantics (associates a meaning to each structure of an input sentence) are needed.

Usually, the input sentence (which has to be a correct sentence of the input language) must have a certain structure. It seems natural and convenient to use the tree structure of a string, called parse tree or derivation tree, which describes how the sentence can be obtained by applying the grammatical rules of the language. Parsing represents the process of analyzing an input sentence and determining the syntactic structure associated with it. As an example, every English sentence can be decomposed in syntactic categories: noun phrase, verb phrase, determiner, etc. These are related by grammatical rules: a sentence is composed of a noun phrase and a verb phrase; a noun phrase can be formed by a determiner and a noun, etc. More about parsing can be found in: (Aho and Ullman 1972). Even if word-for-word translation models use contexts to disambiguate, they do not take advantage of the full syntactical information of a parse tree, so more powerful devices were called for.

Thus in the last couple of years, researchers from computational linguistic community have shown an increasing interest in formalisms that can model trees and tree transformations, especially because of their ability to capture syntax-sensitive processes and perform different reorderings of parts of sentences. Thus, the new field of syntax-based machine translation was established; e.g., see (Knight and Graehl 2005) and the references therein. There are several tools that can be used for the above-mentioned purpose: tree homomorphisms and tree transducers, studied in formal language theory, and synchronous grammars, proposed by linguists' community.

Also, the accuracy is an important issue in machine translation (MT), so the whole theory should rely on a solid mathematical background. It was stated: (Knight and Graehl 2005), (Knight 2008), (Engelfriet, Lilin and Maletti 2008), that in practice, a good translation model should have at least the following four properties: *expressiveness* (reordering parts of sentences, i.e., local rotation), *inclusiveness* (generalizing the finite state machines by accepting λ-moves), *modularity* (breaking the initial problem into smaller pieces easier to solve) and *teachability* (efficient training). Unfortunately, most of the formalisms proposed until now fail in at least one of the above criteria, but we considered that it deserves mentioning what are the strong and weak parts of each one.

In many cases the translation is too complicated to be performed in just one step, so it is desirable to decompose it into smaller task-oriented parts which are easier to understand, test and train, and from which the original translation is obtained as a composition. Closure under composition is a significant problem in MT (just think what a big success finite state machines had, mainly because of the pipeline or noisy-channel concept: (Mohri 1997), (Knight and Graehl 2005)), but unfortunately, such a property does not hold in general or could not be proved for tree transducers and synchronous grammars: (Engelfriet 1975), (Baker 1979), (Gécseg and Steinby 1984), (Gécseg and Steinby 1997), (Shieber 2004: 95). In this paper we focus on presenting an approach, tree bimorphism, which may help in showing such mathematical properties along with a series of results obtained this way, but we also present some really new achievements that use operational models such as tree transducers.

On the other hand, linguists' society believed back in the 1960s that by using a unique "universal language", the so-called "Interlingua", it will be possible to make high quality translations. Although Bar-Hillel criticized from the beginning this line of research, one may think that a certain modification of the original concept can help us improve the mathematical foundation of MT. At least from a philosophical point of view, it may be useful to construct an abstract language for each two natural languages and not one for all. Using a property like closure under composition, one can switch between each two of them more naturally without loosing precious information about the particularities and nested structures encountered in each natural language.

Extensively studied back in the 1970s and 1980s in the formal language community, the tree bimorphisms can model the Interlingua concept and define tree transformations in an algebraic way. They were used with considerable success in proving properties of various classes of tree transformations (especially, closure under composition): (Takahashi 1972), (Takahashi 1977), (Arnold and Dauchet 1982), (Steinby 1984), (Steinby 1986), Steinby (1990), (Bozapalidis 1992), (Steinby and Tîrnăucă 2007). Moreover by taking the yields of the input trees and output trees, the tree bimorphisms are transformed into word-for-word translation devices.

Stuart M. Shieber was the first one who linked classes of tree transformations of synchronous grammars and tree transducers via tree bimorphisms, in an attempt to extend the mathematical framework of the former devices, where no composition results were known: (Shieber 2004). Following this lead, a series of results was obtained: (Shieber 2006), (Maletti 2007), (Steinby and Tîrnăucă 2007), (Tîrnăucă 2007).

The paper is organized as follows. In Section 2 we describe several mechanisms that define trees and tree transformations: tree homomorphisms, tree transducers and synchronous grammars, while in the next two sections we explain what exactly a tree bimorphism is, how it works, what are the most well-known types and which classes were connected until now with synchronous grammars. We finish by presenting an example and some future work.

2. Ways to define tree transformations

In this section, we give basic definitions and notations regarding trees and tree transformations, which will be used throughout the whole paper. Also, we present the simplest way to define a tree transformation, i.e., as a tree homomorphism, as well as a machine which effectively computes tree transformations, namely tree transducer. Moreover, we briefly mention some of the main types of tree transducers applicable in modelling natural language processes. We finish by describing the concepts of synchronous rewriting and syntax-directed translation, firstly introduced as models of simple compilers, and lately proposed mainly by linguistic community as an attempt to improve the translations between natural languages (especially because of their ability to perform local rotations and describe syntax-sensitive transformations). Moreover, we enumerate several types of well-known synchronous grammars that can be found in literature.

2.1 Trees

The usefulness of trees and tree language theory in areas like linguistics, computer science or formal language theory is well established and we can mention, for example, representation of derivations in formal grammars (syntax), functional programming, machine learning, modelling of RNA sequences, etc. Here, we briefly recall some basic notations and definitions; good introductory books are, at least, (Gécseg and Steinby 1984), (Gécseg and Steinby 1997), (Martín Vide, Mitrana and Păun 2004).

The trees considered are finite, their nodes are labelled by symbols, and the branches leaving any given node have a specified order. A **ranked alphabet** Σ is a finite set of symbols each of them having a given nonnegative integer **arity**, or **rank**. Note that in other works, the same symbol may have different ranks. For any $m \geq 0$, the set of m-ary symbols in Σ, i.e., all symbols of rank m, is denoted Σ_m. In examples we may write $\Sigma = \{f_1 / m_1, \ldots, f_k / m_k\}$ to indicate that Σ consists of the symbols $f_1, \ldots, f_k$ with the respective ranks $m_1, \ldots, m_k$. In addition to ranked alphabets, we use ordinary finite alphabets for labelling leaves of trees, that we call **leaf alphabets**, disjoint from the ranked alphabets. As usual, if X is such an alphabet, X^* denotes the set of all the (finite) words over X, and subsets of X^* are called (string) languages. Moreover, λ denotes the empty word.

For any ranked alphabet Σ and leaf alphabet X, the set $T_\Sigma(X)$ of Σ**-terms with variables in** X is the smallest set T such that $X \cup \Sigma_0 \subseteq T$, and $f(t_1,\ldots,t_m) \in T$ whenever $m > 0$, $f \in \Sigma_m$ and $t_1,\ldots,t_m \in T$. Such terms are regarded as representations of labelled trees, and we call them ΣX **-trees**. Any $d \in X \cup \Sigma_0$ represents a one-node tree in which the only node is labelled with d, and $f(t_1,\ldots,t_m)$ is interpreted as a tree formed by adjoining the m trees represented by $t_1,\ldots,t_m$ to a new f-labelled root. Subsets of $T_\Sigma(X)$ are called ΣX **-tree languages**. We may also speak simply about **trees** and **tree languages** without specifying the alphabets. If the leaf alphabet X is empty, then ΣX-trees are called **ground** ΣX **-trees**, and their set $T_\Sigma(\varnothing)$ is denoted by T_Σ. The symbols of rank 0 are named **constants** or nullary symbols, and the leaf symbols, **variables**.

As the definition of the set $T_\Sigma(X)$ is inductive, notions related to ΣX-trees can be defined recursively, and statements about them can be proved by tree induction. For example, an important notion which will be used later is the yield function that extracts a word from each tree the same way as a sentence is obtained from any of its derivation trees (a concatenated sequence of symbols of the leaves, read from left to right). More formally, the function $\mathrm{yd}: T_\Sigma(X) \to X^*$ and the **yield** $\mathrm{yd}(t)$ of a ΣX-tree t are defined as follows: $\mathrm{yd}(x) = x$ for every variable $x \in X$, $\mathrm{yd}(c) = \lambda$ for every constant $c \in \Sigma_0$, and $\mathrm{yd}(t) = \mathrm{yd}(t_1)\ldots\,\mathrm{yd}(t_m)$ for $m > 0$, $f \in \Sigma_m$ and $t = f(t_1,\ldots,t_m)$. The **yield language** of a ΣX-tree language T is the set $\mathrm{yd}(T) = \{\,\mathrm{yd}(t) \mid t \in T\} (\subseteq X^*)$. The notions of **height** and **subtrees** of a tree t are defined as usually, e.g.: (Gécseg and Steinby 1997: 4).

There are several classes of tree languages that are well known and wide used. We will just mention some of them, formal definitions being found in: (Gécseg and Steinby 1984), (Gécseg and Steinby 1997), (Knight and Graehl 2005)): the class LOC of local tree languages, DREC of deterministically regular tree languages, REC of regular tree languages, and ALG of context-free tree languages (also called algebraic tree languages). It is known that any local tree language is regular, and that $\mathrm{DREC} \subset \mathrm{REC} \subset \mathrm{ALG}$ (Gécseg and Steinby 1984). Perhaps one of the most interesting and applicable connection between tree languages and string languages is that the yield of any regular tree language is a context-free language, and vice versa: (Gécseg and Steinby 1984), (Gécseg and Steinby 1997).

Finally, let us note that in many applications or presentations separate leaf alphabets are not used, but a special set of constants is singled out when needed. Although this often can be done without any loss of generality and it would simplify the exposition somewhat, leaf alphabets are convenient in many cases (e.g., leaf alphabets coincide with the alphabets on which natural languages are defined), and we shall use them in this work.

2.2 Tree Transformations

In the sequel, Σ, Γ and Ω are always ranked alphabets, and X, Y and Z are leaf alphabets.

A tree transformation may be interpreted as a collection of pairs of parse (syntax) trees of natural languages sentences (ranked alphabets may code parts of sentences, e.g., noun or verb phrase, and leaf alphabets usual natural languages alphabets, e.g., Kanji, Spanish or Russian), and its translation as a set of pairs of words from the two natural languages considered (just take the yield of such trees). Formally, a **tree transformation** from $T_\Sigma(X)$ (e.g., Spanish) to $T_\Omega(Y)$ (e.g., English) is any relation $\tau \subseteq T_\Sigma(X) \times T_\Omega(Y)$. The fact that $(s,t) \in \tau$ for some $s \in T_\Sigma(X)$ and $t \in T_\Omega(Y)$ means that τ transforms s (a parse tree of a Spanish sentence) into t (the parse tree of its English translation), and t is then called a **transform of** s. The input alphabets of τ are Σ and X, and the output alphabets Ω and Y. The **translation defined** by τ is the relation $\Lambda(\tau) = \{(\mathrm{yd}(s), \mathrm{yd}(t)) \mid (s,t) \in \tau\} \subseteq X^* \times Y^*$. All the general definitions and properties concerning binary relations apply directly to tree transformations.

Also, if $\rho \subseteq T_\Sigma(X) \times T_\Gamma(Z)$ and $\tau \subseteq T_\Gamma(Z) \times T_\Omega(Y)$ are two tree transformations such that the output alphabets of ρ are the input alphabets of τ, then their composition is the tree transformation

$\rho \circ \tau = \{(s,t) \mid s \in T_\Sigma(X), t \in T_\Omega(Y), \exists r \in T_\Gamma(Z) \text{ such that } (s,r) \in \rho \text{ and } (r,t) \in \tau\}$

from $T_\Sigma(X)$ to $T_\Omega(Y)$. Roughly speaking, if we have a tree transformation from English to Spanish, and one from Spanish to Romanian, by composing them we obtain one from English to Romanian.

The composition operation is extended in a natural way to classes of tree transformations: if C and D are classes of tree transformations, then $C \circ D = \{\rho \circ \tau \mid \rho \in C, \tau \in D\}$ is the class of all tree transformations that are the composition of a transformation from C and a transformation from D. For any classes C, D and E of tree transformations,

- - $C \circ D \subseteq E$ means that any composition of a C-transformation and a D-transformation is an E-transformation,
- - $E \subseteq C \circ D$ means that any E-transformation can be decomposed into the composition of a C-transformation and a D-transformation, and
- - $C \circ C \subseteq C$ means that C is **closed under composition**.

Figure 1: An example of a Romanian-English tree transformation

To familiarize the reader with the notions introduced so far, let us present a short example. Let X be the Romanian alphabet, and Y the English one. Let us take $\Sigma = \{\mathrm{V}/4, \mathrm{N}/3, \mathrm{S}/2, \mathrm{NP}/1, \mathrm{VP}/1\}$ and $\Omega = \{\mathrm{V}/5, \mathrm{N}/4, \mathrm{S}/2, \mathrm{NP}/1, \mathrm{VP}/1\}$. As it can be seen in the Figure 1 bellow, $r = \mathrm{S}(\mathrm{VP}(\mathrm{V}(V,i,n,e)), \mathrm{NP}(\mathrm{N}(I,o,n)))$ is a tree of height 3 in $T_\Sigma(X)$, and $e = \mathrm{S}(\mathrm{NP}(\mathrm{N}(J,h,o,n)), \mathrm{VP}(\mathrm{V}(c,o,m,e,s)))$ is a tree of height 3 in

$T_\Omega(Y)$. Then, $\tau = \{(r,e)\}$ is a tree transformation from $T_\Sigma(X)$ to $T_\Omega(Y)$, i.e., e is a transform of r. Moreover, $\mathrm{yd}(r) = "Vine\ Ion"$, $\mathrm{yd}(e) = "John\ comes"$, and hence "John comes" is a translation (into English) of the Romanian sentence "Vine Ion", i.e. $(\mathrm{yd}(r), \mathrm{yd}(e)) \in \Lambda(\tau)$.

2.3 Tree Homomorphisms

One of the simplest ways to define tree transformations is by a mapping called tree homomorphism which, based on certain rules defined for every input symbol, transforms recursively an input tree into a (totally or not) different output tree. To do this, let us consider $\Xi = \{\xi_1, \xi_2, \xi_3, \ldots\}$ a set of variables disjoint from the other alphabets introduced so far. Moreover, let $\Xi_m = \{\xi_1, \ldots, \xi_m\}$ for each $m \geq 0$. The role of these **"auxiliary" variables** is to indicate an occurrence of a subtree in a tree.

Formally, a **tree homomorphism,** cf.: (Gécseg and Steinby 1984), (Gécseg and Steinby 1997), (Comon et all 1997), for example, $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ is determined by a mapping $\varphi_X : X \to T_\Omega(Y)$ and mappings $\varphi_m : \Sigma_m \to T_\Omega(Y \cup \Xi_m)$, for all $m \geq 0$ such that $\Sigma_m \neq \varnothing$, as follows:

- $\varphi(x) = \varphi_X(x)$ for any variable $x \in X$,
- $\varphi(c) = \varphi_0(c)$ for any constant $c \in \Sigma_0$, and
- $\varphi(t) = \varphi_m(f)(\xi_1 \leftarrow \varphi(t_1), \ldots, \xi_m \leftarrow \varphi(t_m))$ for $t = f(t_1, \ldots, t_m)$ $(m > 0)$.

In other words, such a mapping is processing a tree t starting from his root and going to the leaves, by applying the rule corresponding to the current symbol: if it is a symbol f with rank at least 1, it replaces such a symbol from the input tree by a tree $\varphi_m(f)$ in which auxiliary variables appear as leaf symbols. Next, each such ξ_i will be replaced by the corresponding independently processed subtree $\varphi(t_i)$ of the input t (observe that this is done inductively repeating the same procedure and using the rules of φ, and that each ξ_i uniquely corresponds to a subtree of t). It recursively continues until we reach the symbols of t from the leaves: because they do not have subtrees, each constant c will be replaced by a simple tree $\varphi_0(c)$ from the output set of trees $T_\Omega(Y)$, and each leaf symbol x in X is replaced by the corresponding tree $\varphi_X(x)$. Let us note that any tree homomorphism $\varphi : T_\Sigma(X) \to T_\Omega(Y)$ defines a tree transformation $\tau_\varphi = \{(s, \varphi(s)) \mid s \in T_\Sigma(X)\}$; usually one just identifies τ_φ with φ treating φ as a relation.

In the literature there are several types of tree homomorphisms depending on the restrictions imposed on each tree $\varphi_m(f)$ $(m \geq 0, f \in \Sigma)$ and implicitly on auxiliary variables ξ (see the bellow references for complete formal definitions):

- **linear** – if in each $\varphi_m(f)$ no copy of a subtree is allowed: (Arnold and Dauchet 1982), (Gécseg and Steinby 1984), (Gécseg and Steinby 1997);

- **non-deleting,** or complete – if in each $\varphi_m(f)$ no subtree information is lost during the processing: (Gécseg and Steinby 1984), (Gécseg and Steinby 1997), (Arnold and Dauchet 1982);
- **strict**, or ε-free – if no $\varphi_m(f)$ can be reduced to an auxiliary variable ξ_i: (Arnold and Dauchet 1982), (Steinby 1986), (Comon et all 1997);
- **symbol-to-symbol** – each leaf variable or constant is mapped to another leaf variable or constant, respectively, and moreover each $\varphi_m(f)$ is a tree of height 1 in which possible some of the subtrees appear in a different order and possible some of the subtrees were erased: (Comon et all 1997);
- **alphabetic** – if it is linear, each variable is mapped into a variable, and each $\varphi_m(f)$ is replaced by one of the subtrees ξ_i, or is a tree in which possible some of the subtrees appear in a different order and possible some of the subtrees were erased: (Bozapalidis 1992);
- **quasi-alphabetic** – if it is linear, non-deleting, strict, each constant is mapped to a constant or a tree of height 1 (all leaves are symbols from the output leaf alphabet), and each $\varphi_m(f)$ is a tree of height 1 in which the subtrees are possibly reorder via a permutation and supplementary, symbols from the output leaf alphabet may appear as leaves: (Steinby and Tîrnăucă 2007);
- **relabeling** – each input tree is transformed into a tree of exactly same shape but the label of each node may by replaced with a symbol of the same rank: (Engelfriet 1975).

Often tree homomorphisms of the form $\varphi: T_\Sigma \to T_\Omega(Y)$ or $\varphi: T_\Sigma \to T_\Omega$ are considered, and then the mapping φ_X and any conditions concerning it can be ignored. We denote by lH, nH, sH, stsH, aH, qH and rH the classes of all linear, non-deleting, strict, symbol-to-symbol, alphabetic, quasi-alphabetic and strictly alphabetic tree homomorphisms, respectively. Further subclasses of tree homomorphisms can be obtained by combining any of these restrictions. For example, lnH is the class of all linear non-deleting tree homomorphisms.

Now let us see an example. Let $\Sigma = \{\mathrm{NP}/4, \mathrm{VP}/2, \mathrm{N}/0\}$, $\Omega = \{\mathrm{NP}/6, \mathrm{V}/3, \mathrm{N}/2, \mathrm{D}/0\}$, $X = \{i\}$ and $Y = \{\hat{\imath}, æ\}$. Let us define the three tree homomorphisms $\varphi, \psi, \eta: T_\Sigma(X) \to T_\Omega(Y)$ as follows:

- $\varphi_X(i) = \hat{\imath}$, $\varphi_4(\mathrm{NP}) = \mathrm{NP}(æ, \xi_4, \xi_1, \xi_3, \hat{\imath}, \xi_2)$, $\varphi_2(\mathrm{VP}) = \mathrm{V}(\xi_1, \xi_2, \hat{\imath})$, $\varphi_0(\mathrm{N}) = \mathrm{N}(æ, \hat{\imath})$
- $\psi_X(i) = æ$, $\psi_4(\mathrm{NP}) = \mathrm{V}(\xi_3, \xi_1, \xi_3)$, $\psi_2(\mathrm{VP}) = \mathrm{NP}(\xi_1, \xi_1, \xi_2, \xi_1, \xi_2, \xi_1)$, $\psi_0(\mathrm{N}) = \mathrm{D}$
- $\eta_X(i) = \hat{\imath}$, $\eta_4(\mathrm{NP}) = \xi_3$, $\eta_2(\mathrm{VP}) = \mathrm{N}(\xi_2, \xi_1)$, $\eta_0(\mathrm{N}) = \mathrm{D}$.

Then φ is quasi-alphabetic, ψ symbol-to-symbol (and not linear), and η alphabetic. If we take the tree $t = \mathrm{NP}(i, \mathrm{VP}(\mathrm{N}, i), \mathrm{N}, i)$ in $T_\Sigma(X)$, then $\varphi(t) = \mathrm{NP}(æ, \hat{\imath}, \hat{\imath}, \mathrm{N}(æ, \hat{\imath}), \hat{\imath}, \mathrm{V}(\mathrm{N}(æ, \hat{\imath}), \hat{\imath}, \hat{\imath}))$, $\psi(t) = \mathrm{V}(\mathrm{D}, \hat{\imath}, \mathrm{D})$ and $\eta(t) = \mathrm{D}$ (see Figure 2).

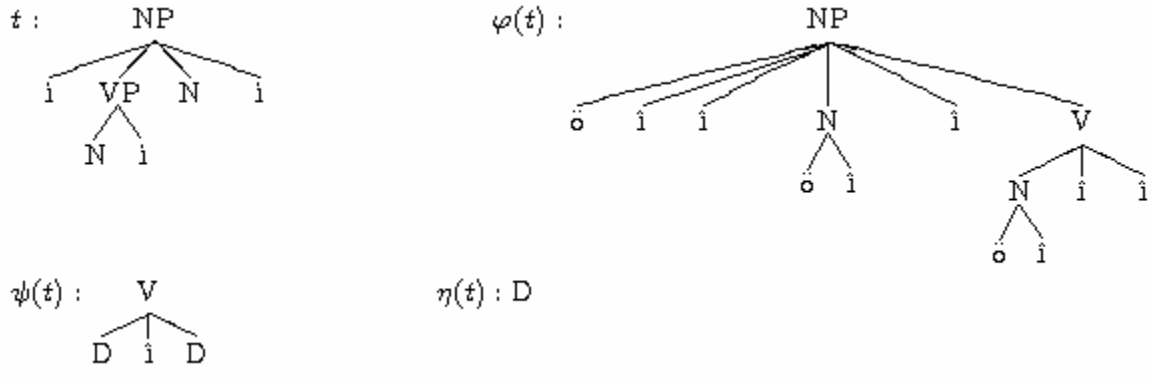

Figure 2: Examples of tree homomorphisms

2.4 Tree transducers

Independently introduced in formal language theory by: (Rounds 1970) and (Thatcher 1970), and motivated by problems in natural language processing (see: (Rounds 1970: 257), (Knight and Graehl 2007: 3)), the tree transducer is a machine that generalizes usual finite state automata and transducers, and effectively computes a tree transformation: given an input tree over the input ranked and leaf alphabets, it computes an output tree over the output ranked and leaf alphabets. Because they do not represent the core of the present exposition, we will not give formal definitions; instead, we present in few words the two basic types of tree transducers, top-down and bottom-up, and their main characteristics, and we just list some other kinds of such devices. For details, see the references below.

As already mentioned, there are two main categories of tree transducers, depending on how the input tree is being processed: the machine starts at the root and moves towards the frontier formed by the leaves (that is why, it is called **top-down** or **root-to-frontier tree transducer**), or it starts at the leaves and goes up to finish at the root (named **bottom-up** or **frontier-to-root tree transducer**). Because of this way of moving, we can distinguish immediately the two main properties of top-down tree transducers: during a computation, a subtree of the input tree may be deleted before processing, or it can be multiplied and then processed in different modes. By contrary, during a computation in a bottom-up tree transducer, a subtree of the input tree cannot be deleted before processing, and it is first processed and then the (only) result may be multiplied.

Depending on the restrictions imposed on the rules that define the processing of the input tree, and the asymmetry between top-down and bottom-up directions, there are several types of tree transducers, most of them models of syntax-directed semantics: total, deterministic, linear, non-deleting, with regular look-ahead, (monadic) macro, attributed, modular (see: (Engelfriet 1975), (Gécseg and Steinby 1984), (Gécseg and Steinby 1997: 58)), etc.

Three special models deserve maybe even a further investigation because of their strongly linguistic motivation (machine translation): **extended top-down tree transducers**: (Knight and Graehl 2005), (Knight 2008), also called transducteurs généralisés descendants: (Arnold and Dauchet 1976), **multi bottom-up tree transducers**: (Fülöp, Kühnemann and Vogler 2005), (Maletti 2007) and **extended multi bottom-up tree transducers**: (Engelfriet, Lilin and Maletti 2008), also called S-transducteurs ascendants generalices: (Lilin 1981). The one proposed by Knight and Graehl was introduced as an attempt to perform very complicate rotations (multi-level) during the translation process, but it also has the inclusiveness and the trainability (unfortunately no closure under composition). The second one has modularity and

efficiency, but fails on inclusiveness (still no result is known about their trainability). The best candidate for model natural language translations seems to be the last one because it has efficiency, modularity and inclusiveness, and most likely trainability.

2.5 Synchronous grammars

Synchronous grammars use the idea of synchronous rewriting: two formal grammars (used to model the syntax of the natural language sentences, for example) work in parallel, productions being linked by some relation and applied synchronously. This way, pairs of recursively related words (sentences) are generated simultaneously.

The first implemented model of synchronous rewriting is the **syntax-directed translator** (and its corresponding **syntax-directed translation**), originally introduced as a simple model of a compiler: (Irons 1961). There are two context-free grammars (one for the input language and one for the output language) with productions paired by a permutation via nonterminal symbols. An example of a production in a syntax-directed translator is $S \rightarrow NP_1 V_2 \, on \, NP_3 ; a \, NP_3 V_2 \, the \, NP_1$, where the indexes tell us how the nonterminals are related (e.g., the first nonterminal NP from the output part has the index 3 which means that it is related with the third nonterminal from the input part). In other words, a syntax-directed translation can be viewed as a three-step process one of which is a tree transformation: for a given input sentence u, construct a parse tree for u, transform the parse tree into a tree in the output grammar, and take the yield of the output tree as a translation for u. So, they can easily capture syntax-sensitive transformations.

Because of the pairing between applied productions, syntax-directed translators can perform certain types of reordering of parts of sentences that usual finite state machine designed for word-for-word translations cannot. Hence, they were a suitable tool used with considerable success in syntax-based machine translation and also for doing semantic interpretation as was stated in: (Chiang 2006).

Having such a generous potential to explore, many types of syntax-directed translation devices were introduced in the literature, from the late 1960s until present, and we should mention at least: **syntax-directed translation schemata**: (Aho and Ullman 1972), **synchronous context-free grammars**: (Satta and Peserico 2005), **inversion transduction grammars**: (Wu 1997), **multitext grammars**: (Melamed 2003), **tree-to-string models**: (Yamada and Knight 2001), (Galley et al 2004), **hierarchical phrase-based models**: (Chiang 2007), etc. A good, short, not very formal survey on most of the syntax-directed translators along with examples and comparisons between them is: (Chiang 2006).

Unfortunately, such devices could not do every kind of local rotation that may appear between natural languages so, more powerful types of synchronous grammars were called for: **synchronous tree substitution grammars**: (Eisner 2003), (Shieber 2004), and **syntax-directed translation with extended domain of locality**: (Huang, Knight and Joshi 2007). These two formalisms allow multilevel rules, that is to say, they can generate parse trees with very different structures and shapes. Moreover, there were proposed even more powerful types, beyond the generative capacity of context-free rewriting, the ones that involve discontinuous constituents: **generalized multitext grammars**: (Melamed, Satta and

Wellington 2004) and **synchronous tree adjoining grammars**: (Shieber and Schabes 1990), (Shieber 1994).

To finish this short "trip" between synchronous formalisms, we have to remark that the mathematical framework provided by synchronous grammars is quite poor as Shieber himself mentioned in: (Shieber 2004: 95): "...the **bimorphism characterization** of tree transducers has led to a series of composition closure results. Similar techniques may now be applicable to *synchronous formalisms, where no composition results are known*..."

3. What is a tree bimorphism?

As was stated before, a tree bimorphism is a formal algebraic model to describe classes of tree transformations in contrast with the dynamic view of tree transducers, and used with considerable success in proving mathematical properties (e.g., closure under composition) of such classes. It consists of two tree homomorphisms defined on the same (regular) tree language, which work in parallel. Moreover, by imposing suitable restrictions on the tree language or the tree homomorphisms, one can get classes of tree bimorphisms with special properties that may be useful for applications in linguistics as we will see in Section 4. In what follows, we give the formal definition and a (not necessarily complete) overview of the most well-known types of tree transformations defined in terms of tree bimorphisms.

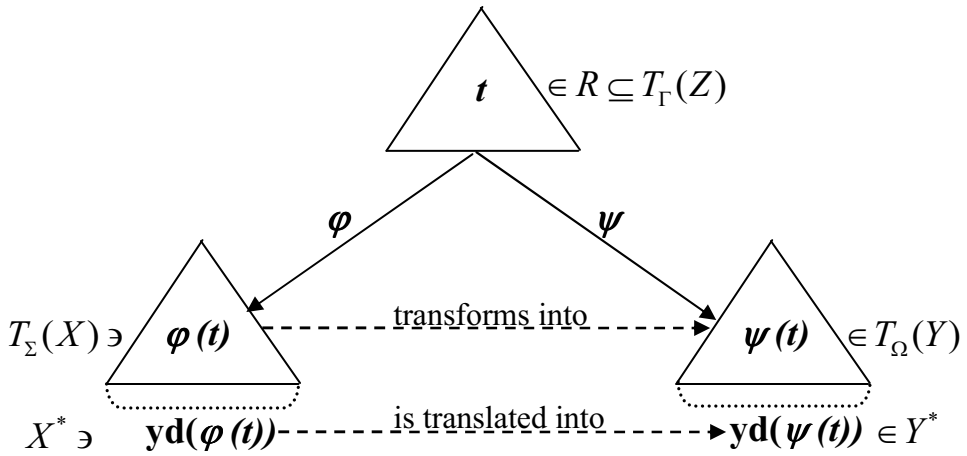

Figure 3: The scheme of a tree bimorphism

Formally, a **tree bimorphism**, cf. (Arnold and Dauchet 1982), (Gécseg and Steinby 1984), (Gécseg and Steinby 1997), for example, is a triple $B = (\varphi, R, \psi)$, where $R \subseteq T_\Gamma(Z)$ is a regular tree language, and $\varphi : T_\Gamma(Z) \to T_\Sigma(X)$ and $\psi : T_\Gamma(Z) \to T_\Omega(Y)$ are tree homomorphisms. The **tree transformation defined by** B is the relation $\tau_B = \{(\varphi(r), \psi(r)) \mid r \in R\} \subseteq T_\Sigma(X) \times T_\Omega(Y)$, and the **translation defined** by B is the relation $\mathrm{yd}(\tau_B) = \{(\mathrm{yd}(r\varphi), \mathrm{yd}(r\psi)) \mid r \in R\} \subseteq X^* \times Y^*$ (see Figure 3).

For any classes H_1 and H_2 of tree homomorphisms and any class R of regular tree languages, we denote by $\mathrm{B}(\mathrm{H}_1, \mathrm{R}, \mathrm{H}_2)$ the class of all tree bimorphisms $B = (\varphi, R, \psi)$ with $\varphi \in \mathrm{H}_1$, $R \in \mathrm{R}$ and $\psi \in \mathrm{H}_2$, and by $B(\mathrm{H}_1, \mathrm{R}, \mathrm{H}_2)$ the corresponding class of tree

transformations. For example, $\mathrm{B}(\mathrm{lnH}, \mathrm{LOC}, \mathrm{rH})$ is the class of tree bimorphisms in which the first tree homomorphism component is linear and non-deleting, the second is a relabeling and the tree language is local, and $B(\mathrm{lnH}, \mathrm{LOC}, \mathrm{rH})$ is the class of all the tree transformations defined by such tree bimorphisms.

We finish this section by browsing from the literature some classes of tree transformations defined by tree bimorphisms. The exposition is informal but requires additional information about term algebra and formal language theory (see: (Gécseg and Steinby 1984) and the references therein). Notice that in all cases the tree bimorphism approach was essential in proving properties like closure under composition or preservation of regular tree languages!

One of these classes, called **primitive transformations** and denoted by PT: (Takahashi 1972), is an extension from words to trees of the length-preserving finite state transductions introduced by: (Elgot and Mezei 1965), and is characterized in terms of tree bimorphisms as a relation between two projective images of one recognizable set. Further generalizations are presented in the same paper: the description in terms of bimorphisms of a more general class of tree transformations, namely **primitive transformation with permutation** (PTP), and the characterization of regular sets in terms of inverse images by projections. Both classes, PT and PTP, are closed under composition.

To extend Nivat's results about rational transductions, another class of tree transformations called **rational relations of binary trees** (a binary tree is a tree with all interior nodes labelled by two-rank symbols) is introduced in: (Takahashi 1977). They are characterized by using recognizable sets of binary trees and two linear non-deleting tree homomorphisms - called tree-morphisms there. The main results presented are: the class of rational relations is closed under composition, and the recognizable sets of binary trees are preserved not only by tree-morphisms but also by their inverses.

Following the same line of research, (Arnold and Dauchet 1976) presented the class BI of tree bimorphisms where the two tree homomorphisms are linear, non-deleting and strict, and the tree language regular. Although this class is not closed under composition, the new class $\mathrm{BI} \circ \mathrm{BI}$ is. Other classes of tree transformations are introduced and explained in: (Arnold and Dauchet 1976), (Arnold and Dauchet 1982), together with their connection with BI. In particular, one of these classes is TT which contains usual tree transformations defined by a tree transducer and their inverses. Furthermore, it is characterized in terms of tree bimorphisms, i.e., $\mathrm{TT} = \mathrm{BI} \circ \mathrm{BI}$, and hence it is easier to manipulate. Moreover, TT is closed under composition and preserves regular tree languages.

In: (Steinby 1986) the author offers a more directly algebraic approach to tree transformations by introducing **Σ-rational tree transformations** (Σ-RTTs) and **Σ-algebraic tree transformations** (Σ-ATTs). The Σ-RTTs are rational subsets of the direct product of two finitely generated term algebras $T_\Sigma(X)$ and $T_\Sigma(Y)$. Their class is denoted $\mathrm{Rat}_\Sigma(X, Y)$ and resembles Nivat's transductions in many ways: Σ-RTTs are defined by certain tree bimorphisms (called Σ-rational tree bimorphisms); they preserve the recognizability and the rationality of tree languages; the translations defined by Σ-RTTs are exactly the rational transductions; the converse of a Σ-RTT is Σ-rational, and so is the composition of any two of them. Also, Σ-RTTs are different from rational transductions,

because for example, they are always locally finite and their class is closed under intersection. Corresponding to algebraic transductions: (Mezei and Wright 1967), Σ-ATTs are defined as the algebraic subsets of the direct product of two finitely generated term algebras $T_\Sigma(X)$ and $T_\Sigma(Y)$, and their class is denoted by $\mathrm{Alg}_\Sigma(X,Y)$. In the representation of Σ-ATTs by means of tree bimorphisms, rational tree languages are now replaced by regular tree languages. Using this, many results for Σ-RTTs are extended to Σ-ATTs. In particular, contrary to the case of algebraic transductions, $\mathrm{Alg}_\Sigma(X,Y)$ is closed both under composition and under intersection. The translations defined by Σ-ATTs are exactly the algebraic transductions.

Moreover in: (Steinby 1984), it is shown that many basic problems, undecidable for rational transductions, are decidable for Σ-RTTs and Σ-ATTs when these tree transformations are defined by tree bimorphisms. In particular, the equivalence problem is decidable for them. Because of the close connection with the rational and algebraic transductions, they could model the correction transformations required by local and structural errors in tree representations of patterns: (Steinby 1990). To conclude, we mention that Σ-RTTs and Σ-ATTs have an obvious limitation: the input trees and their transforms are always over the same ranked alphabet.

The class of **alphabetic tree relations**: (Bozapalidis 1992), denoted by Alph, is the class of tree transformations defined by tree bimhorphisms where the two tree components are alphabetic tree homomorphisms (called “démarquages linéairs” in: (Arnold and Dauchet 1982)), and the tree language is regular (or local). It was shown that Alph is closed under composition and inverses, preserves regular and algebraic tree languages, and contains most of the classical tree transformations (top-catenation, branches, subtrees, union and intersection with a regular tree language, etc.), being so far the best candidate for building the AFL theory for trees.

In: (Steinby and Tîrnăucă 2007), the notion of quasi-alphabetic tree homomorphism and the new class of **quasi-alphabetic tree bimorphisms** are introduced. This class, denoted by $\mathrm{B(qH, LOC, qH)}$, consists of all tree bimorphisms in which the tree language is local and the two tree homomorphism components are quasi-alphabetic. It is closed under composition and inverses, and preserves regular tree languages.

The connection between some of the classes introduced above is immediate. Since rational tree languages are regular and morphisms of term algebras are special linear non-deleting strict homomorphisms, the class $\mathrm{Rat}_\Sigma(X,Y)$ forms a proper subclass of BI; a natural generalization of the rational relations of Takahashi also yields the class BI. The classes PT and PTP are incomparable with $\mathrm{Rat}_\Sigma(X,Y)$ since primitive transformations may relabel nodes and permute subtrees. We finish by mentioning that several classes of tree transformations described in terms of tree bimorphisms are similar with the ones computed by tree transducers. For example, every Σ-RTT can be defined by a linear bottom-up tree transducer. Another nice result says that the tree transformations computed by bottom-up tree transducers are equivalent with the ones defined by tree bimorphisms where the first component is a linear non-deleting strict tree homomorphism: (Comon et all 1997), (Engelfriet 1975). On the other hand, alphabetic tree relations are incomparable with the

classes of tree transformations defined by top-down and bottom-up tree transducers: (Bozapalids 1992).

4. What can we obtain with the help of tree bimorphisms?

This section mainly presents closure/no closure under composition results of various types of classes of tree transformations defined by synchronous grammars and tree transducers that were obtained using the tree bimorphism formalism. With such a wide range of models, there still are many others to explore. The exposition is extremely brief; for details and complete proofs see the references below.

In: (Shieber 2004), it was shown that synchronous tree substitution grammars are as powerful as $B(lnH, REC, lnH)$, denoted there $B(LC, LC)$, i.e., with tree bimorphisms in which the two tree homomorphism are linear and non-deleting, and the tree language regular. On the other hand in: (Maletti 2007), $B(lnH, REC, lnH)$ are proved to be equivalent with linear and non-deleting extended tree transducers of (Knight and Graehl 2005). Using the result in: (Maletti et al 2007) that linear non-deleting extended tree transducers are not closed under composition and the observations in: (Arnold and Dauchet 1982), we get immediately that linear non-deleting synchronous tree substitution grammars and $B(lnH, REC, lnH)$ are not closed under composition as well.

In: (Shieber 2006), we find out that synchronous tree adjoining grammars are equal in power with linear non-deleting embedded tree transducers which usually are called linear non-deleting monadic macro tree transducers (see Section 2.4 for further references about macro tree transducers). Following the same idea as in: (Shieber 2004), it is straightforward to check that synchronous tree adjoining grammars are exactly the same as $B(lnsH, REC, lnsH)$, denoted by $B(ELC, ELC)$ there, about which we know from: (Arnold and Dauchet 1976), (Arnold and Dauchet 1982) that it is not closed under composition.

In: (Steinby and Tîrnăucă 2007) and (Tîrnăucă 2007), it is shown that the translations defined by the quasi-alphabetic tree bimorphisms $B(qH, LOC, qH)$ are effectively equal to the ones defined by syntax-directed translation schemata of: (Aho and Ullman 1972) and by synchronous context-free grammars of: (Satta and Peserico 2005), respectively.

5. An extended example

In this section, we show a very simple example of how to construct a syntax-directed translation schema, how the closure under composition of classes of tree transformations may help in improving the quality of translations, and also how a concrete tree bimorphism looks like. For the sake of simplicity, in the construction of the following parse trees, the symbols will not have a single rank (e.g., V may have arity 3 or 5 depending how many branches are leaving the node labelled with V), and hence we omit writing their rankings.

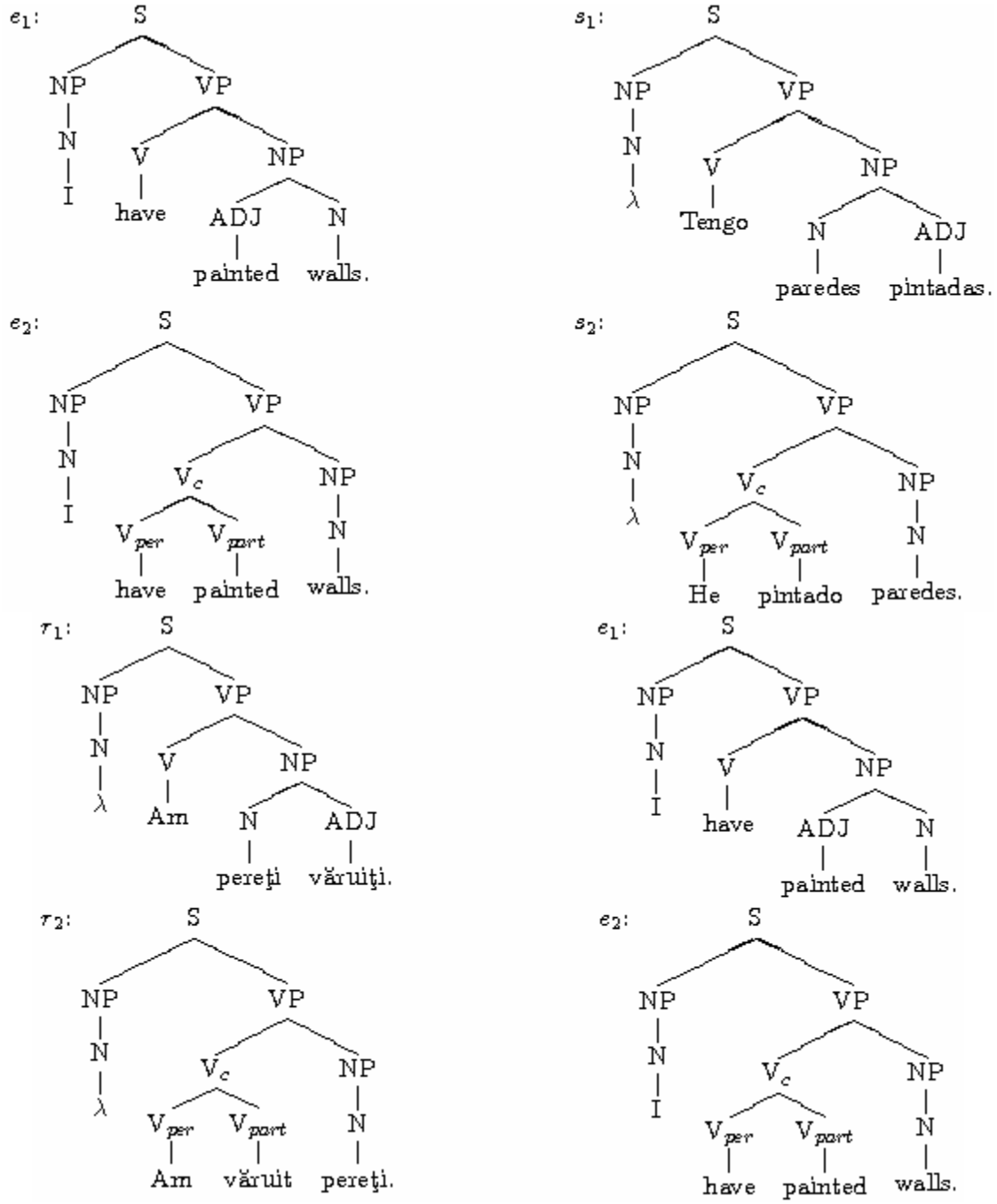

Figure 4: The two English-Spanish and Romanian-English tree transformations

To begin with, let us consider the following ranked and leaf alphabets: $\Sigma = \Omega = \Gamma = \{\text{S, NP, VP, V, N, ADJ, } \text{V}_\text{c}, \text{V}_\text{part}, \text{V}_\text{pers}\}$, X is the Romanian alphabet, Y the English one and Z the Spanish. Let us take in English the sentence "I have painted walls". There are two parse trees e_1 and e_2 in $T_\Omega(Y)$, as we can see in Figure 4, depending on how "painted" is interpreted: as an adjective or verb participle. When translated into Spanish, the two possible interpretations of this particular sentence produce the following Spanish sentences: "Tengo paredes pintadas" (with the parse tree s_1 in $T_\Gamma(Z)$ corresponding to e_1) and "He pintado paredes" (with the parse tree s_2 in $T_\Gamma(Z)$ corresponding to e_2). In a similar manner, if we consider the Romanian into English translation, we obtain two pairs of

parse trees (r_1, e_1) and (r_2, e_2) corresponding to the translations of "Am pereți văruiți" and "Am văruit pereți" into "I have painted walls" (see again Figure 4).

A syntax-directed translation schema T_1 that generates (e_1, s_1) and (e_2, s_2) contains the rules

$S \to NP_1\ VP_2; NP_1\ VP_2$ $\qquad$ $V_c \to V_{per1}\ V_{part2}; V_{per1}\ V_{part2}$

$NP \to N_1; N_1$ $\qquad$ $NP \to ADJ_1\ N_2; N_2\ ADJ_1$

$VP \to V_1\ NP_2; V_1\ NP_2$ $\qquad$ $VP \to V_{c1}\ NP_2; V_{c1}\ NP_2$

$N \to I; \lambda$ $\qquad$ $N \to walls.; paredes.$

$ADJ \to painted; pintadas.$ $\qquad$ $V_{per} \to have; He$

$V \to have; Tengo$ $\qquad$ $V_{part} \to painted; pintado$,

and a syntax-directed translation schema T_2 produces (r_1, e_1) and (r_2, e_2) by the rules

$S \to NP_1\ VP_2; NP_1\ VP_2$ $\qquad$ $V_c \to V_{per1}\ V_{part2}; V_{per1}\ V_{part2}$

$NP \to N_1; N_1$ $\qquad$ $NP \to ADJ_1\ N_2; N_2\ ADJ_1$

$VP \to V_1\ NP_2; V_1\ NP_2$ $\qquad$ $VP \to V_{c1}\ NP_2; V_{c1}\ NP_2$

$N \to I; \lambda$ $\qquad$ $N \to pereți; walls.$

$ADJ \to văruiți; painted.$ $\qquad$ $V_{per} \to Am; have$

$V \to Am; have$ $\qquad$ $V_{part} \to văruit;\ painted$

Now, let us construct a (quasi-alphabetic) tree bimorphism that defines the tree transformation $\{(e_1, s_1)\}$ (via a renaming of the symbols in order to get unique ranks). To do this, we can take $\Delta = \{p_1/2, p_2/1, p_3/2, p_4/0, p_5/0, p_6/2, p_7/0, p_8/0\}$ and the regular (in fact, local) tree language $R = \{p_1(p_2(p_4), p_3(p_5, (p_6(p_7, p_8))))\} \subseteq T_\Delta$ (see Figure 5).

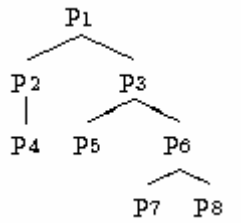

Figure 5: A tree in T_Δ

We consider the input ranked alphabet $\Theta = \{\text{ADJ}/7, \overline{\text{N}}/5, \text{V}/4, \text{S}/2, \text{VP}/2, \overline{\text{VP}}/2, \text{N}/1, \text{NP}/1\}$, and the output ranked alphabet $\Psi = \{\text{ADJ}/8, \overline{\text{N}}/7, \text{V}/5, \text{S}/2, \text{VP}/2, \overline{\text{VP}}/2, \text{N}/1, \text{NP}/1\}$. The input and output leaf alphabets are Y (English) and $Z \cup \{\lambda\}$ (Spanish). If we define two (quasi-alphabetic) tree homomorphisms $\varphi : T_\Delta \to T_\Theta(Y)$ and $\psi : T_\Delta \to T_\Phi(Z)$ by

$$\varphi_2(p_1) = S(\zeta_1, \zeta_2) \qquad \psi_2(p_1) = S(\zeta_1, \zeta_2)$$
$$\varphi_1(p_2) = NP(\zeta_1) \qquad \psi_1(p_2) = NP(\zeta_1)$$
$$\varphi_2(p_3) = VP(\zeta_1, \zeta_2) \qquad \psi_2(p_3) = VP(\zeta_1, \zeta_2)$$
$$\varphi_0(p_4) = N(I) \qquad \psi_0(p_4) = N(\lambda)$$
$$\varphi_0(p_5) = V(h,a,v,e) \qquad \psi_0(p_5) = V(T,e,n,g,o)$$
$$\varphi_2(p_6) = \overline{VP}(\zeta_1, \zeta_2) \qquad \psi_2(p_6) = \overline{VP}(\zeta_2, \zeta_1)$$
$$\varphi_0(p_7) = ADJ(p,a,i,n,t,e,d) \qquad \psi_0(p_7) = ADJ(p,i,n,t,a,d,a,s)$$
$$\varphi_0(p_8) = \overline{N}(w,a,l,l,s) \qquad \psi_0(p_8) = \overline{N}(p,a,r,e,d,e,s),$$

it is easy to see that $\tau_B = \{(e_1, s_1)\}$ (the construction is in : (Steinby and Tîrnăucă 2007)) .

Because of the closure under composition of such tree transformations, we can get a direct and correct translation from Romanian to Spanish. We obtain only the pairs (r_1, s_1) and (r_2, s_2), and hence the translations “Am pereți văruiți.” into “Tengo paredes pintadas.”, and “Am văruit pereți” into “He pintado paredes”. To conclude, note that wrong translations like (“Am pereți văruiți.”, “He pintado paredes”) are eliminated because the closure refers to trees and not to their yields.

6. Conclusions and future work

We believe that the results presented in the previous sections may deserve some further investigation. From a theoretical point of view, one can try to prove other (closure) properties that may improve the translation process, e.g., intersection and union, preservation of regularity of tree languages. Also from a practical point of view, it may be interesting to implement the connections presented above, and compare the results with other systems developed so far, or to see what other applicable synchronous formalism can be described in terms of tree bimorphisms.

7. Acknowledgements

This work was supported by the scholarship 2007BRDI/06-11 provided by Rovira i Virgili University. Also, I am very grateful to Andreas Maletti, Gemma Bel Enguix, Alexander Perekrestenko and my beloved wife Cristina Tîrnăucă for fruitful discussions.

8. Bibliography

Aho, Alfred V. and Jeffrey D. Ullman (1972). *The theory of parsing, translation, and compiling, Volume I: Parsing*. New Jersey: Prentice Hall Professional Technical Reference.

Arnold, André and Max Dauchet (1976). “Bi-transductions de forêts”. In: S. Michaelson and Robin Milner, eds., *Third International Colloquium on Automata, Languages and Programming*, *University of Edinburgh, July 20-23, 1976*. Edinburgh: Edinburgh University Press, pp. 74–86.

Arnold, André and Max Dauchet (1982). “Morphismes et bimorphismes d’arbres”, *Theoretical Computer Science* 20: 33–93.

Baker, Brenda S. (1979). “Composition of top-down and bottom-up tree transductions”, *Information and Control* 41(2): 186–213.

Bozapalidis, Simeon (1992). “Alphabetic tree relations”, *Theoretical Computer Science* 99(2): 177–211.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra and Robert L. Mercer (1993). “The mathematics of statistical machine translation: Parameter estimation”, *Computational Linguistics* 19(2): 263–312.

Chiang, David (2006). “An introduction to synchronous grammars”. *Manuscript.* URL: http://www.isi.edu/~chiang/papers/synchtut.pdf.

Chiang, David (2007). “Hierarchical phrase-based translation”, *Computational Linguistics* 33(2):201–228.

Comon, Hubert, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison and Marc Tommasi (1997). *Tree automata techniques and applications*. URL: http://www.grappa.univ-lille3.fr/tata.

Eisner, Jason (2003). “Learning non-isomorphic tree mappings for machine translation”. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - 2, Sapporo, Japan, July 07 - 12, 2003*, Morristown: Association of Computational Linguistics, pp. 205-208.

Elgot, Calvin C. and Jorge E. Mezei (1965). “On relations defined by generalized finite automata”, *IBM Journal of Research and Development* 9: 47-68.

Engelfriet, Joost (1975). “Bottom-up and top-down tree transformations – a comparison”, *Mathematical Systems Theory* 9(3): 198–231.

Engelfriet, Joost, Eric Lilin and Andreas Maletti (2008). “Extended multi bottom-up tree transducers”. *Manuscript.* URL: http://wwwtcs.inf.tu-dresden.de/~maletti/pub/engmal08.pdf.

Fülöp, Zoltán, Armin Kühnemann and Heiko Vogler (2005). “Linear deterministic multi bottom-up tree transducers”, *Theoretical Computer Science* 347 (1-2): 276-287.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu (2004). “What's in a translation rule?” In: *HLT-NAACL 2004, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, May 2-7, Boston, MA*, pp. 273-280.

Gécseg, Ferencz and Magnus Steinby (1984). *Tree automata*. Budapest: Akadémiai Kiadó.

Gécseg, Ferencz and Magnus Steinby (1997). “Tree languages”. In: Arto Salomaa and Grzegorz Rozenberg, eds., *Handbook of Formal Languages: Volume 3. Beyond Words*. Berlin: Springer, pp. 1–68.

Huang, Liang, Kevin Knight and Aravind Joshi (2006). “Statistical syntax-directed translation with extended domain of locality”. In: *Proceedings of the AMTA 2006, Boston, Cambridge, Massachusetts, August 8-12, 2006*. URL: http://www.mt-archive.info/AMTA-2006-Huang.pdf.

Irons, E (1961). “A syntax directed compiler for ALGOL 60”, *Communications of the ACM* 4(1): 51–55.

Knight, Kevin (2008). “Capturing practical natural language transformations”. *Manuscript*. URL: http://www.isi.edu/natural-language/mt/capturing.pdf.

Knight, Kevin and Yaser Al-Onaizan (1998). “Translation with finite-state devices”. In: David Farwell, Laurie Gerber and Eduard H. Hovy, eds., *Machine Translation and the Information Soup, Third Conference of the Association for Machine Translation in the Americas, AMTA ’98, Langhorne, PA, USA, October 28-31, 1998. Proceedings*, *vol. 1529 of LNCS*. London: Springer-Verlag, pp. 421–437.

Knight, Kevin and Jonathan Graehl (2005). “An overview of probabilistic tree transducers for natural language processing”. In: Alexander F. Gelbukh, ed., *Computational Linguistics and Intelligent Text Processing 6th International Conference, CICLing 2005, Mexico City, Mexico, February 13-19, 2005, Proceedings, vol. 3406 of LNCS*. Berlin: Springer-Verlag, pp. 1–24.

Lilin, Eric (1981). “Propriétés de clôture d’une extension de transducteurs d’arbres déterministes”. In: Egidio Astesiano and Corrado Böhm eds., *CAAP ‘81, Trees in Algebra and Programming, 6th Colloquium, Genoa, Italy, March 5-7, 1981, Proceedings. Vol. 112 of LNCS.* Berlin: Springer, pp: 280-289.

Maletti, Andreas (2007). “Compositions of extended top-down tree transducers”. In: Remco Loos, Szilárd Z. Fazekas and Carlos Martín Vide, eds., *Proceedings of the 1st International Conference on Language and Automata Theory and Applications, LATA 2007, March 29-April 4, 2007, vol. 35/07 of GRLMC Reports*. Tarragona: Universitat Rovira i Virgili, pp. 379–390.

Maletti, Andreas, Jonathan Graehl, Mark Hopkins and Kevin Knight (2007). “On extended tree transducers”. *Submitted to SIAM Journal on Computing*.

Martín Vide, Carlos, Victor Mitrana and Gheorghe Păun (2004). *Formal languages and applications, Studies in fuzziness and soft computing, Vol. 148*. Berlin: Springer.

Melamed, I. Dan (2003). “Multitext grammars and synchronous parsers”. *In: HLT-NAACL 2003, Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, Edmonton, Canada, May 27 - June 01, 2003*. Morristown: Association for Computational Linguistics, pp. 79–86.

Melamed, I. Dan, Giorgio Satta and Benjamin Wellington (2004). “Generalized multitext grammars”. In: *ACL-04, 42nd Annual Meeting on Association for Computational Linguistics, Proceedings of the Conference, 21 - 26 July, 2004, Barcelona, Spain*. Morristown: Association for Computational Linguistics, pp. 563–569.

Mezei, J. and Jesse B. Wright (1967). “Algebraic automata and context-free sets”, *Information and Control* 11(1-2): 3–29.

Mohri, Mehryar (1997). “Finite-state transducers in language and speech processing”, *Computational Linguistics* 23(2): 269–311.

Rounds, William C. (1970). “Mappings and grammars on trees”. *Mathematical Systems Theory* 4(3): 257-287.

Satta, Girogio and Enoch Peserico (2005). “Some computational complexity results for synchronous context-free grammars”. In: *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*. Morristown: Association for Computational Linguistics, pp. 803–810.

Shieber, Stuart M. (1994). “Restricting the weak generative capacity of synchronous tree-adjoining grammars”, *Computational Intelligence* 10(4): 371-385.

Shieber, Stuart M. (2004). “Synchronous grammars as tree transducers”. In: *Proceedings of the TAG+7: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms, May 20-22, 2004, Vancouver, BC, CA, 2004,* pp. 88–95.

Shieber, Stuart M. (2006). “Unifying synchronous tree-adjoining grammars and tree transducers via bimorphisms”. In: *EACL 2006, 11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, April 3-7, 2006, Trento, Italy.* Association for Computer Linguistics, pp. 377-384.

Shieber, Stuart M. and Yves Schabes (1990). “Generation and synchronous tree-adjoining grammars”. In: *COLING 1990, 13th International Conference on Computational Linguistics, August 20-25, 1990, University of Helsinki, Proceedings, Volume 3*, pp. 253-258.

Steinby, Magnus (1984). “Some decidable properties of Σ-rational and Σ-algebraic tree transformations”, *Annales Universitatis Turkuensis, Ser. A I* 186: 102-109.

Steinby, Magnus (1986). “On certain algebraically defined tree transformations”. In: János Demetrovics, Lothar Budach and Arto Salomaa, eds., *Algebra, Combinatorics and Logic in Computer Science, Vol. I, II, September 12-16, 1983, Györ, Hungary, vol. 42 of Colloquia Mathematica Societatis János Bolyai*. Amsterdam: North Holland, pp. 745–764.

Steinby, Magnus (1990). “A formal theory of errors in tree representations of patterns”, *Journal of Information Processing and Cybernetics*, EIK 26 (1/2): 19-32.

Steinby, Magnus and Cătălin I. Tîrnăucă (2007). “Syntax-directed translations and quasi-alphabetic tree bimorphisms”. In Jan Holub and Jan Zdárek, eds,, *Implementation and Application of Automata, 12th International Conference, CIAA 2007, Prague, Czech Republic, July 16-18, 2007, Revised Selected Papers.* Berlin: Springer-Verlag, pp. 265–276.

Takahashi, Masako (1972). “Primitive transformations of regular sets and recognizable sets”. In: Maurice Nivat, ed., *Automata, Languages and Programming, Colloquium, Paris, France, July 3-7, 1972*. Amsterdam: North Holland, pp. 475-480.

Takahashi, Masako (1977). “Rational relations of binary trees”. In: Arto Salomaa and Magnus Steinby, eds., *Automata, Languages and Programming Fourth Colloquium, University of Turku, Finland, July 18-22, vol. 52 of LNCS*. Berlin Heidelberg: Springer-Verlag, pp. 524–538.

Thatcher, James W. (1970). “Generalized^2 sequential machine maps”, *Journal of Computer and System Sciences* 4(4): 339–367.

Tîrnăucă, Cătălin I. (2007). “Synchronous context-free grammars by means of tree bimorphisms”. In: Gemma Bel Enguix and Maria Dolores Jiménez Lopez, eds., *Proceedings of the 1st International Workshop on Non-Classical Formal Languages in Linguistics (ForLing 2007), August 31, 2007, Budapest, Hungary, vol. 36/07 of GRLMC Reports*. Tarragona: Universitat Rovira i Virgili, pp. 97–107.

Yamada, Kenji and Kevin Knight (2001). “A syntax-based statistical translation model”. In: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, Toulouse, France, July 06 - 11, 2001.* Morristown: Asociation of Computational Linguistics, pp. 523-530.

Wu, Dekai (1997). “Stochastic inversion transduction grammars and bilingual parsing of parallel corpora”, *Computational Linguistics* 23(3): 377–403.