REPRESENTATION TREES AND STRING-TREE CORRESPONDENCES

presented for COLIN-88
Budapest, 22-27 August 1988

by

Ch. BOITET & Y. ZAHARIN

GETA, BP 68
Université de Grenoble et CNRS
38402 Saint-Martin-d'Hères, FRANCE

PTIK, Universiti Sains Malaysia
11800 Penang, MALAYSIA

ABSTRACT

The correspondence between a string of a language and its abstract representation, usually a (decorated) tree, is not straightforward. However, it is desirable to maintain it, for example to build structured editors for texts written in natural language. As such correspondences must be compositional, we call them "Structural String-Tree Correspondences" (SSTC).

We argue that a SSTC is in fact composed of two interrelated correspondences, one between nodes and substrings, and the other between subtrees and substrings, the substrings being possibly discontinuous in both cases. We then proceed to show how to define a SSTC with a Structural Correspondence Static Grammar (SSSG), and which constraints to put on the rules of the SSSG to get a "natural" SSTC.

KEYWORDS: linguistic descriptors, discontinuous constituents, discontinuous phrase structure grammars, structural string-tree correspondences, structural correspondence static grammars

ACRONYMS: DPSG, M, NL, SSTC, STG.

INTRODUCTION

Ordered trees, annotated with simple labels or complex "decorations" (property lists), are widely used for representing natural language (NL) utterances. This corresponds to a hierarchical view the utterance is decomposed into groups and subgroups. When the depth of linguistic analysis is such that a representation in terms of graphs, networks or sets of formulas would be more appropriate, one of the most used structures are trees, at the price of encoding the desired information in the decorations (e.g., "by "coindexing" two or more nodes). In both cases, trees are conceptually and algorithmically easier to manipulate, and also because all usual interpretations based on the linguistic structure are more or less "compositional" in nature.

If a language is described by a classical Phrase Structure Grammar, or by a (projective) Dependency Grammar, the tree structure "contains" the associated linguistic information. In particular, the surface order of the string is derived from some ordered traversal of the tree (left-to-right order of the leaves of a constituent tree, or index order for a dependency tree).

However, if one wants to associate "natural" structures to strings, for example abstract trees for programs or preicate-argument structures for NL utterances, this is no longer true. Elements of the string may have been erased, or duplicated, some "discontinuous" groups may have been put together, and the surface order may not be reflected in the tree (e.g., for a normalized representation). Such correspondences must be compositional: the complete tree corresponds to the complete string, so subtrees correspond to substrings, etc. Hence, we call them "Structured String-Tree Correspondences" (SSTC).

For some applications, like classical (batch) Machine Translation (MT), it is not necessary to keep the correspondence explicit: for revising a translation, it is enough to show the correspondence between two sentences or two paragraphs. However, if one wants to build structured editors for texts written in natural language, thereby using at the same time a string (the text) and a tree (its representation), it seems necessary to represent explicitly the associated SSTC.

In the first part, we briefly review the types of string-tree correspondences. In the second part, we review the types of string-tree correspondences. We argue that a SSTC should be composed of two interrelated correspondences, one between nodes and substrings, and the other between subtrees and substrings, the substrings being possibly discontinuous both in both cases. This is presented in more detail in the second part. In the last part, we show how to define a SSTC with a Structural Correspondence Static Grammar (SSSG), and which constraints to put on the rules of the SSSG to get a "natural" SSTC.

1. Why is a correspondence between a string and a tree?

Classical Phrase Structure trees give rise to a very simple kind of SSTC. To string w = a1...an, let us associate the set of intervals [i, j], 0 <= i <= j <= n, of length 0 or 1 (we may extend this to any length if terminals are allowed to be themselves strings). Then, the correspondence is such that any internal node of the tree, or equivalently each tree 'complete' in breadth and depth, corresponds to w(i, j). If the tree has daughters (or its immediate subtrees) in order, correspond to a sequence w(i1, j1), ..., w(im, jm), such that i1 <= ... <= im and j1 <= ... <= jm for 0 <= k <= m.

This type of correspondence is "projective". It has however been argued that classical phrase structure trees are inadequate to represent syntactic representations in general, especially in the case of so-called "discontinuous" constituents. Here are some examples:

- (1) John talked, of course, about politics.
- (2) He picked the ball up.
- (3) Je ne le lui ai pas donné.
   (I did not give it to him)

According to (McCawley 82), sentence (1) contains a verb phrase "talked about politics", which is divided by the adverbial phrase "of course", which modifies the whole sentence, and not only the verbal phrase (or the verbal phrase, in Thomps's terminology). Sentence (2) contains the particle "up", which is separated from its verb "picked" by the "ball". In sentence (3), the discontinuous negation "ne...pas" overlaps with the composed form of the verb "ai...donnè". Moreover, in a sentence in active voice it is to be represented in a standard order (subject verb object complement), this sentence contains two displaced elements, namely the object "le" and the complement "lui".

(McCawley 82) and later (Bunt & a) (87) have argued that "meaningful" representations of sentences (2) and (3) should be the following phrase structure trees, (4) and (5), respectively.

...
Along the same line, and taking into consideration the displaced elements, a "meaningful" representation for sentence (3) would be tree (6).

Figure 2: Example of discontinuity and displacement

Here, the correspondence is established between a node (or equivalently the complete subtree rooted at a node) and a sequence of intervals. If a displacement arises, as in (3), the left-to-right order of nodes in the tree may be incompatible with the order of the corresponding sequences of intervals in the string (the considered ordering is the natural lexicographic extension).

Rather than to introduce the awkward notion of "discontinuous" tree, as above, with intersecting branches, we suggest to keep the tree diagrams in their usual form and to show the string separately. For sentence (3), then, we get the following diagram.

Figure 3: Separation of a string and its "discontinuous" PS tree

Now, as before, the root of the tree still corresponds to \( w(0, n) \), and a leaf corresponds to an interval of length 0 or 1 (or more, see above). But an internal node with \( n \) daughters corresponds to a sequence of intervals, which is the union of the \( n \) sequences corresponding to its daughters.

More precisely, a "sequence" of intervals is a list of the form \( S = w(i_1, j_1), \ldots, w(i_p, j_p) \), in order \( \langle k \rangle \leq k' \langle k \rangle \) for \( k < k' \), and without overlapping \( \langle k \rangle \leq k' \langle k \rangle \) for \( k < k' \). Its union (denoted by \( \bigcup \)) with an interval \( \langle i, j \rangle \) is the smallest list containing all elements of \( S \) and \( \langle i, j \rangle \). For example, \( S \) is:

- \( S \) itself, if there is an \( i \) such that \( i \leq k \leq j \);
- \( S \), augmented with \( w(i, j) \) inserted in the proper place, if \( k \) or \( j \) or there is a \( k' \) such that \( k \leq k' \leq j \).
of the form "1A r --~ 1 u r", 1 and r being the left and grammar in normal form for this language (all rules are right ~:ontext, respectively).

It is natural to consider "incomplete" subtrees than the syntactic tree derived from a context-sensitive grammar in normal form for this language (all rules are right ~:ontext, respectively).

In order to simplify the trees, we abstract this by

"John and Mary give Paul and Ann trousers and
dressess.

In order to simplify the trees, we abstract this by

...
b) It can be encoded on the tree by attaching to each node N two sequences of intervals, called SNODE(N) and STREE(N), such that:

1. SNODE(N) \subseteq STREE(N), which means that SNODE(N) is contained in STREE(N) with respect to its basic elements (the w(i,j)'). That is, that STREE(N) = STREE(N) \cup SNODE(N). Note that equality can not be required, even on the leaves, because the string "(b)" may well have a representation tree with the unique node b.

2. If N has a daughters N1...Nn, then STREE(N) \geq STREE(N1)+...+STREE(Nn) \geq SNODE(N). In case of strict containment, the difference corresponds to the elements of the string which are represented by the subtree but which are not explicitly represented, like "(b)" in "(b)".

3. The sequence SSUB1(X,N) corresponding to a given incomplete subtree X rooted at node N of the whole tree T is defined recursively as:
   - SSUB1(X,N) = STREE(X) if X = N, that is, if X is reduced to one node, not necessarily a leaf of T;
   - SSUB1(X,N) = SSUB1(X,...,SSUB1(Xp),U SNODE(N), if N, the root of X, has p successors X1...Xp in T.

In other words, one takes the smallest sequence containing the biggest sequence corresponding to the leaves of X (STREE on the leaves) and compatible with the monotony rules above.

2. PROPERTIES

Here are some interesting properties of SSTCs which may help to classify them.

A SSTC is non-overlapping if

- STREE(N1) and STREE(N2) have an empty intersection if N1 and N2 are independent;
- SNODE(N1) and STREE(N2) have an empty intersection if N2 is a daughter of N1.

A SSTC is projective if

- it is non-overlapping;
- for any two sister nodes N1 and N2, N1 to the left of N2, STREE(N1) is completely to the left of STREE(N2). This means that, if STREE(N1) = w[i1..j1], w[ip..jp] or ji and STREE(N2) = w[i1..j1], w[kd..le] or kl, then jdp.

A SSTC is direct if, for each elementary interval w[i..j+1], there is a node N such that SNODE(N) = \{x i..j\}.

A SSTC is complete if each elementary interval is contained in SNODE(N) for some node N.

A SSTC is of the constituent type if SNODE(N) is empty for each non-terminal node N.

3. QUESTIONS OF REPRESENTATION

In the examples above, we have encoded the correspondence in the tree. However, this is in practice not always necessary, or even practical.

In the case of explicit and projective SSTCs, for instance, the string can be contained directly from the tree, and there is no need to show the intervals.

Note that, in the process of generating a string from a tree, one naturally starts from the top, not knowing the final length of the string, and goes down recursively, dividing this interval into smaller intervals. Rather than to introduce variables representing the extremities of the created intervals, it may be more practical to start from a fixed interval, say 0,1 or 0,100. Then, the positions between the elements of the string will be denoted by an increasing sequence of rational numbers (0, 1/2, 1/2, 5/7, etc).

In the case of "local" non-projectivity, we have tried some devices using two relative integers (POS,LEV) associated with each node N. POS(N) shows the relative order in the subtree rooted at mother(N), if LEV(N)=0, or more generally at its LEV(N)+1 ancestor, if LEV(N)>0. Unfortunately, all these schemes seem to work only for particular situations.

Also, if the SSTC is overlapping, or not complete, it may be computationally costly to find the (smallest) subtree associated with a given (possibly discontinuous) substring. But this operation would be essential in a "structural" editor of NL texts. A possibility is then to encode the correspondence both in the tree and in the string.

Finally, take the example of tree (15) above. Suppose that the user of a NL editor wants to change the Paul, in the corresponding NL example, in a way which may contradict some agreement constraints between a, b, c, and cl. One should be able to find the smallest SSTC containing a and other elements, that is, the subtree V(a1blbc) and the discontinuous substring a b c cl (the notation a...b...c... might be suitable, if one wants to avoid indices).

For these reasons, it may be worthwhile to consider the possibility of representing the SSTC independently of both the tree and the string. This is actually the idea behind the formalism of STCS (String-Tree Correspondence Grammar).

III. EXTENDING THE STCS FORMALISM TO DEFINE A SSTC

1. BASIC NOTIONS ABOUT STCS

The static grammars of (Vauquols & Chappuy 85) are devices to define string-tree correspondences. They have been formalized by the STCSs of (Zaharir 86).

Here, a context-free like apparatus of rules (also called "boards", for "planches" in French, because they are usually written with two-dimensional tree diagrams) is used to construct the set of "legal" SSTCs.

The axioms are all pairs (X,Y($F)), where X is an unbounded string variable, Y a starting node (standing for SENTENCE, or TITLE, for example), and $F is an unbounded forest variable.

The terminals are all pairs (x,x'), where x is an element of a string and x' a one-node tree which represents it.

The rules show how a SSTC is made up of smaller ones. The generated language is the set of all variable-free (string,tree) pairs derivable from an axiom by the grammar rules.

In order to avoid undue formalism, let us give an example for the formal language (can bn cn 1n 1n).

\[ \text{Rule R1: (a,b,c),(a,b,c)} \]
\[ \text{Rule R2: (a,b,c), (a,b,c)} \]

Figure 8: A simple STCS for an bn cn

X, Y and Z are string variables, $F$ a forest variable, and the indices are just there to distinguish elements with the same label.

Actually, the formalism is a bit more precise and powerful, because it is possible to express that a correspondence in the r.h.s. (right hand side) is obtained only by certain rules, and to restrict the possible unifications (rather, a special kind called

\[ \text{Rule R1: (a,b,c),(a,b,c)} \]
\[ \text{Rule R2: (a,b,c), (a,b,c)} \]

Figure 9: A simple SCG for an bn cn

X, Y and Z are string variables, $F$ a forest variable, and the indices are just there to distinguish elements with the same label.
identifications" in (Zaharim 86)). To illustrate this, we may rewrite the last element of the r.h.s as:

\[ (X Y Z, S.2(SF)) \]

with ref

(R1: X/a, Y/b, Z/c, S.Z/S)
(R2: X/aX, Y/bY, Z/cZ, SF/(a,b,c,S.2(SF)))

![Figure 10: Example of with ref part in a r.h.s.](image)

R2: \( X/aX, ... \) means that the subcorrespondence \((XZ, S.2(SF))\) may be generated by rule R2, thereby identifying \( X \) in \( XYZ \) with \( ax \) in \( axyz \) (in the r.h.s.).

In the version of (Zaharim 86), the correspondence is always of constituent type, because the only applications considered had been to \( \lambda \)-structures used for \( \lambda \)-trees, where non-terminal nodes do not directly correspond to substrings.

But this is by no means necessary, as the next example illustrates, with the language \((an \ v \ bn \ cn \ n>0)\).

\[ \text{Rule RI: } (a b c, V(a, b, c)) \]
\[ \text{Rule R2: } (a X b Y c Z, V.1(a, b, c, V.2(SF))) \]

![Figure 11: STCG for an \( an \ v \ bn \ cn \) on giving tree (15)](image)

This STCG generates correspondences such as \((anbombo) \text{ tree (15)}\). But something has to be added to distinguish the STREE and SNODE parts.

2. AN EXPANSION

We simply associate to each constant or variable appearing in a STCG rule one or two expressions representing the STREE and SNODE sequences, separated by a "/" if necessary, with basic elements of the form "p_q", where \( p \) and \( q \) are constant or variable indices.

In any given \((\text{string}, \text{tree})\) pair, we associate such an expression to each element of \( \text{string} \), and to each node of \( \text{tree} \), the first for STREE and the second for SNODE. The second may be omitted; by default, SNODE is taken to be empty on internal nodes and equal to STREE on leaves.

Our last example may now be rewritten as follows.

\[
\begin{align*}
\text{Rule R1: } & (a b c, V(a, b, c)) \\
& \Rightarrow (a,a), (b,b), (c,c) \\
\text{Rule R2: } & (a X b Y c Z, V.1(a, b, c, V.2(SF))) \\
& \Rightarrow (a,a) (b,b) (c,c) (X Y Z, V.2) \\
& \text{with ref (R1: X/a, Y/b, Z/c, S.2(SF)) (R2: X/aX, Y/bY, Z/cZ, S.2(SF))}
\end{align*}
\]

![Figure 12: Extended STCG for an \( an \ v \ bn \) on](image)

We will now give examples of STCGs which give rise to unnatural correspondences and try to derive some constraints on the rules. Let us first slightly modify our first STCG for an \( an \ bn \) on.

\[
\begin{align*}
\text{Rule RI: } & (a b c, S(a, b, c)) \\
& \Rightarrow (a,a), (b,b), (c,c) \\
\text{Rule R2: } & (a X b Y c Z, S.1(a, b, c, S.2(SF))) \\
& \Rightarrow (a,a) (b,b) (c,c) (X Y Z) \\
& \text{with ref (R1: X/a, Y/b, Z/c, S.2(SF)) (R2: X/aX, Y/bY, Z/cZ, S.2(SF))}
\end{align*}
\]

![Figure 11: STCG for an \( an \ v \ bn \) on giving tree (15)](image)

In the first element of R2, XYZ has been replaced by ZYX. The following representation tree (16) would have been naturally associated with the string \( a1a2a3b1b2b3c1c2c3 \) by our first STCG. With this modification, it becomes associated with \( a1a2a3b1b2b3c1c2c3 \), as shown in the next diagram.

\[
\begin{align*}
\text{Rule R1: } & (a b c, S(a, b, c)) \\
& \Rightarrow (a,a), (b,b), (c,c) \\
\text{Rule R2: } & (a Z b Y c X, S.1(a, b, c, S.2(SF))) \\
& \Rightarrow (a,a) (b,b) (c,c) (X Y Z, S.2(SF))
\end{align*}
\]

![Figure 13: Example of "unordered" STCG](image)

The problem here is that the subtree rooted at \( S.2, \) considered as a whole tree, should correspond to the string \( a1a2a3b1b2b3c1c2c3 \) or \( a2a3b1b2b3c1a2c3 \) embedded in the whole tree rooted at \( S.1. \)

The STREE correspondences are not properly defined, because one should be able to distinguish between different permutations of the intervals, which is clearly

![Figure 14: Example of STCG "unordered" w.r.t. the strings](image)
impossible with our previous definitions and representations of SSTCs.

This is because the order of the elements of the strings is not compatible in the l.h.s. and in the r.h.s.: our first constraint will be to forbid this in STCG rules.

Our second constraint will be to forbid the use of auxiliary variables which do not correspond to substrings (subtrees) of the terminal (variable-free) pairs produced by the STCG.

Let us illustrate this with the following STCG, which constructs the representation tree $S(A(u),B(v))$ for each word $w$ on $(a,b,c)$ of even length such that $w=uv$ and $u\neq v$.

\[
\text{Rule R1: } (P, S(A(x),P)) \quad \Rightarrow \quad x \in (a,b,c) \\
\text{Rule R2: } (A X Y P, S(A(x),Y),B(P)) \quad \Rightarrow \quad x \in (a,b,c) \\
\text{Rule R3: } (A X Y, S(A(X),B(Y))) \quad \Rightarrow \quad x \in (a,b,c)
\]

Figure 15: Example of STCGs with auxiliary variables

There is a natural SSTC between the representation tree and the string. For example, we get $S(A(a,b,c),B(d,b,c))$ for $w=abcd$. But the construction of this final correspondence involves the construction of pairs such as $(a,b,c),P,P)$, which are only used for counting.

If we try to put sequence expressions on the $P$ nodes and string elements, we notice that it would be necessary to extend the intervals of $w$, rather than to divide them. Otherwise, we would make the first $P$ of $aPPPP$ correspond to the second $b$ of $wababc$, which is quite natural, but what would we associate to the first $P$ of $aPPPP$?

If we represent explicitly (and separately) the structure of a given <string,tree> element of the SSTC by its derivation tree in the STCG, the second constraint will allow us to instantiate all variables by substrings or subtrees of <string and tree>, without having to construct other auxiliary strings and trees. This, of course, would permit a more economical implementation, in terms of space.

Finally, note that the interesting properties of SSTCs mentioned in 11.1 above have simple expressions as constraints on the rules of our extended STCG formalism.

CONCLUDING REMARKS

Trees have been widely used for the representation of natural language utterances. However, there have been arguments saying that they are not adequate for representing the so-called “discontinuous” structures. This has led to various solutions, relying, for instance, on encoding the desired information in the nodes (e.g. “composing”), or on defining trees with “discontinuous” constituents.

We have presented here a proposal for representing discontinuous constituents, and, more generally, non-projective and uncomplete SSTCs with overlapping.

The proposal uses the ordinary definition of ordered trees. This is made possible by separating the representation tree from the surface utterance (which the tree is a representation of). The correspondence between the two may be represented explicitly by means of sequences of intervals attached to the nodes.

This opens up a discussion on (and definitions of) structured string-tree correspondences in general. This representation might also be used in syntactic editors for programs or in syntactic-analytic editors for NL texts.

Finally, the formalism of the String-Tree Correspondence Grammar has been extended to give the means of representing the said structured correspondences.

An analogous problem is to define structured correspondences between representation trees, for instance between source and target interface structures in transfer-based MT systems. We do not yet know of any satisfactory proposal.

A solution to this problem would give two very interesting results:

- first, a way to specify structural transfers in a reasoned manner, just as STCGs are used to specify structural analysers or generators,

- second, a way to put a text and its translation in a very fine-grained correspondence. This is quite easy with word-for-word approaches, of course, and also for approaches using classical (projective) PS trees or dependency trees, but has become quite difficult with more sophisticated approaches using p-structures or m-structures.

REFERENCES

(Bunt & al 87) H. BUNT, J. THESENH & K. VAN DER SLOOT (1987) Discontinuous constituents in trees, rules and parsing. Proc. 3rd Conf. ACL European Chapter, Copenhagen, April 1987.

(McCawley 82) J.D. MCCAWLEY (1982) Parenthetical and discontinuous constituent structure. Linguistic Inquiry 13 (1), 91-106, 1982.

(Vauquois 87b) B. VAUQUOIS (1987) Description de la structure intermédiaire. Communication présentée au colloque de Luxembourg, April 1976. G/TA document, Grenoble.

(Vauquois & Boitet 87) B. VAUQUOIS & CH. BOITET (1987) Automated translation at G/TA (Grenoble University) Computational Linguistics, 11:1, 28-36, January 1985.

(Vauquois & Charpy 85) B. VAUQUOIS & S. CHARPY (1985) Static Grammars. Proc. Conf. on theoretical & methodological issues in MT, Colgate Univ., Hamilton, N.Y., August 1985.

(Zaharin 86) Y. ZAHARIN (1986) Strategies and neuistics in the analysis of natural language in Machine Translation. Ph.D. Thesis, Universiti Sains Malaysia, March 1986 (Research conducted under G/TA-USM cooperation G/TA document, Grenoble).

(Zaharin 87a) Y. ZAHARIN (1987) String-trees: A correspondence grammar: a declarative formalism for defining the correspondence between strings of terms and tree structures. Proc. 3rd Conf. ACL European Chapter, Copenhagen, April 1987.

(Zaharin 87b) Y. ZAHARIN (1987) The linguistic approach to G/TA: a sequel to the journal TECHNOLOGOS (LISH-CNRS), printemps, 1987, Paris.

(Zajac 88) R. ZAJAC (1988) SEASL: A linguistic specification language for MT. Proc. of COLING-88, IKS, 399-398, Bonn, August 26-28, 1988.