A Standard Representation Framework for TAG

Fabrice ISSAC
L.I.P.N. - Institut Galilée
Av J. B. Clément 93430 Villetaneuse
e-mail : fabrice.issac@lipn.univ-paris13.fr

Abstract

We present in this paper a markup language suitable for representing a tree adjoining grammar. Using a uniform way to represent TAG, the development of tools, e.g. parser/recognizer, editor, ..., could be done to the benefit of the entire TAG community.

Key words: SGML, linguistic tagging, data exchange.

Our work consists of proposing a framework dedicated to designing an XTAG-like standard environment. We present in this paper a markup language suitable for representing a tree adjoining grammar. The TAG formalism is used in plenty of works all around the world. However it is difficult to exchange syntactic data (*i.e.* a grammar or a piece of grammar written in the TAG formalism) as well as computer tools based on these syntactic data. Using a uniform way to represent TAG, the development of tools, e.g. parser/recognizer, editor, ..., could be done to the benefit of the entire TAG community. Note that all kinds of TAG (LTAG, MCTAG, ...) can be represented in our language. We choose SGML as descriptive language, as seen as briefly in the first section. in the second section, we provide an overview of the structure of a TAG document, followed in the third section by an example.

1 SGML

We describe our language in SGML(Goldfarb90; Herwijnen95) (Standard Generalized Markup Language). SGML is itself a metalanguage. SGML is an efficient tool to describe classes of documents because (i) it is an ISO specification[1], thus, a standard (ii) a lot of tools can be used to edit, verify or exploit SGML based documents[2].

SGML is a meta-language which allows specification through a Document Type Definition (DTD) :

- a set of markups;
- how these markups can be combined.

In our case, the class of documents is the set of TAG grammars. The most popular DTD is HTML which is used as a norm for data representation on the *Web* but there are other projects, notably the TEI project. The Text Encoding Initiative (TEI) (SMB94) is an international project to develop guidelines for the preparation and interchange of electronic texts.

The TEI proposes recommendations for feature structure markup (LS95) which can be used to represent any feature structure, including TAG. However, we think the markup set defined is not specific enough to be easily treated.

2 Structure of a TAG document

First a *good* TAG document is preceded by a prologue which indicates the TAG DTD version:
`<!DOCTYPE DTD PUBLIC "DTD TAG 0.2">`

The whole document is enclosed by the `<tag>` and `</tag>` markup. It is composed of a *header* and a *body*[3]. The *header*, enclosed by the `<tagheader>` and `</tagheader>` markup, contains information about the document itself: title, date of creation, name of the creator, origin of the data, type of data.

The *body*, enclosed by the `<ts>` (Tree Set) and `</ts>` markup, forms the usable part of the document. A tree set is a list of tree families (`<tf>`), elementary trees (`<et>`) or parsed trees (`<pt>`).

[1] ISO 8879:1986.

[2] For instance the `sgmlql` tool can extract part of document in relation to a query on the tags.

[3] As HTML, in fact it is a classical way to describe a SGML document.

<tf> (tree family) encloses a tree family. The attribute (name) indicates the name of the family. A tree family is composed of a list of trees (markup <t>).

<et> (elementary trees) encloses elementary trees. The attribute (name) indicates the name of the tree. As a family tree, elementary trees are composed of a list of trees (possibly one).

<pt> (parsed tree) encloses a parsed tree (*i.e.* a derived tree). Three markups are used to describe (i) the string recognized by the tree (<string>) (ii) the tree itself (<t>) and a set of derivation trees (DT).

A tree contains only one node (a node is indicate by <n>): the root node. The node markup can then be used recursively (a <n> can contain a <n>) to define the tree. A node contains some markups:

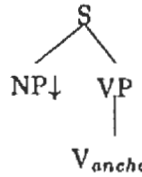
<val> (value) is the tag of the node (*i.e.* category for elementary and derived trees or tree name for a derived tree);

<fs> (feature structure) for FB-TAG.

The tree of the figure 1 indicates the relationships between markups[4].

3 An example

We give below a simple example. Let us suppose we have a tree with the features associated to the nodes :




```
NP_0.t:<num>=VP.t:<num>
NP_0:<pers>=VP.t:<pers>
S.b:<mode>=VP.t:<mode>
VP.b:<mode>=V.b:<mode>
VP.b:<num>=V.b:<num>
VP.b:<pers>=V.b:<pers>
NP_0.t:<wh>=-
S.b:<inv>=-
```

The SGML result is the following :

```
<!DOCTYPE DTD PUBLIC "DTD TAG 0.2>
<tag lang=french>
```

[4]Note that this tree is automaticaly generated with a SGML tool: dtdtree.

```
TAG
 |_(tagheader,
 |   |_(title &
 |   |   |_(#PCDATA)
 |   |
 |   |__date &
 |   |   |_(#PCDATA)
 |   |
 |   |__creator &
 |   |   |_(#PCDATA)
 |   |
 |   |__orig* &
 |   |   |_(#PCDATA)
 |   |
 |   |__type?)
 |       |_(#PCDATA)
 |
 |__ts+)
     |_(tf |
     |   |_(t+)
     |       |_(n)
     |           |_(val,
     |           |   |_(#RCDATA)
     |           |
     |           |__fs?,
     |           |   |_(f+)
     |           |       |_(l |
     |           |       |   |_(EMPTY)
     |           |       |
     |           |       |__#RCDATA)
     |           |
     |           |__n+) ...
     |
     |__et |
     |   |_(t+) ...
     |
     |__pt)+
         |_(string,
         |   |_(#PCDATA)
         |
         |__t+, ...
         |__dt?)
             |_(n) ...
```

Figure 1: DTD tree

```
<tagheader>
<title>TAG for an intransitive
structure</title>
<date>april 1998</date>
<creator>Fabrice Issac</creator>
<orig>Anne Abeillé</orig>
</tagheader> <tf name=s.np(v)>
<t name=s.np(v)>
<n id=n1>
<val>&S;</val>
<fs type=b>
<f name=mode><l id=f1></f>
<f name=inv>-</f>
</fs>
<n id=n2 type=substitute>
<val>&NP;</val>
<fs type=t>
<f name=num><l id=f2></f>
<f name=pers><l id=f3></f>
<f name=wh>-</f>
</fs>
</n>
```

```
<n id = n3>
<val>&VP;</val>
<fs type=t>
<f name=num><l id=f2></f>
<f name=pers><l id=f3></f>
<f name=mode><l id=f4></f>
</fs>
<fs type=b>
<f name=mode><l id=f5></f>
<f name=num><l id=f6></f>
<f name=pers><l id=f7></f>
</fs>
<n id=n4 type=anchor>
<val>&V;</val>
<fs type=b>
<f name=mode><l id=f5></f>
<f name=num><l id=f6></f>
<f name=pers><l id=f7></f>
</fs>
</n>
</n>
</n>
</t>
...
</tf>
</tag>
```

4 conclusion

Finally we will mention the way we can use such a description. It is obvious that an SGML document, as this one, is not supposed to be directly understandable/readable to humans. It is, in fact, used as input/output to computer tools. For instance, if developers follow these guidelines, the three steps of a parser – grammar generation, tree elimination, parsing – could be built by three different people.

The final environment will contain ;

- a graphical TAG editor, in order to create or to modify TAG grammars;
- tools for parsing (generation, disambiguation, parsing);
- miscellaneous tools (for instance a LaTeX or HTML transduction of TAG trees).

The aim of this paper is not to give a final (nor complete) version of a language providing descriptions of TAG, but rather act as a starting point. Actually, I think a data interchange norm can't be established by only one person. That's why I wish the community take a part in the development of this norm.

References

Goldfarb (Charles F.). – *The SGML handbook.* – Oxford, Clarendon Press, 1990.

Herwijnen (Eric Van). – *SGML pratique.* – Paris, International Thomson Publishing France, 1995.

Langendoen (D. Terence) et Simons (Gary F.). – A rationale for the tei recommendations for feature-structure markup. *In : Computers and the humanities*, pp. 191–209. – Kluwer Academic Publishers, 1995.

Sperberg-McQueen (C.M.) et Burnard (Lou). – *Guidelines for Electronic Text Encoding and Interchange (TEI P3).* – Text Encoding Initiative, 1994.

Appendix: The DTD

```
<!--**********************************************************
       file : tag.dtd
       author: Fabrice Issac
       date : April 1998
       Email : fabrice.issac@lipn.univ-paris13.fr
**********************************************************-->

<!ENTITY % TAG.Version
       "DTD TAG 0.2"
       -- Typical usage:
           <!DOCTYPE DTD PUBLIC "DTD TAG 0.2">
           <tag>
           ...
           </tag>
       --
       >

<!--======= Character mnemonic entities for french ====-->

<!ENTITY % Categories PUBLIC "TAG entities VERSION 1.0 FRENCH">
%Categories;

<!--======= ========================================== ====-->

<!ENTITY % TAG.Simple           "IGNORE">
<!ENTITY % TAG.Recommended      "INCLUDE">

<![%TAG.Recommended [
       <!ELEMENT TAG            - - (TAGHEADER,TS+)>
]]>

<![%TAG.Simple [
       <!ELEMENT TAG            - - (TAGHEADER?,TS+)>
]]>

<!ELEMENT TAG           - - (TAGHEADER,TS+)>
```

```
<!ELEMENT TAGHEADER     - - (TITLE & DATE & CREATOR & ORIG*
                             & TYPE?)>

<!ELEMENT TITLE         - - (#PCDATA)>
<!ELEMENT DATE          - - (#PCDATA)>
<!ELEMENT CREATOR       - - (#PCDATA)>
<!ELEMENT ORIG          - - (#PCDATA)>
<!ELEMENT TYPE          - - (#PCDATA)>

<!ELEMENT TS            - - (TF|ET|PT)+>
<!ATTLIST TS    LANG            CDATA           #IMPLIED
                ID              CDATA           #IMPLIED>

<!ELEMENT (TF|ET)       - - (T+)>
<!ATTLIST TF    NAME            CDATA           #REQUIRED
                ID              CDATA           #IMPLIED>
<!ATTLIST ET    ID              CDATA           #IMPLIED>

<!ELEMENT PT            - - (STRING,T+,DT?)>
<!ATTLIST PT    ID              CDATA           #IMPLIED>

<!ELEMENT STRING                - - (#PCDATA)>
<!ATTLIST STRING        ID      CDATA           #IMPLIED>

<!ELEMENT DT            - - (N)>

<!ELEMENT T             - - (N)>
<!ELEMENT N             - - (VAL,FS?,N*)>

<!ELEMENT VAL           - - (#RCDATA)>
<!ELEMENT FS            - - (F*)>
<!ELEMENT F             - - (L|#RCDATA)>

<!ELEMENT L             - O (EMPTY)>
<!ATTLIST L     ID              CDATA           #IMPLIED>
<!--=========================================================-->
```