Environnement de traitement des TAG dans le groupe ATOLL

Eric de la Clergerie

Projet ATOLL – INRIA

25 Octobre 2000

Groupe de travail TAGML

Vue générale

Actuellement, l'environnement TAG / ATOLL comprend :

- Des DTD (grammaire TAG et forêt de dérivation TAG)
- Des grammaires en format XML (et autres formats)
- Des outils de maintenance en Perl
- Des « *parseurs* »
  (grammaires ; approches RCG et DyALog ; stratégies d'analyse)
- Un serveur de parseurs (+ « *frontend* » WEB)
  (choix du parseur ; envoi d'une phrase ; récupération de la forêt de dérivation)
- Interface Java de navigation dans les grammaires et les dérivations
  (adaptée de P. Lopez ; utilise le serveur de parseurs).

DTD

Grammaire http://atoll.inria.fr/~clerger/tag.dtd,xml

- Adaptée des discussions du groupe et de la DTD de P. Lopez
- Couvre Arbre, Famille, Lexiques syntaxiques et morphologiques, Structures de traits, ...
- DTD prototype incluant templates (morph and synt) et ancres multiples
- Pb sur la notion d'ancre principale

Forêt de dérivation http://atoll.inria.fr/~clerger/foret.dtd,xml

Décrite plus loin

Morphologie

```
<morph lex="donne">
      <lemmaref cat="v" name="*DONNER*">
          <fs>
              <f name="mode"><val>ind</val><val>subj</val></f>
              <f name="num"><val>sing</val></f>
          </fs>
      </lemmaref>
</morph>
```

Lemmes

```
<lemma cat="v" name="*DONNER*">
     <anchor tree_id="family[@name=tn1pn2]">
         <coanchor node_id="p_2"> <lex>à</lex> </coanchor>
         <equation node_id="np_0" type="top">
             <fs> <f name="restr"><val>plushum</val></f> </fs>
         </equation>
         <equation node_id="np_2" type="top">
             <fs> <f name="restr"><val>plushum</val></f> </fs>
         </equation>
     </anchor>
</lemma>
```

Arbres et familles

```
<family name="tn1pn2">
    <tree name="tn1pn2">
        <node cat="s" adj="yes" type="std">
            <node cat="np" id="np_0" adj="no" type="subst" />
            <node cat="vp" adj="yes" type="std">
                <node cat="v" adj="yes" type="anchor" />
                <node cat="np" adj="no" type="subst" />
                <node cat="pp" adj="yes" type="std">
                   <node cat="p" id="p_2" adj="no" type="subst" />
                   <node cat="np" id="np_2" adj="no" type="subst" />
                </node>
            </node>
         </node>
    </tree>
</family>
```

Noeud détaillé

```
<node cat="s" adj="yes" type="std">
   <narg type="bot">
       <fs>
          <f name="inv"><minus /></f>
          <f id="X2" name="mode" />
       </fs>
   </narg>
   <node cat="np" id="np_0" adj="no" type="subst">
   ...
</node>
```

Notes sur la DTD Grammaire

- Un arbre peut appartenir à plusieurs familles (élément `treeref`).
- Tout arbre appartient à au moins une famille
- Les valeurs des co-ancres sont des lexicaux

Nouvelle DTD Grammaire (prototype)

1. La valeur d'une co-ancre peut être un `lemmaref`

2. « Template morphologique » `template.fs` ≡ structure de traits (`fs`) nommée.

   – Une séquence de `fs` et références `fsref` à un template ≡ unification

   – `template.fs` peut être défini avec d'autres `template.fs`

   – `fsref` autorisé partout où `fs` est autorisé.

3. « Template syntaxique » ≡ `equation` nommée. Référence avec `eqref`.

4. Disparition de la notion de lemme, ancre principale d'une série d'ancrage remplacée par une liste de `lexicalization` indiquant comme lexicaliser les arbres.

   Le choix de l'ancre « principale » ou discriminante est laissé au parseur.

Grammaires

Conversions depuis le(s) format(s) XTAG, réalisées par F. Barthelemy.

Grammaire	Taille	XML	RCG	LP	Stripped XML
Small French TAG	50/115/338	oui	oui	oui	oui
Medium English TAG	456/333/507	oui	oui	non	oui
Large English TAG	1K//320K	non	oui	non	non
Large French TAG	5K//48K	non	oui	non	non

– dans [ME-TAG] : arbres dans plusieurs familles

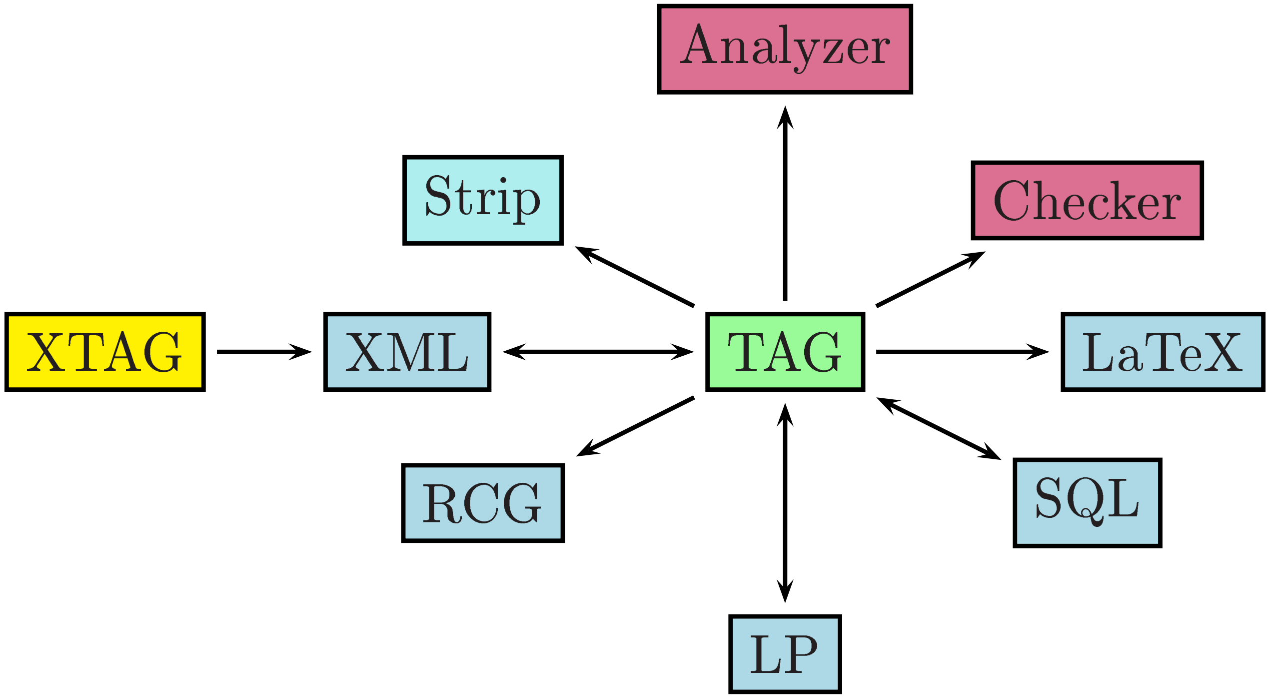
– dans [LE-TAG] : formes morphologiques contractées (*might've*) associées à des séquences de lemmes ; pas de notion d'ancre principale ; une ancre est un lemme.

Modules Perl de maintenance

– Collection de module Perl pour différentes taches de maintenances (conversions, analyses, transformations).

– S'appuie sur une représentation objet reflétant la DTD

– Developpement rapide des modules (facilité par les modules Perl du CPAN).



Description des parseurs

RCG Transformation TAG vers RCG ; ni unification ni coancres

DyALog Transformation TAG vers LP ; unification + coancres

- **Tabulation Forte** Complexité « pire des cas » optimale $O(n^6)$
  - **Stratégie Top-Down**
  - **Stratégie Hybride** bottom-up sur les structures de traits ; + efficace
- **Tabulation Faible** Complexité « pire des cas » non optimale mais + efficace
  - **Stratégie Top-Down**
  - **Stratégie Hybride** bottom-up sur les structures de traits ; + efficace
- **Tabulation Faible Interprétée** Méta-interprète TAG avec lexicalisation simulant la tabulation faible. Ni unification ni coancres.

Guided Un parseur RCG guide un parseur DyALog modifié
Tabulation forte ou faible ; stratégie Top-Down

Liste des Parseurs

Grammaire	RCG	Fort	Faible	Int
Small French TAG	oui	oui	oui	oui
Medium English TAG	oui	non	non	oui
Large English TAG	oui	non	non	non
Large French TAG	oui	non	non	non

Résultats (pour Small French TAG)

	RCG	Strong			Weak			Int.
Sentence		**TD**	**Hyb**	**Gd**	**TD**	**Hyb**	**Gd**	**TD**
Yves montre des fleurs à Sharon	0.00	1.84	1.23	0.30	0.38	0.29	0.29	0.33
Kathy regarde un livre avec Stéphane	0.00	1.44	1.24	0.29	0.36	0.31	0.29	0.31
un ami américain de Yves qui visite la France boit du bon vin	0.01	2.87	2.01	0.30	0.64	0.48	0.27	0.34
la jolie fille pense que Andrew est le grand ami de son chien	0.00	2.10	1.65	0.28	0.50	0.35	0.32	0.40
Marie comprend que Kathy dort bien chez Andrew	0.00	1.74	1.39	0.28	0.42	0.34	0.31	0.31
Anne voit les fleurs que Yves cherche sur la table	0.00	2.22	1.73	0.33	0.46	0.39	0.30	0.31
Jean pense que Béatrice perd les pédales	0.01	1.81	1.43	0.32	0.44	0.34	0.28	0.33
le petit chat de Sabine court vite	0.00	1.37	1.11	0.28	0.43	0.32	0.27	0.35
Andrew court parfois les filles	0.00	1.30	1.11	0.29	0.33	0.31	0.28	0.31
la fille de Marie aime bien les fleurs	0.00	1.61	1.30	0.32	0.37	0.33	0.30	0.32
qui dit que Yves a un chien	0.01	1.62	1.29	0.29	0.39	0.29	0.27	0.30
un enfant intelligent de Marie comprend lentement que sa mère est bête	0.01	2.53	1.75	0.31	0.57	0.38	0.33	0.50

Serveur de Parseurs

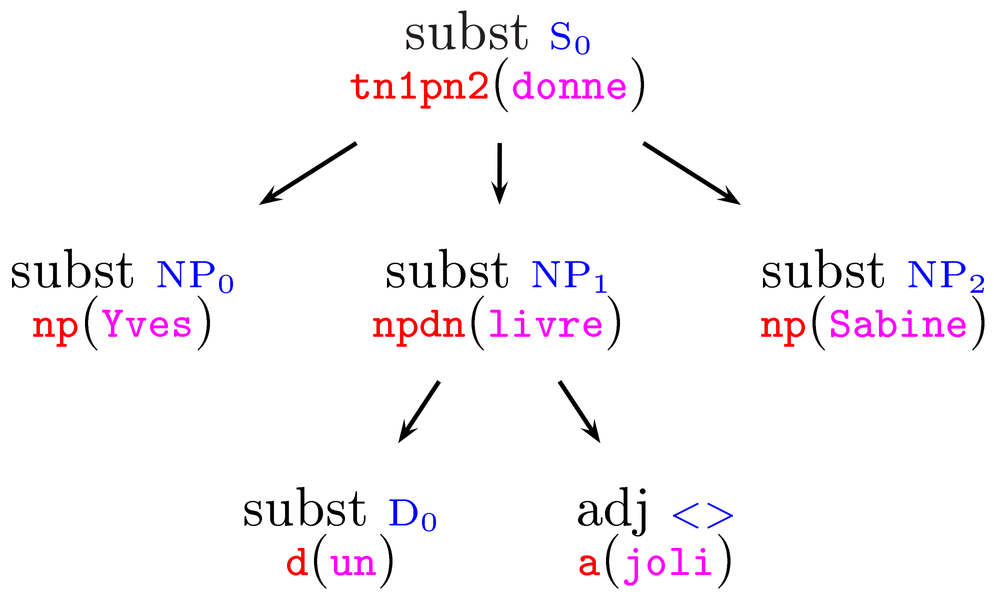
Gestion simple des parseurs ⇒ mise en place d'un serveur de parseurs

1. Sélection d'un parseur
2. Envoi d'une phrase
3. Récupération de la forêt de dérivation en format `raw`, `html`, ou `xml` (respectant la DTD Forêt)

Utilisation directe : `telnet medoc.inria.fr 8999`

Front-end WEB : `http://medoc.inria.fr/pub-cgi-bin/parser.cgi`

Forêt de dérivation




```
<op cat="s" span="0 7" id="1" type="subst">
    <deriv tree="tn1pn2" anchor="donne">
      <node id="p_2"><opref ref="5" /></node>
      <node id="np_0"><opref ref="2" /></node>
      <node id="1"><opref ref="4" /></node>
      <node id="np_2"><opref ref="6" /></node>
    </deriv>
</op>
```

Navigateur Java (dérivations et grammaires)

Travail de DEA de Linda Kaouane à partir de l'outil de P. Lopez.

– Appelle un serveur de parseurs

– Récupère la forêt de dérivation

– Permet la navigation pas à pas dans les arbres de dérivation et arbres d'analyse

– Permet la navigation dans les lexiques et les arbres d'une grammaire

Distribution libre si obtention des droits
([Thomson] binaires Arbres ; [Bonhomme] binaires XML ; [Lopez] sources)

Future de ce navigateur ?

– améliorer l'interface et l'ergonomie

– naviguer autrement dans les forêts

– évolution avec DTD Grammaire

Evolution

– Stabilisation DTD Grammaire

– Mise à niveau des modules et outils avec cette DTD

– Conversion des grammaires (XML et autres formats)

– Mise en place d'un server de grammaires, permettant d'obtenir des fragments de grammaire (nécessite un petit langage de requêtes).

– Parseurs pour les « grosses » grammaires (RCG, Faible, Guided)

Examples d'ancrages multiples (dans [LE-TAG])

– Erreurs ou justification linguistique ?

```
N P: N [7] { case:case:in thanks:thanks:to thanks:thanks:to
             order:order:in order: order:in kind:of:kind sort:of:sort }
N P: P [1] { for:all:for }
```

```
P1 P2: P1 [32] { as:as:from as:as:in as:of:as as:as:if ... }
P1 P2: P2 [1]  { as:as:so }
```

– Choix arbitraire (première ancre rencontrée)

```
Punct1 Punct2: Punct1 [3] { ‘:’:‘ ‘‘:‘‘:’’ (:(:) }
```

– Ancre principale (`V`) mais non discriminante

```
N V: V [262] { give:demonstration:give give:description:give give:*
               make:* do:* ... }
```

Lexicalisation (remplace Lemme)

```
<lexicalization tree_id="family[@name=tn1pn2]">
    <anchor node_id="v_"> <lemmaref cat="v" name="*DONNER*"/> </anchor>
    <anchor node_id="p_2"> <lex>à</lex> </anchor>
    <equation node_id="np_0" type="top">
        <fs> <f name="restr"><val>plushum</val></f> </fs>
    </equation>
    <equation node_id="np_2" type="top">
        <fs> <f name="restr"><val>plushum</val></f> </fs>
    </equation>
</lexicalization>
```