Statically Typed XML an XTATIC Experience

François Garillot

University of Pennsylvania

École Normale Supérieure

Xtatic

Xtatic : une extension de C# pour le XML statiquement typé

- descendant de la famille de XDUCE
- Objectifs :
  - simple : à utiliser et à comprendre
  - flexible : utilisation de valeurs de type partiellment connu
  - intégration avec C#, de syntaxe légère
- Actuellement
  - Compilateur de source à source
  - Plusieurs applications : *Caml Weekly News*, requêtes vers la *Library of Congress* ...
  - nouvellement : internationalisation

Un exemple

Un simple carnet d'addresses :

```
<person>
  <name>Haruo Hosoya</name>
  <email>hahasoya</email>
</person>
<person>
  <name>Jerome Vouillon</name>
  <tel>123</tel>
</person>
```

```
[[ <person>
     <name>‘Haruo Hosoya‘</name>
     <email>‘hahasoya‘</email>
   </person>
   <person>
     <name>‘Jerome Vouillon‘</name>
     <tel>‘123‘</tel>
   </person> ]]
```

Principes

- types de XML basés sur une grammaire régulière
- tags des éléments de type C#
- contenu d'un élément défini par les types de ses sous-arbres
- système de types généré par une grammaire : opérations de
  - concaténation,
  - itération (*,?),
  - alternative,
  - tags explicites
- les non-terminaux sont les types définis
- les terminaux sont les données textuelles (pcchar, pcdata)

Pattern matching et autres fonctionnalités

- définissable à l’aide de la même grammaire
- structures efficaces : stamping pour downcasting rapide
- bons résultats contre ses concurrents
- attributs simplement (non) typés

Types

```
regtype Name
regtype Tel
regtype Email
regtype TPers
regtype APers
regtype carnet
regtype contact
```

```
[[ <name>pcdata</> ]]
[[ <tel>pcdata</>  ]]
[[ <email>pcdata</> ]]
[[ <person> Name, Tel?, Email* </> ]]
[[Apers*]]
[[Tel | Email]]
```

An example in XTATIC

```
static [[ TPers* ]] addrbook ([[ APers* ]] ps)
 [[ TPers* ]] res = [[ ]];    bool cont = true;
 while (cont)
   match (ps)
    case [[<person> <name>any</> n, <tel>any</> t, any </>, any rest]]:
      res = [[ res, <person> n, t </> ]];   ps = rest;
    case [[ <person> any </person>, any rest ]]:     ps = rest;
    case [[ ]]:    cont = false;
 return res;
```

Une expérience

- Le XML typé : un sujet concurrentiel (XDuce, CDuce, XACT, XQuery, ...)
- basés sur l'idée qu'un système de types riche est un bénéfice significatif
- attirant, mais ouvert au questionnement :
  - des idiomes classiques faciles à typer ?
  - des structures courantes difficiles à typer ?
  - trop de génération de code ? "boilerplate ?"
  - Comment traite-t-on les mises à jour ?

Le format XMLSPEC

- un format externe, réellement utilisé
- le traitement de documents structurés, de manière similaire à DocBook
- la tâche : de XMLSPEC à XHTML
- un travail conséquent : 102 éléments, 57 entités pour XMLSPEC, 89 éléments, 65 entités pour XHTML

Le fonctionnement du XSLT

- Organisation en templates, appelés ou reconnaissant un élément de la page.
- Templates agissant sur la base d'expressions XPath
- récursion implicite.

Modèle de fonctionnement

- ```
  <xsl:template match="abstract">
     <h2><a name="abstract">Abstract</a></h2>
     <xsl:apply-templates/>
  </xsl:template>
  ```

- ```
  static [[h_block+]] TemplateAbstract ([[s_abstract]] markup)
     [[<abstract> s_hdr_mix* hdrmix</>]] = markup;
     return [[<h2>
                <a name=‘abstract‘>‘Abstract‘</>
              </>,
              DispatchContentBlockable(hdrmix)]];
  ```

Typer une feuille de style XSLT

- définir une transition hétérogène
  - un morphisme de types $\mathtt{f}$ de XMLSPEC vers XHTML
  - un morphisme des valeurs $\mathtt{g}$ tel que pour tout $\mathtt{t}$ de type T, $\mathtt{g}(\mathtt{t})_{\mathsf{T}} \in \mathtt{f}(\mathsf{T})$
- le morphisme des valeurs est donné
- certains types sont précisés
- d'autres ne sont pas explicites : à inférer

A partial order relation

Let's consider the grammar $\mathrm{G} = (\mathrm{A}, \mathrm{V}, \mathrm{P})$ of the types that can be specified with the operations of our type algebra. Let's define the $\leq$ relation as:

$$(\mathtt{S} \leq \mathtt{T}) \Leftrightarrow \exists\ \alpha, \beta \in (\mathrm{A} + \mathrm{V})^*, \mathtt{S} \rightarrow^* \alpha \mathtt{T} \beta$$

Two properties of this stylesheet:

- For all $\mathtt{t}$ of type T, the type of $\mathrm{g}(\mathtt{t})_{\mathrm{T}}$ is unique and equal to $\mathrm{f}(\mathrm{T})$.
- The type algebra is the same as the value algebra.

Les templates vus comme extension de la grammaire d'arrivée

- Si un template définit :

$$\mathtt{g}(\mathtt{a})_{\mathtt{A}} = \mathtt{h}_{\mathtt{a}}, \mathtt{g}(\mathtt{b})_{\mathtt{B}}, \mathtt{g}(\mathtt{c})_{\mathtt{C}}$$

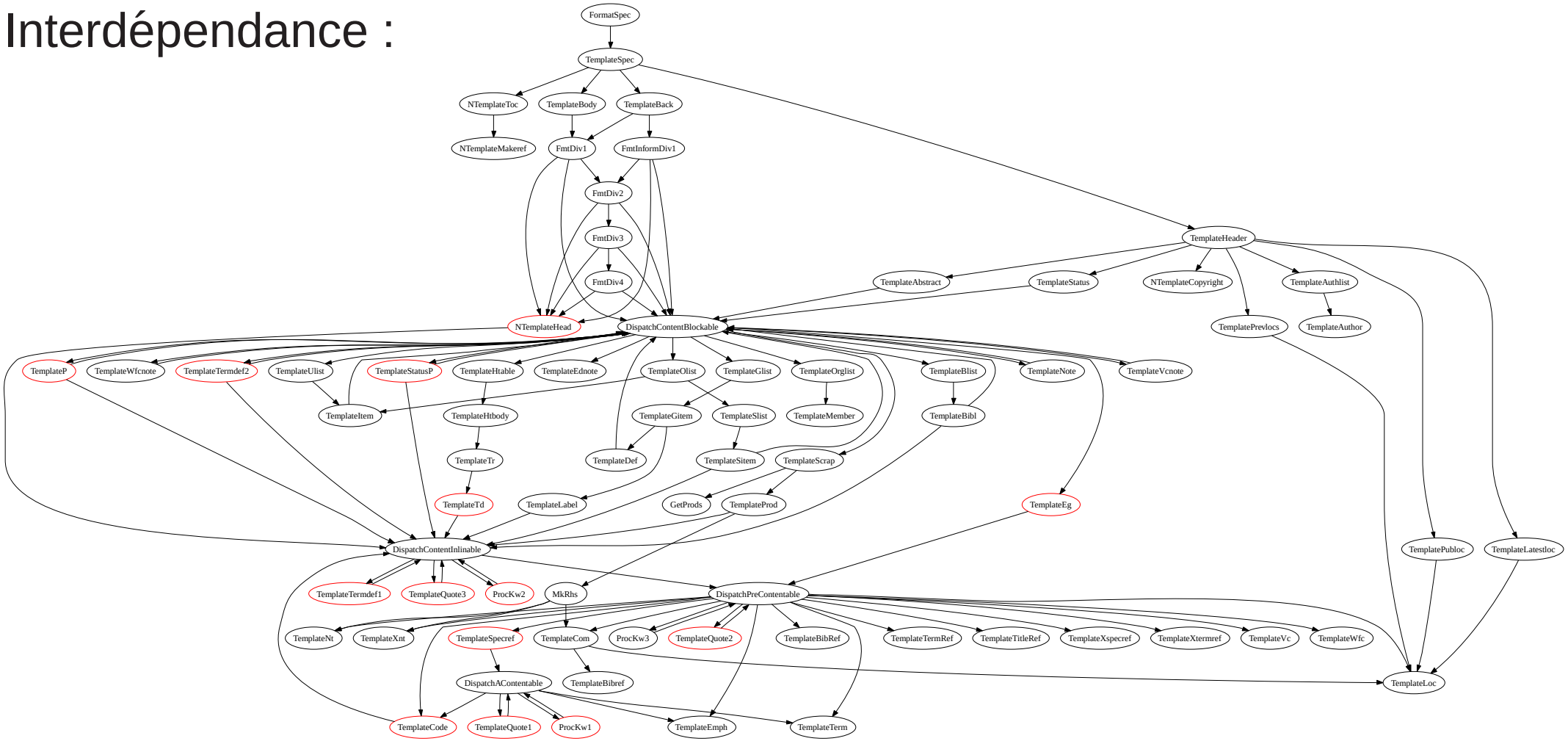
- On peut inférer :

$$\mathtt{f}(\mathtt{A}) = \mathtt{h}_{\mathtt{a}}, \mathtt{f}(\mathtt{B}), \mathtt{f}(\mathtt{C})$$

- $\rightarrow$ de nouvelles règles de production !

Graphe des appels

Interdépendance :



Inférence de type

- Mélange d’approches top-down et bottom-up.
- l’inférence bottom-up demandait une idée du tri topologique du graphe des appels.
- L’inférence top-down se repose sur le compilateur, mais peut être plus lente.
- 11 bugs de typage.
- Trois bugs confrontés aux données : modification du contenu du formatage.

Correction des bugs

- Expérimentalement : rupture de la relation d'ordre.

$$\mathtt{A} \leq \mathtt{B} \text{ et } \mathtt{f}(\mathtt{B}) \leq \mathtt{f}(\mathtt{A})$$

-

$$\mathtt{f}(\mathtt{A}) \leftarrow \mathtt{f}(\mathtt{B})$$

  d'où modification de $\mathtt{g}(\mathtt{B})$ (fonction de coercion)

- amélioration : définition de plusieurs types pour un élément XML

- ```
  TemplateTermdef1 : [[<termdef> inlinable+ </>]] -> [[a,Inline]]
  TemplateTermdef2 : [[<termdef> (inlinable|blockable)* </>]]
          -> [[block*]]
  ```
  ```

L’expansion de la structure du document

Le poids d’un parcours récursif explicite : la feuille contient

- Une table des matières
- Des références croisées
- Des numérotations internes

Autres aspects intéressants :

- fonctions prédéfinies : traitement de texte
- `vertical axis` : les expressions XPath

Conclusion

- Un programme encore long (2500 vs. 750 lignes)
- beaucoup de *self-control* nécessaire : graphe des appels touffu
- fonctionne sur des documents réels (XML, XPath, XQuery...)
- à un facteur 5 en vitesse de XSLT
- Code plus simple à comprendre
  - récursion explicite
  - langage habituel
  - rien n'est "sous le capot"
- garanties de conformité

Future Work

- Profiling : utilité des structures complexes.
- Etude d'autres applications en XSLT(étude d'expressions XPath)
- cahier des charges d'une aide au typage ?
- interpréteur XSLT

Une impression sur XTATIC

- nombreux comportements surprenants du compilateur
- plusieurs aspects objets manquants
- incréments simples : variables de type, commentaires
- polymorphisme ?
- Un langage très agréable à utiliser
- des expressions régulières bien utiles
- structures de contrôle riches : foncitonne avec du XML “atypique”
- bonne intégration avec les chevaux de bataille de l’OO : WebServices, ASP

Perspectives pour XTATIC

- à paraître dans un futur rapport de l'Université de Pennsylvanie
- peut-être PLAN X 2005 ?
- Développement de XTATIC ralenti ...
- beaucoup de choses à faire
- mais de nombreuses perspectives dans un domaine actif