Generating SGML specific editors
from DTDs to Attribute Grammars

José Carlos Ramalho
Alda Reis Lopes
Pedro Rangel Henriques

epl@di.uminho.pt

Contents

• Behind the scene
• The main goal
• Attribute grammars: why and what
• DTD ⇒ AG conversion
• Future work
Behind the scene

• Last year conference (Washington)
  – Semantic Validation: the possibility to add constraints and context conditions

• Markup Languages Journal
  – Processing constraints
    • type inference
    • value normalization

Type inference

Problem: how to process ... ?

Constraint: latitude > 39 and latitude < 42.5

Document: ...

Answer:

...<latitude type="float">41.32</latitude>...
Value normalization

Problem: How can I identify ...?

... King <name> Affonso </name> proclaimed several ...
... And his soldiers battled against <name> Afonso </name>
... and that church was built in the <date> XVIII century </date>
... it all happened on <date> the fifth October </date> ...

Answer

... King <name value=" Afonso "> Afonso </name> ...
... it all happened on <date value=" xxxx.10.05 "> the fifth... 

Programs ⇔ SGML Documents

- Have a support language formally defined
- Processing - compiler
  - lexical analysis
  - syntactic analysis
  - semantic analysis
    - complex: type checking; type inference, ...
    - Can be formally specified: Attribute Grammars
- Have a support markup language defined in SGML
- Processing - parser
  - lexical analysis
  - syntactic analysis
  - semantic analysis
    - very simple: ID - IDREF coupling
INES - Document Programming Environment

INES

Designer

Constraints

Style Specification

Texto “Y” Texto “X” Texto “Z”

Utilizador A Utilizador B Utilizador C

INES: inside

Designer

DTD Editor

Scheme code

SGML text

DTD

Constraints

Context Editor

Style Specification

DSSSSL Editor

“X” Editor

RTF

PostScript

Doc X

Errors

working
projected

Utilizador
Why Attribute Grammars?

The Chomsky Hierarchy

Grammar level

Algorithm complexity

- Chomsky Hierarchy
  - 0 - unrestricted
  - 1 - Context-Sensitive
  - 2 - Context-Free
  - 3 - Regular

A grammar is classified by the highest Chomsky level it fits
Attribute Grammars

To have a level 2 grammar with level 1 expressive power!

Interesting languages have context!

- Is this element already defined?
- Are open brackets paired with closing ones?
- Does this variable have the correct type?

• Semantics
  – Context conditions and constraints as attribute equations

• New concepts in compiling
  – Incremental parsing: being able to produce an instance of the AST at any time
  – Incremental evaluation: recompiling only what is really necessary
AG: formal definition

AG = <G, A, R, C>

G is a context free grammar (level 2): G = <T, N, S, P>
- T - set of terminal symbols (alphabet)
- N - set of nonterminal symbols
- S - start symbol or axiom (S belongs to N)
- P - set of derivation rules

A is the set of all attributes: intrinsic, inherited and synthesized

R is the set of attribute evaluation rules

C is the set of all contextual conditions

AG: example

```
DTD → Decls
  Decls.ElemTab = ()

Decls → Dec Decls
  Decls.ElemTab = Decls.ElemTab
  Decls$2.ElemTab = Dec.ElemNewTab
| ε

Dec → ElemDec
  ElemDec.ElemTab = Dec.ElemTab
| AttDec ...

ElemDec → gi min min Content
  if not exist( gi, ElemDec.ElemTab )
  else error("Element already defined!")
```
### AG implementation tool: SGen

- It allows attribute definitions
- It allows attribute equations definitions
- It provides an evaluator for those equations

- Sgen
  - based in a formal language: the interface, attribute equations, ...
  - multiple views of the internal representation

---

**AG: example (cont.)**

![Diagram](Image)
DTD2AG conversion

- Elements and Attributes
  - Content models
    - connectors
    - occurrence indicators
  - SGML “specials”

The AG internal representation is very similar to a grove. Disjoint things in SGML will have to merge.

Elements and Attributes

SGML:

```xml
<!ELEMENT Letter - - (Date,Name+,Message,End)>
<!ATTLIST Letter Type (work|family|friend) work
Idiom (en|pt|fr|sp) en>
```

S1: Attributes as part of the content model

```
Letter → Date NameList Message End
Letter.type = ...
Letter.idiom = ...
```
Connectors

C1: ,

SGML:

<!ELEMENT Letter - - (Date,Name+,Message,End)>

AG:

Letter -> Date NameList Message End

C2: |

SGML:

<!ELEMENT Letter - - (Date|Name+|Message|End)>

AG:

Letter -> Date | NameList | ...

C3: &

SGML:

<!ELEMENT Letter - - (Date&Name+&Message&End)+>

AG1: all possible combinations

Letter -> Date NameList ...
| NameList Date ...
...

AG2: transform SGML and add constraint

<!ELEMENT Letter - - (Date|Name+|Message|End)>

CC: x in Content, \exists x

Letter -> X
Letter -> X Letter
X -> Date
if( exists( Date.gi, X.ElemTab ) ) then error “...”
| NameList
| Message | End
Occurrence indicators

SGML:
<!ELEMENT Letter - - (Date?|...)>  
AG:
Letter → Date ...  
Date → ε | ...

SGML:
<!ELEMENT Letter - - (Date|Message|...)*>  
AG:
Letter → ε  
| LetterContent Letter

AG: an open path to semantics
Questions?

New SGML auth. and proc. model