Feature grammar systems. Incremental maintenance of indexes to digital media warehouses

Windhouwer, M.A.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (http://dare.uva.nl)

Download date: 27 Sep 2020
Appendix A

The Feature Grammar Language

This appendix contains the feature grammar language in an EBNF notation, i.e., the notation as used by the W3C for the XML specification [W3C00].

```plaintext
#character classes
digit ::= [0-9]
exponent ::= [Ee][+-]?[D]+
letter ::= [_a-zA-Z]
any ::= [#x0-#xFFFE]

#literals or constants
float ::= '[-+] digit+ . digit+ exponent?
integer ::= '[-+] digit+
unsigned-integer ::= digit+
string ::= "" any* ""
constant ::= float | integer | string

# a symbol
symbol ::= letter ( letter | digit )*

# a scope
scope ::= letter ( letter | digit )*

# the prefix puts an symbol in a scope, this scope
# may refer to a feature grammar, an ADT module or
# a detector plugin
prefix ::= scope "::"

# simplified XPath expression
```
Appendix A: The Feature Grammar Language

\[
\text{xpath} ::= \text{absolute} \mid \text{relative} \\
\text{absolute} ::= '/' \text{relative} \mid '//' \text{relative} \\
\text{relative} ::= \text{step} ( '/' \text{step} \mid '//' \text{step} )^* \\
\text{step} ::= \text{axis} \? ( ( \text{symbol} \mid '*' ) \mid \text{dereference} ) \\
\mid \text{abbreviation} \\
\text{axis} ::= 'self::' \mid 'parent::' \mid 'child::' \\
\mid 'ancestor::' \mid 'ancestor-or-self::' \\
\mid 'preceding::' \mid 'preceding-sibling::' \\
\mid 'descendant::' \mid 'descendant-or-self::' \\
\mid 'following::' \mid 'following-sibling::' \\
\text{abbreviation} ::= '.' \mid '..' \\
\# a feature grammar specific addition \\
\text{dereference} ::= '&symbol' \\
\text{detector-params} ::= specialised XPath expression \\
\# a list of detector parameters, if no axis is specified for 
\# the first step it defaults to preceding: \\
\text{detector-params} ::= '(', ( \text{detector-param} ( ',',' \\
\text{detector-param} )* ) ')' \\
\text{detector-param} ::= constant \mid xpath \\
\# even more simplified XPath expression \\
\text{s-path} ::= \text{s-absolute} \mid \text{s-relative} \\
\text{s-absolute} ::= '/' \text{s-relative} \mid '//' \text{s-relative} \\
\text{s-relative} ::= \text{s-step} ( '/' \text{s-step} \mid '//' \text{s-step} )^* \\
\text{s-step} ::= \text{s-axis}? ( \text{symbol} \mid '*' ) \\
\text{s-axis} ::= 'self::' \mid 'child::' \\
\mid 'ancestor::' \mid 'ancestor-or-self::' \\
\text{start-params} ::= specialised XPath expression \\
\# a list of start symbol parameters, if no axis is specified 
\# for the first step it defaults to the standard child: \\
\text{start-params} ::= '(', ( \text{start-param} ( ',',' \\
\text{start-param} )* ) ')' \\
\text{start-param} ::= \text{s-path} \\
\# collection type and bounds specification for symbols on the 
\# right-hand side of a rule \\
\text{bounds} ::= \text{list} \mid \text{set} \mid \text{tuple} \\
\text{list} ::= '[' \text{range} ']' \mid \text{range} \\
\text{set} ::= '{' \text{range} '}' \\
\text{tuple} ::= '<' \text{int-range} '>' \\
\text{range} ::= \text{wild-range} \mid \text{int-range} \\
\text{int-range} ::= \text{unsigned-integer} \\
\text{wild-range} ::= '*' \mid '+' \mid '?' \\
\# the feature grammar language, i.e. the start symbol of this
feature-grammar ::= module-decl decl*
decl ::= use-decl | start-decl
       | poll-decl | atom-decl
       | detector-decl | classifier-decl
       | version-decl | rule-decl

module-decl ::= "%module" scope ';''
use-decl ::= "%use" scope (',' scope)* ';''

start-decl ::= "%start" symbol start-params ';''
poll-decl ::= "%poll" symbol ".poll" start-params ';''

atom and atom rule declarations
atom-decl ::= "%atom" prefix? symbol
            ( (( symbol (',' symbol )*)
              | '{' any+ '}') )? ';''

detector declarations
detector-decl ::= "%detector" prefix? symbol detector-params ';''
detector-decl ::= "%detector" prefix? symbol
               '[' any+ ']' ';''

classifier declaration
classifier-decl ::= "%classifier" prefix? symbol detector-params ';''

version declaration
version-decl ::= "%version" symbol
              unsigned-integer '.'
              unsigned-integer '.'
              unsigned-integer ';''

rule declaration
rule-decl ::= rhs ':' lhs ';''
rhs ::= prefix? symbol
lhs ::= ( '&'? prefix? symbol bounds? | constant )+
lhs ::= '(' lhs ')'
lhs ::= lhs '|' lhs