UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

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

Windhouwer, M.A.

Publication date
2003

Link to publication

Citation for published version (APA):
Windhouwer, M. A. (2003). *Feature grammar systems. Incremental maintenance of indexes to digital media warehouses*. [Thesis, externally prepared, Universiteit van Amsterdam].

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, P.O. Box 19185, 1000 GD 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 (https://dare.uva.nl)

Download date:18 Nov 2026

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].

```
1  #character classes
2  digit              ::= [0-9]
3  exponent           ::= [Ee][-+]?{D}+
4  letter             ::= [_a-zA-Z]
5  any                ::= [#x0-#xFFFE]

6  #literals or constants
7  float              ::= '-'? digit+ '.' digit+ exponent?
8  integer            ::= '-'? digit+
9  unsigned-integer:: = digit+
10 string             ::= '"' any* '"'
11 constant           ::= float | integer | string

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

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

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

20 #simplified XPath expression
```

```
xpath           ::= absolute | relative
absolute        ::= '/' relative? | "//" relative
relative        ::= step ( '/' step | "//" step )*
step            ::= axis? ( ( symbol | '*' ) | dereference )
                  | abbreviation
axis            ::= "self::" | "parent::" | "child::"
                  | "ancestor::" | "ancestor-or-self::"
                  | "preceding::" | "preceding-sibling::"
                  | "descendant::" | "descendant-or-self::"
                  | "following::" | "following-sibling::"
abbreviation    ::= '.' | ".."
#a feature grammar specific addition
dereference     ::= '&' symbol

#a list of detector parameters, if no axis is specified for
#the first step it defaults to preceding::
detector-params ::= '(' ( detector-param ( ','
                          detector-param )* ) ')'
detector-param  ::= constant | xpath

#even more simplified XPath expression
s-path          ::= s-absolute | s-relative
s-absolute      ::= '/' s-relative? | "//" s-relative
s-relative      ::= s-step ( '/' s-step | "//" s-step )*
s-step          ::= s-axis? ( symbol | '*' )
s-axis          ::= "self::" | "child::"
                  | "descendant::" | "descendant-or-self::"

#a list of start symbol parameters, if no axis is specified
#for the first step it defaults to the standard child::
start-params    ::= '(' ( start-param ( ','
                          start-param )* ) ')'
start-param     ::= s-path

#collection type and bounds specification for symbols on the
#right-handside of a rule
bounds          ::= list | set | tuple
list            ::= '[' range ']' | range
set             ::= '{' range '}'
tuple           ::= '<' int-range '>'
range           ::= wild-range | int-range
int-range       ::= unsigned-integer
                    ( ':' unsigned-integer )?
wild-range      ::= '*' | '+' | '?'

#the feature grammar language, i.e. the start symbol of this
```

```
#EBNF
feature-grammar ::= module-decl decl*
decl            ::= use-decl | start-decl
                  | poll-decl | atom-decl
                  | detector-decl | classifier-decl
                  | version-decl | rule-decl

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

#start declaration
start-decl      ::= "%start" symbol start-params ';'

#poll declaration
poll-decl       ::= "%detector" 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
```