# Grammars and signatures

**Magne Haveraaen**
**Universitet i Bergen**

**Inf329 – Selected topics in programming theory**

**2006-02-03**

# Grammar

A **context free grammar** CFG(T,N,R,n) consists of

| | |
|---|---|
| T | - set of terminal symbols |
| N | - set of non-terminal symbols, N disjoint from T |
| $R \subseteq N \times (N \cup T)^*$ | - set of rules (productions, lhs $\times$ rhs) |
| $n \in N$ | - start symbol |

and defines a set of syntax trees

**Syntax tree** for CFG(T,N,R,n): a tree with

- rules R as nodes
- every rhs symbol *s* has exactly one sub-tree with root node lhs=*s*
- root node where lhs=n, the start symbol

**Pattern**: a syntax (sub-)tree which may lack some sub-trees
**Word**: string of terminals given by an inorder traversal of a syntax tree
**Context free language**: the set of all words from a context free grammar CFG(T,N,R,n)
**Parsing**: (re)creating a syntax tree from a word

 

# Signature Grammar

**Signature Grammar** Sig(Sset,Mset,n) = CFG(T<Sset,Mset>,N,R<Sset,Mset>,n)

Sset - set of sort names

Mset - set of method names

T<Sset,Mset> = { ":", "→", ",", "**obs**", "**upd**", "**giv**", "**ded**" } ∪ Sset ∪ Mset

N = { S, MNAME, MODE, RMODE, P, R, F, G, H, M, N, L }

R<Sset,Mset> = {

| | |
|---|---|
| S ::= Sset↓ | MNAME ::= Mset↓ |
| MODE ::= "**obs**" \| "**upd**" | RMODE ::= "**obs**" \| "**upd**" \| "**giv**" \| "**ded**" |
| P ::= MODE S | R ::= RMODE S |
| F ::= MNAME ":" S* "→" S | G ::= MNAME ":" S* "→" S* |
| | H ::= MNAME ":" ("**obs**" S)* "→" S* |
| M ::= MNAME ":" P* | N ::= MNAME ":" P* "→" S |
| | L ::= MNAME ":" R* "→" S* |

}

n ∈ { F, G, H, M, N, L }

**Signature**: **sorts** Sset, **methods** subset of the syntax trees of a signature grammar

# Signatures

**Algebraic signatures**

| | | |
|---|---|---|
| plain functions: | Sig(Sset,Mset,F) | m : s1, s2, …, sk → s |
| multivalued: | Sig(Sset,Mset,G) | m : s1, s2, …, sk → s1', …, sn' |
| resourced: | Sig(Sset,Mset,H) | m : **obs** s1, **obs** s2, …, **obs** sk → s1', …, sn' |

Partial projection functions p_i:Sig(Sset,Mset,G) → Sig(Sset,Mset,F)

p_i( m : s1, s2, …, sk → s1', …, sn' ) = m_i : s1, s2, …, sk → si'

The syntax trees of Sig(Sset,Mset,H) are isomorphic to those of Sig(Sset,Mset,G)

**Method signatures**

| | | |
|---|---|---|
| imperative: | Sig(Sset,Mset,M) | m : **obs** s1, **upd** s2, …, **upd** sk |
| with return: | Sig(Sset,Mset,N) | m : **obs** s1, **upd** s2, …, **upd** sk → s |
| general: | Sig(Sset,Mset,L) | m : **obs** s1, **giv** s2, …, **upd** sk → s1', …, sn' |

The syntax trees of Sig(Sset,Mset,H), Sig(Sset,Mset,M) and Sig(Sset,Mset,N) are subsumed by those of Sig(Sset,Mset,L)

# Plain Expressions

Expression-terms from plain function signatures

Σ - a plain function signature given by Sig(Sset,Mset,F), Sset=**sort**(Σ)
X - an Sset-indexed family of variable names X_s for each s∈Sset
S - a sort in Sset

- ExpTerm<Σ,X,S> = CFG(T<Σ,X>, **sort**(Σ), R<Σ,X>, S)

  T<Σ,X> = ∪X ∪ { "(", ",", ")" } ∪ { m∈Mset | (m : s1, s2, …, sk → s) ∈ **methods**(Σ) }

  R<Σ,X> = {
  s ::= X_s↓ ∀ s∈Sset
  s ::= m "(" s1 "," s2 "," … "," sk ")" ∀ (m : s1, s2, …, sk → s) ∈ **methods**(Σ)
  }

# Weaker Typed Plain Expressions

Expression-terms from plain function signatures

Σ - a plain function signature given by Sig(Sset,Mset,F), Sset=**sort**(Σ)
X - an Sset-indexed family of variable names X_s for each s∈Sset

- CountTerm <Σ,X> = CFG(T<Σ,X>, { S }, RC<Σ,X>, S)

  RC<Σ,X> = {
  S ::= ∪_s∈Sset X_s↓
  S ::= m "(" S "," S "," … "," S ")" ∀ (m : s1, s2, …, sk → s) ∈ **methods**(Σ)
  }

- WeakTerm <Σ,X> = CFG(T<Σ,X>, { S }, RW<Σ,X>, S)

  RW<Σ,X> = {
  S ::= ∪_s∈Sset X_s↓
  S ::= m "(" S* ")" ∀ (m : s1, s2, …, sk → s) ∈ **methods**(Σ)
  }

 

# General Imperative Expressions

Imperative terms from general imperative signatures

Σ, X, S - as before, given by Sig(Sset,Mset,N), Sset=**sort**(Σ)

- ImpTerm<Σ,X,S> = CFG(T<Σ,X>, N<Sset>, R<Σ,X>, S)

  T<Σ,X> = ∪X ∪ { "(", ",", ")", ":=", ";", "**return**" }

  ∪ { m∈Mset | ( m : **obs** s1, …, **upd** sk → s1', …, sn' ) ∈ **methods**(Σ) }

  N<Sset> = Sset ∪ { E<s>, V<s> | ∀ s∈Sset } ∪ { STM }

  R<Σ,X> = { E<s> ::= X_s V<s> ::= X_s

  s ::= STM* "**return**" E<s> ";" ∀ s∈Sset

  STM ::= "(" V<s1'> "," … "," V<sn'> ")" ":=" m "(" mo1' ", … "," mok' ")" ";"

  ∀ ( m : mo1, mo2, …, mok → s1', …, sn' ) ∈ **methods**(Σ),

  moi'=E<si> if moi="**obs**" si, moi'=V<si> otherwise

  E<s> ::= m "(" mo1' ", … "," mok' ")"

  ∀ ( m : mo1, mo2, …, mok → s ) ∈ **methods**(Σ),

  moi'=E<si> if moi="**obs**" si, moi'=V<si> otherwise

  }