Semantiline analüüs

Semantiline analüüs

- Semantiline analüüs kontrollib programmi kontekstuaalsete sõltuvuste korrektsust:
  - leiab vastavuse defineerivate ja kasutusesinemiste vahel,
  - leiab esinemiste tüübid ja kontrollib nende vastavust reeglitele,
  - ...
- Süntaksipuu dekoreeritakse tüübi- ja muu kontekstist sõltuva infoga.

Semantiline analüüs

- Semantiline analüüs tegeleb staatilise semantika poolt pandud kitsenduste kontrollimisega.
- Mõnikord on nende kontrollimine osaliselt saavutatav ka kontekstivabade grammatikatega, aga reeglina muudab see grammatika loetamatuks või kitsendab keelt väga tugevalt.
- Näide – lihtsalt tüübitud avaldised:

$$
\begin{array}{lcl}
\text{IntExp} & \rightarrow & \mathit{int} \mid \mathit{intVar} \\
& \mid & \text{IntExp} + \text{IntExp} \\
\text{BoolExp} & \rightarrow & \mathit{true} \mid \mathit{false} \\
& \mid & \mathit{boolVar} \\
& \mid & \text{IntExp} \leq \text{IntExp} \\
& \mid & \mathit{not}\ \text{BoolExp} \\
& \mid & \text{BoolExp}\ \&\ \text{BoolExp}
\end{array}
$$

Semantiline analüüs

- Toodud grammatika tundub esmapilgul igati mõistlik, aga:
  - kogu skeem baseerub tugevalt sellel, et muutujad on eraldatud kahte klassi – täisarvulised- ning tõeväärtusmuutujad;
  - rohkemate tüüpide korral tuleb ka muutujahulki rohkem tükeldada;
  - samas, enamik programmeerimiskeeli lubab ei sea muutujanimedele tüübi järgi kitsendusi;
  - veelgi enam, reeglina on lubatud erinevas kontekstis kasutada sama nime erinevat tüüpi muutujate tähistamiseks.

Atribuutgrammatikad

- Atribuutgrammatika (attribute grammar) on kontekstivaba grammatika üldistus, kus:
  - iga grammatika sümboliga on seotud mingi hulk atribuute;
  - iga produktsioonireegliga on seotud vastavate atribuutide arvutamise reeglid (nn. semantilised reeglid).

- Eesmärk on iga süntaksipuu korral leida semantiliste reeglitega kooskõlas olev atribuutide väärtustus.

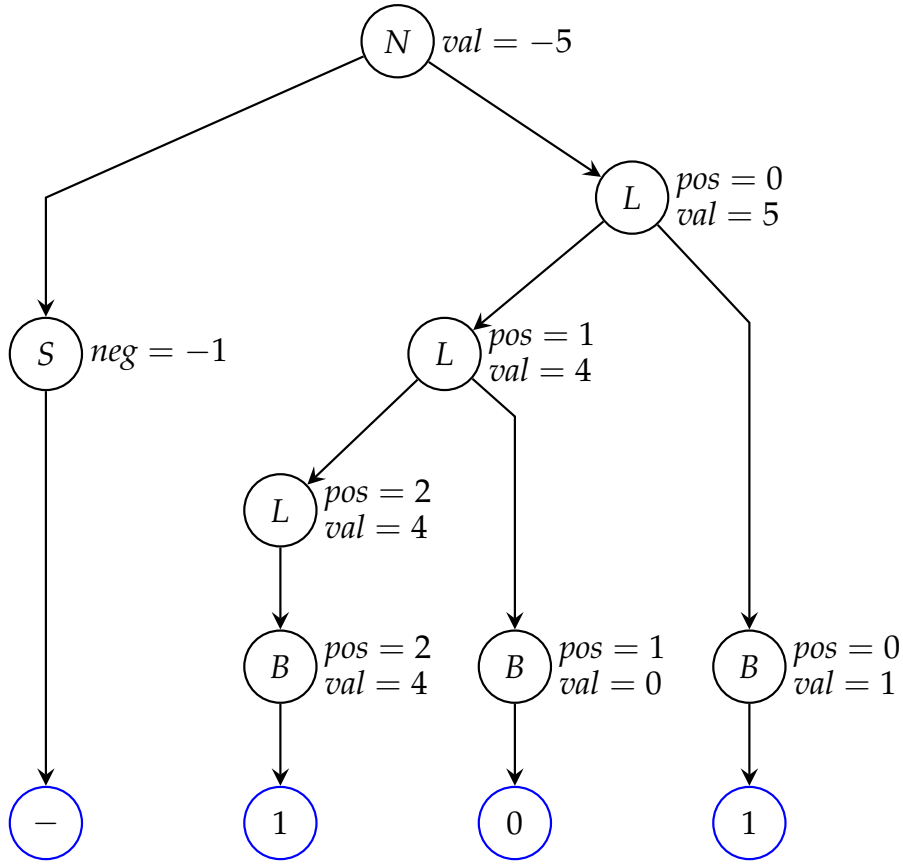
Atribuutgrammatikad

Näide:

Produktsioonid	Semantilised reeglid
$N \rightarrow S\ L$	$L.pos := 0$ $N.val := S.neg * L.val$
$S \rightarrow +$	$S.neg := 1$
$S \rightarrow -$	$S.neg := -1$
$L \rightarrow L_1\ B$	$L_1.pos := L.pos + 1$ $B.pos := L.pos$ $L.val := L_1.val + B.val$
$L \rightarrow B$	$B.pos := L.pos$ $L.val := B.val$
$B \rightarrow 0$	$B.val := 0$
$B \rightarrow 1$	$B.val := 2^{B.pos}$

Atribuutgrammatikad

Näide:



Atribuutgrammatikad

- Produktsiooniga $A \to \alpha$ seotud semantilised reeglid on kujul $y = f(x_1, \ldots, x_n)$, kus y ja x_i on mingile grammatika sümbolile kuuluvad atribuudid ning f on funktsioon.

- Eristatakse kahte liiki atribuute:
  - sünteesitud (synthesized) atribuudid: y on sümboli A atribuut;
  - päritud (inherited) atribuudid: y on mingi produktsioonireegli parempooles α esineva sümboli atribuut.

- Sünteesitud atribuutide väärtused sõltuvad ainult otseste alampuude juurtippude atribuutide väärtustest.

- Päritud atribuutide väärtused sõltuvad vahetu ülemtipu ning kõrvaltippude atribuutide väärtustest.

Atribuutgrammatikad

Näide:

Produktsioonid	Semantilised reeglid
$N \rightarrow S\ L$	$L.pos := 0$ $N.val := S.neg * L.val$
$S \rightarrow +$	$S.neg := 1$
$S \rightarrow -$	$S.neg := -1$
$L \rightarrow L_1\ B$	$L_1.pos := L.pos + 1$ $B.pos := L.pos$ $L.val := L_1.val + B.val$
$L \rightarrow B$	$B.pos := L.pos$ $L.val := B.val$
$B \rightarrow 0$	$B.val := 0$
$B \rightarrow 1$	$B.val := 2^{B.pos}$

sünteesitud atribuudid

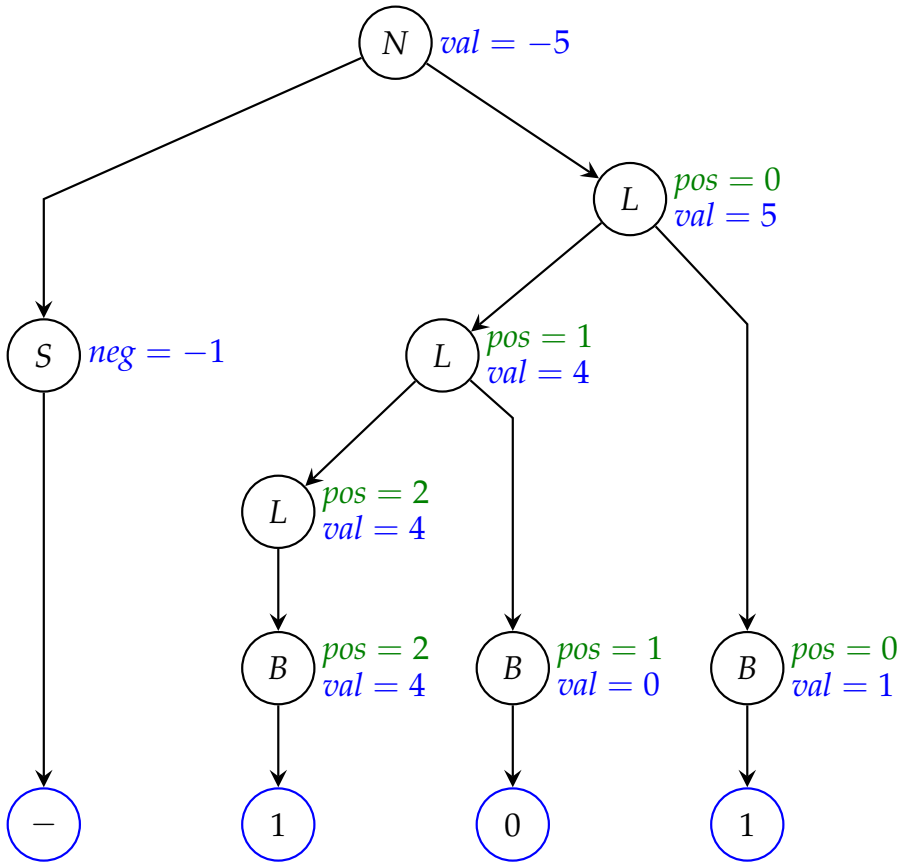
päritud atribuudid

Atribuutgrammatikad

- Kui atribuudi a väärtuse arvutamiseks on tarvis atribuudi b väärtust (a sõltub b-st), siis tuleb b väärtustada enne kui a.
- Atribuutide vahelised sõltuvused defineerivad atribuutide väärtustamiseks sõltuvusgraafi (dependency graph):
  - suunatud graaf, kus servad näitavad atribuutide vahelisi sõltuvusi;
  - kirjeldab andmete voogu atribuutide väärtustamisel.
- Sünteesitud atribuutide korral on servad suunatud alt üles.
- Päritud atribuutide korral on servad suunatud ülalt alla ja/või vasakult paremale ning paremalt vasakule.

Atribuutgrammatikad

Näide:



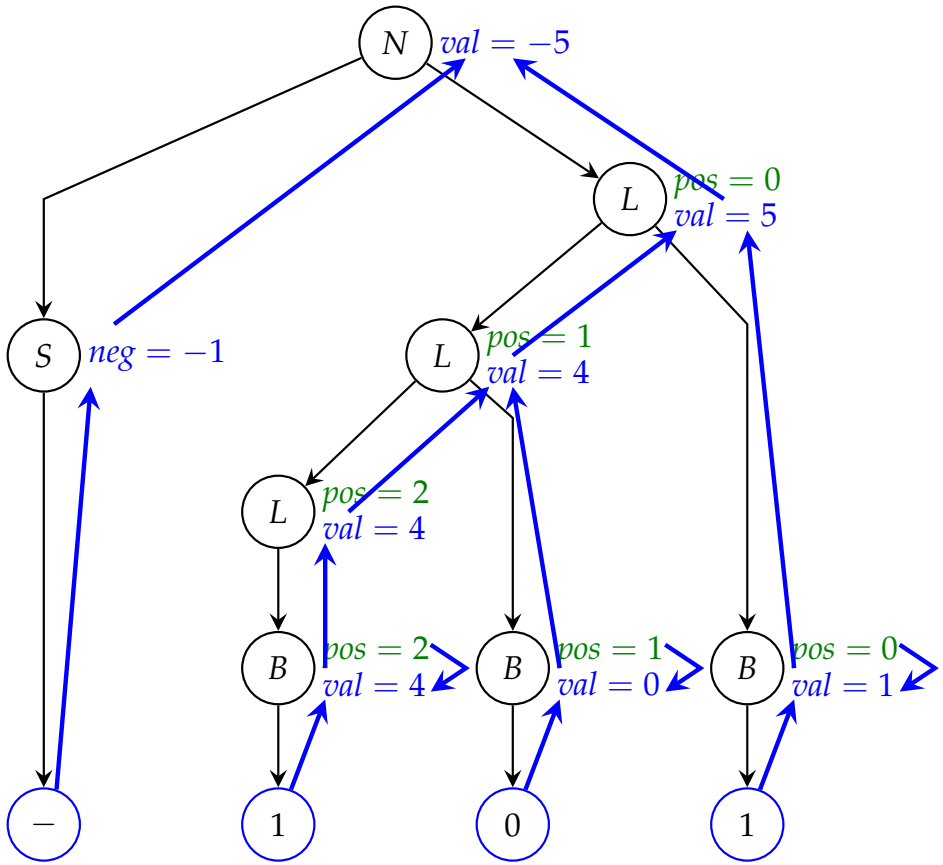
sõltuvusgraaf

sünteesitud atribuudid

päritud atribuudid

Atribuutgrammatikad

Näide:



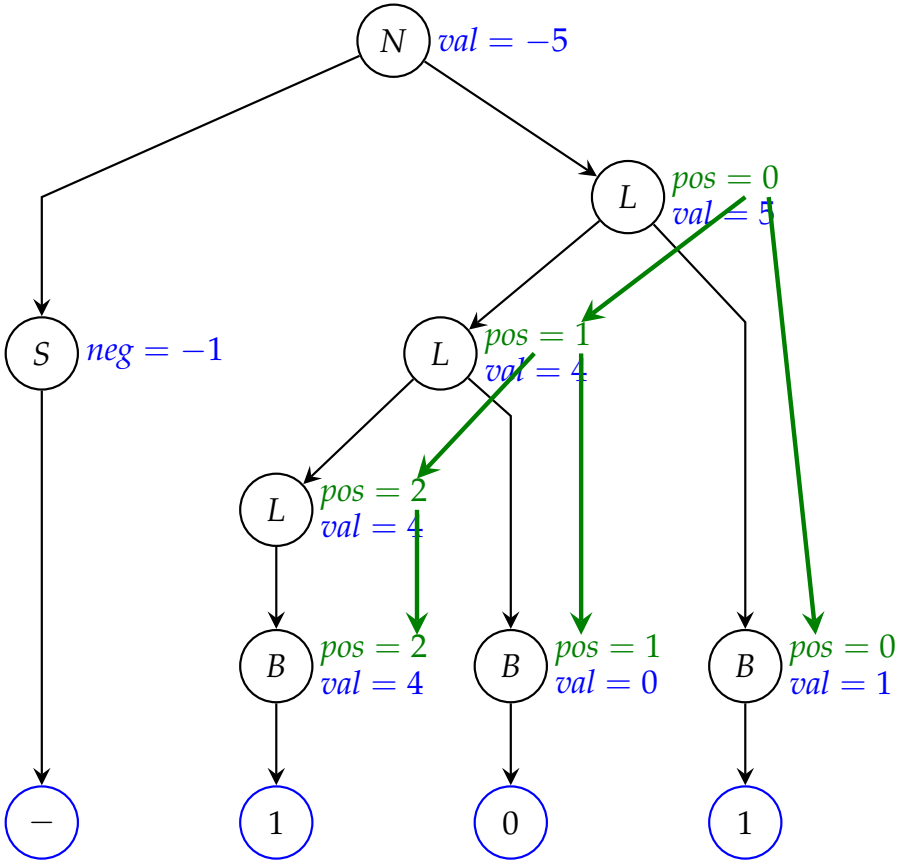
sõltuvusgraaf

sünteesitud atribuudid

päritud atribuudid

Atribuutgrammatikad

Näide:



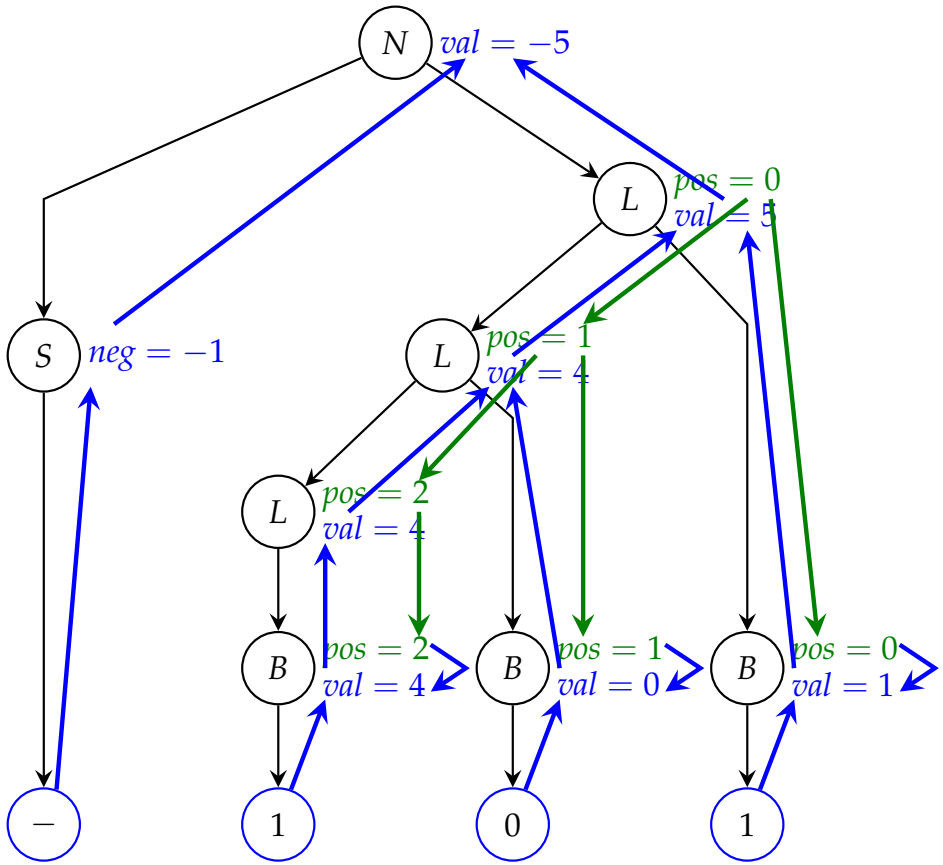
sõltuvusgraaf

sünteesitud atribuudid

päritud atribuudid

Atribuutgrammatikad

Näide:



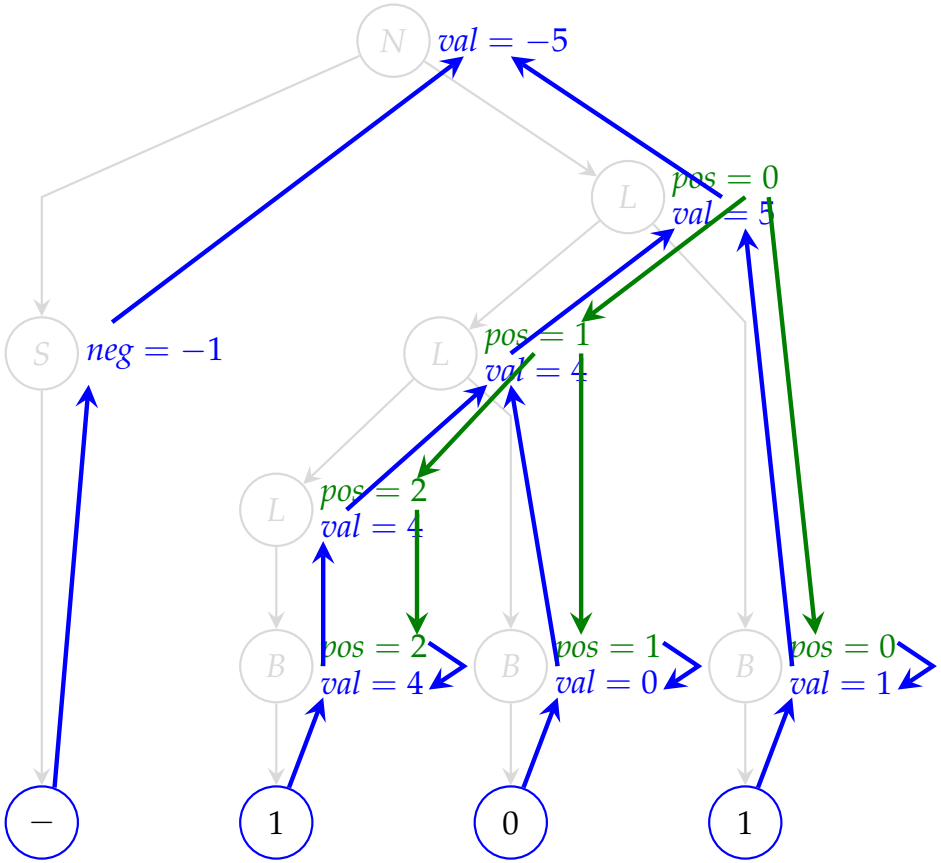
sõltuvusgraaf

sünteesitud atribuudid

päritud atribuudid

Atribuutgrammatikad

Näide:



sõltuvusgraaf

sünteesitud atribuudid

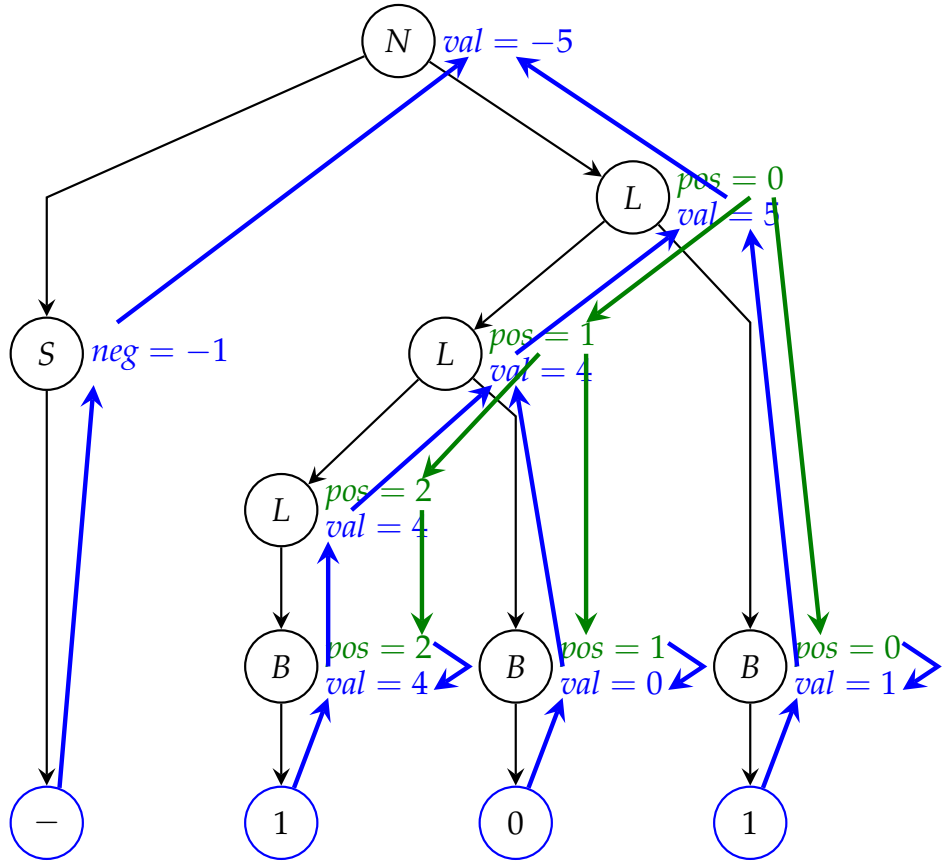
päritud atribuudid

Atribuutgrammatikad

- Tsükliteta suunatud graafi topoloogiliseks sorteerimiseks nimetatakse graafi tippudel $n_1, \ldots, n_k$ sellise järjestuse leidmist, et iga kaare $n_i \to n_j$ korral $n_i < n_j$.
- Sõltuvusgraafi topoloogiline sorteerimine annab kehtiva järjestuse atribuutide väärtustamiseks.
- NB! Üldjuhul saab väärtustada ainult mittetsüklilisi semantilisi reegleid.

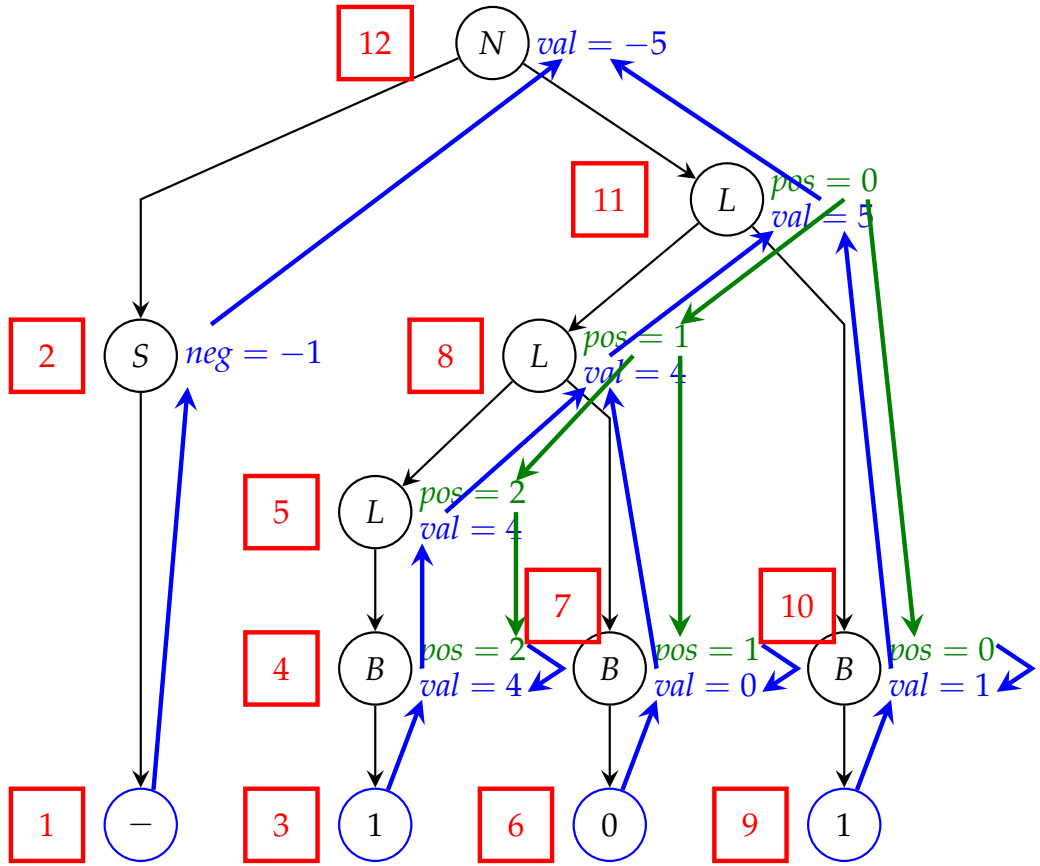
Atribuutgrammatikad

Näide:



Atribuutgrammatikad

Näide:



Atribuutgrammatikad

- S-atribuutgrammatika on selline AG, kus kõik atribuudid on sünteesitud.
- Kuna atribuutide väärtustamine toimub alt üles, siis on S-atribuutgrammatikad sobilikud kasutamiseks koos LR(k) parseritega.
- Atribuutide väärtused saab koos sümboliga salvestada magasini.
- Kui toimub produktsioonireegli $A \rightarrow \alpha$ taandamine, siis on tema parempoole α atribuudid magasini tipus.
- Seega saab taandamise ajal ka arvutada lisatava tipu sünteesitavad atribuudid.

Atribuutgrammatikad

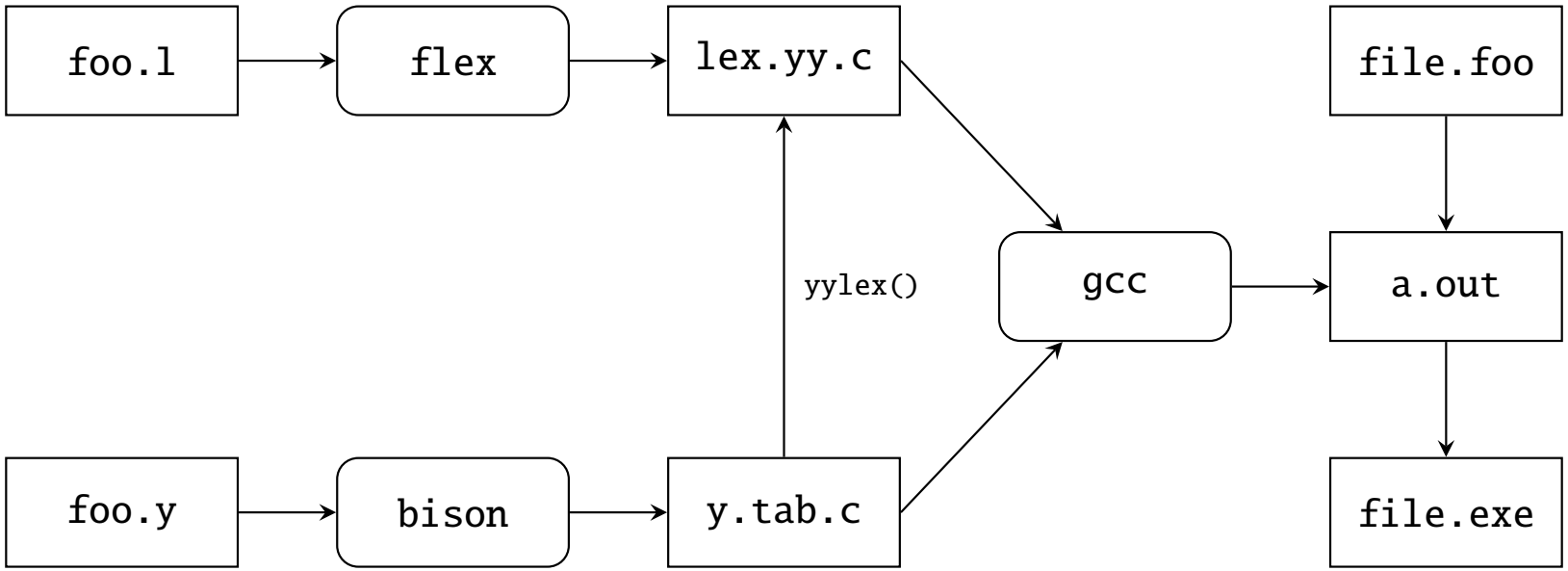
- L-atribuutgrammatika on selline AG, kus iga produktsioonireegli $A \rightarrow X_1 X_2 \ldots X_n$ korral, sümboli X_i $(1 \leq i \leq n)$ kõik päritavad atribuudid sõltuvad ainult A päritavatest atribuutidest ning temast vasakul asuvate sümbolite X_j $(j < i)$ atribuutidest.

- NB! Iga S-atribuutgrammatika on ka L-atribuutgrammatika.

- L-atribuutgrammatikate korral saab atribuute väärtustada sügavuti vasakult-paremale järjekorras.

- Sobilikud kasutamiseks koos LL(k) parseritega (nii rekursiivselt laskuvate kui tabeljuhitavatega).

Parserite generaator Bison



Parserite generaator Bison

Sisendfaili formaat:

- Bison-i sisendfail koosneb kolmest osast:

```
definitions
%%
rules
%%
user code
```

- Sama üldstruktuur mis Flex-il.

Parserite generaator Bison

```
%{
#include <stdio.h>
%}
%token INTEGER
%%
program:
    | program expr '\n'  { printf("%d\n", $2); }
    ;
expr:
    INTEGER              { $$ = $1; }
    | expr '+' expr      { $$ = $1 + $3; }
    | expr '-' expr      { $$ = $1 - $3; }
    ;
%%
void yyerror(char *s) {
    fprintf(stderr, "%s\n", s);
}
int main(void) {
    yyparse();
    return 0;
}
```

Parserite generaator Bison

```
%{
#include <stdio.h>
%}
%token INTEGER
%%
```

Definitsioonide osa koosneb:

- `%{` ja `%}` vahel asuv C kood, mis kopeeritakse otse genereeritavasse faili;
- Bison-i deklaratsioonid:
  - `%token` terminaalsümbolite loend (tarvilikud nii produktsioonireeglites, kui skanneris);
  - `%start` algsümboli määramine (kui puudub, siis on algsümboliks esimene mitteterminal);
  - `%union`, `%left`, `%right`, ...

Parserite generaator Bison

```
program:
    | program expr '\n'  { printf("%d\n", $2); }
    ;
expr:
    INTEGER              { $$ = $1; }
    | expr '+' expr      { $$ = $1 + $3; }
    | expr '-' expr      { $$ = $1 - $3; }
    ;
```

- Teine osa koosneb produktsioonireeglitest.
- Peab olema defineeritud vähemalt üks produktsioonireegel.
- Reeglite parempooled koosnevad terminalidest ja mitteterminalidest ning võivad sisaldada aktsioone.
- Mitteterminalid võivad olla nii eelnevalt deklareeritud, kui ka üksikmärgid (mida pole vaja eeldeklareerida).

Parserite generaator Bison

```
program:
    | program expr '\n'  { printf("%d\n", $2); }
    ;
expr:
    INTEGER              { $$ = $1; }
    | expr '+' expr      { $$ = $1 + $3; }
    | expr '-' expr      { $$ = $1 - $3; }
    ;
```

- Aktsioonid on loogeliste sulgudega ümbritsetud C kood.
- Vastavad semantilistele reeglitele.
- Võivad viidata grammatika sümbolitega seotud semantilistele väärtustele (sünteesitud atribuutidele):
  - $$ vastab vasakpoole väärtusele;
  - $1 vastab parempoole esimese sümboli väärtusele;
  - ...

Parserite generaator Bison

```
program:
    | program expr '\n'  { printf("%d\n", $2); }
    ;
expr:
    INTEGER              { $$ = $1; }
    | expr '+' expr      { $$ = $1 + $3; }
    | expr '-' expr      { $$ = $1 - $3; }
    ;
```

- Aktsioonid asuvad enamasti parempoole lõpus ning nende täitmine toimub produktsioonireegli taandamisel.
- Võib olla ka sümbolite vahel, misjuhul on see samaväärne lisa mitteterminaliga, mille prempool koosneb antud aktsioonist (ja on muidu tühi).
- Kui aktsioon puudub, siis vaikimisi aktsiooniks on

  `{$$ = $1;}`

Parserite generaator Bison

```
%%
void yyerror(char *s) {
    fprintf(stderr, "%s\n", s);
}
int main(void) {
    yyparse();
    return 0;
}
```

Sisendfaili kolmas osa koosneb C koodist, mis kopeeritakse genereeritavasse faili ilma ühegi muutuseta.

- Olulisemad funktsioonid:
  - `main()` kutsub välja `yyparse();`
  - `yyerror()` raporteerib süntaksi vigadest;
  - `yylex()` tunneb ära mitteterminalid (reeglina defineeritud Flex-i abil).
- Kolmas osa võib puududa, millisel juhul võib ka teise eraldusrea ära jätta.

Parserite generaator Bison

- Funktsioon `yyparse()` kasutab uue lekseemi saamiseks funktsiooni `yylex()`.
- Skanneri ja parseri suhtlusliides on spetsifitseeritud Bison-is:
  - mitteterminalid on deklareeritakse direktiiviga `%token`;
  - ühemärgilised mitteterminalid võivad olla deklareerimata;
  - funktsiooni `yylex()` väljundväärtuseks on kas deklareeritud mitteterminal või üksik märk.
- Kompileerimiseks tuleb skannerisse kaasata päisfail `"*.tab.h"`
  - genereerikas Bison-i poolt andes argumendiks võtme `-d`.
- Alternatiivselt võib parseri spetsifikatsioonifaili kolmads osas kaasata skanneri fail `"lex.yy.c"`.

Parserite generaator Bison

- Mitteterminalide atribuutväärtus antakse edasi muutujas `yylval`.
- Atribuudid on tüüpi `YYSTYPE`, mis vaikimisi on tüüp `int`.
- Erinevatele sümbolitele erinevat tüüpi atribuutide määramiseks tuleb:
  - kõik tüübid spetsifitseerida direktiiviga `%union`

```
%union {
    type1 name1;
    type2 name2;
    ...
}
```

  - määrata terminalide ja mitteterminalide tüübid

```
%token <name> TOKEN
%type <name> non-terminal
```

- Skanneris terminalsümbolile vastava atribuutväärtuse omistamiseks tuleb viidata vastavat tüüpi ühendiväljale (`yylval.name`).

Parserite generaator Bison

- Shift-reduce aktsioonide otsustamiseks kasutatakse ühe sümboli ettevaatamist.
- Konfliktid lahendatakse prioriteedireeglite ja vaikimisi reeglite abil:
  - direktiivid `%left`, `%right` ja `%nonassoc` määravad sümboli assotsiatiivsuse ning prioriteedi;
  - prioriteet on antud kaudselt direktiivide tekstuaalse järjekorraga (mida hilisem seda suurema prioriteediga);
  - reegli prioriteet on sama, mis tema parempoole viimasel mitteterminalil (võib ilmutatult muuta kasutades direktiivi `%prec`);
  - shift/reduce konflikti korral eelistatakse vaikimisi nihutamist;
  - reduce/reduce konflikti korral eelistatakse vaikimisi esimest reeglit.