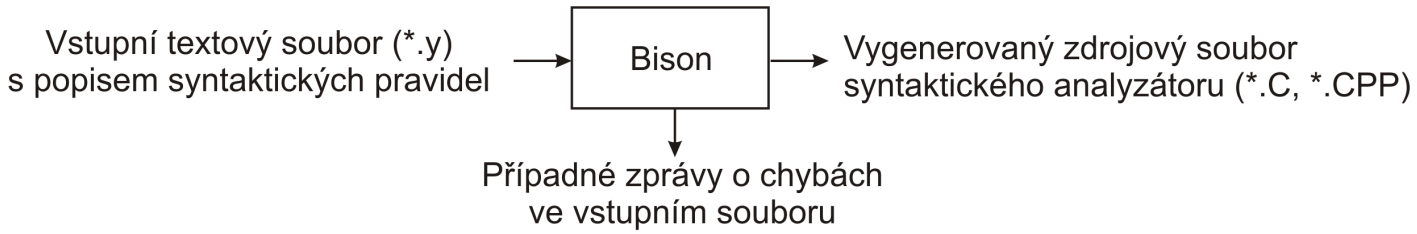
Bison – parser generator



Formát vstupního souboru:

```
%{
  .. hlavičkové soubory, deklarace a jiné ..
%}
deklarace Bisonu
%%
gramatická pravidla
%%
uživatelský kód
```

Část, která je mezi symboly `%{` a `%}`, Bison nijak nevyužívá a jen ji překopíruje do výstupního souboru. Obsahuje jména hlavičkových souborů a různé deklarace, které jsou zapotřebí pro následný překlad vygenerovaného syntaktického analyzátoru.

V části deklarací Bisonu jsou popsány neterminální a terminální symboly bezkontextové gramatiky. Každému symbolu může být přiřazen jeden syntetizovaný atribut. Přitom v deklaracích Bisonu se uvádí proměnná, jejíž datový typ určuje datový typ atributu. Tyto proměnné jsou deklarovány v datovém typu YYSTYPE. Ten je zpravidla vytvořen jako strukturovaný typ *union*.

V následujících zápisech jsou neterminální symboly psány malými písmeny a terminální symboly, které mají délku větší než 1 znak, jsou psány velkými písmeny (konvence používaná programy Flex a Bison).

Deklarace neterminálního symbolu má zápis

`%type <typ> symbol₁ symbol₂ ...`

Příklad.

```
struct Uzel { /* ... */ };
union YYSTYPE { int c; const char *r; Uzel *u; };
%type <u> prikaz blok
```

Pokud některým neterminálním symbolům není přiřazen žádný atribut, nemusí být deklarován nebo deklarace má tvar

`%type symbol₁ symbol₂ ...`

Deklarace terminálních symbolů má zápis

`%token <typ> symbol₁ symbol₂ ...`

Nemá-li terminální symbol přiřazený atribut, deklarujeme ho

`%token symbol₁ symbol₂ ...`

U deklarace operátorů uvádíme jejich asociativitu. Operátory s levou asociativitou se deklarují

%left symbol$_1$ symbol$_2$ ...

Operátory s pravou asociativitou se deklarují

%right symbol$_1$ symbol$_2$ ...

Dále u operátorů uvádíme precedenci. Operátory se stejnou precedencí jsou na jednom řádek. Řádky se zápisy operátorů jsou seřazeny od operátorů s nejnižší precedencí po operátory s nejvyšší precedencí.

Příklad.

```
%right  '='
%left   '+' '-'
%left   '*' '/' '%'
%right  '!' '~'
```

Počáteční symbol gramatiky deklarujeme

%start *symbol*

V části gramatických pravidel jsou uvedena jednotlivá gramatická pravidla bezkontextové gramatiky. Pravidlo bezkontextové gramatiky

A → α

se zapíše ve tvaru

```
A : α ;
```

Za pravidlem zpravidla bývá uveden blok se sémantikou. Zápis je pak ve tvaru

```
A : α { … sémantika … };
```

Více pravidel se stejnou levou stranou

A → α$_1$ | α$_2$ | ... | α$_s$

se zapíší ve tvaru

```
A :   α1 { … sémantika … }
    | α2 { … sémantika … }
    | …
    | αs { … sémantika … };
```

Atributy přiřazené symbolům pravidla mají označení $$, $1, $2, …

Přiřazení označení atributů pro pravidlo **A → X$_1$X$_2$...X$_m$** je

```
A : X1  X2 ..... Xm
$$  $1  $2 ..... $m
```

Příklad.

```
enum Typ { INT,FLOAT,CHYBA };
union YYSTYPE { Typ typ; };
%type <typ> vyraz
vyraz: vyraz '+' vyraz { if ($1==INT && $3==INT) $$=INT;
                         else
                           if ($1==CHYBA || $3==CHYBA) $$=CHYBA;
                           else $$=FLOAT; }
```

```
    | vyraz '&' vyraz { if ($1==INT && $3==INT) $$=INT;
                        else $$=CHYBA; };
```

Hodnoty a funkce dostupné pro uživatele

`void yyerror(const char *);` – tuto funkci volá syntaktický analyzátor při nalezení chyby v analyzovaném programu. Parametrem funkce je popis chyby.