БНФ - метамова опису синтаксису мов програмування

У процесі створення нових мов програмування та в зв'язку з розробкою трансляторів виникає проблема формалізації опису синтаксису мов програмування. Мови, що використовуються для формалізації синтаксису інших мов, називається метамовами. Зараз найбільш уживаною для опису синтаксису мов програмування є метамова форм Бекуса-Наура (скорочено БНФ). Ідея цієї метамови полягає в структуруванні понять вихідної мови програмування і визначення більш складних понять через більш прості.

Формальні мови та граматики.

Алфавіт - це довільна множина символів. Поняття символу не визначається. Ланцюжок символів (слово) - це довільна послідовність символів, що записані рядом. Множина усіх ланцюжків, що складаються з елементів множини X позначають через X*. Мова - це підмножина X*. Приклади мов: Паскаль, C, C++, $\{0^n1^n \mid n >= 0\}$.

Мову можна задати так:
- перелічити всі ланцюжки (слова);
- за допомогою механізму породження слів - граматики;
- написати програму, що одержує на вхід ланцюжок символів і видає відповідь "так", якщо ланцюжок належить мові і "ні" у супротивному випадку.

Щоб задати граматику G=<N,T,P,S>, потрібно вказати:
- множину символів алфавіту (чи термінальних символів) T. Найчастіше позначають їх малими символами алфавіту та цифрами;
- множину N нетермінальних символів (чи метасимволів), що не перетинається з T зі спеціально виділеним початковим символом S (аксіомою). Будемо позначати їх великими буквами;
- множину P правил виводу, що визначають правила підстановки для ланцюжків.

Кожне правило складається з двох ланцюжків (наприклад, p та q), причому ланцюжок p повинен містити принаймні один нетермінал. Правило виводу означає, що ланцюжок p у процесі виводу можна замінити на q. Вивід ланцюжків мови починається з нетермінала S (аксіоми). Правила граматики записують у вигляді

$$p \to q.$$

Більш строго поняття виведеного ланцюжка подамо так:
- аксіома S - виведений ланцюжок;
- якщо $\alpha p \beta$ - виведений ланцюжок і в граматиці є правило $p \to q$, то $\alpha q \beta$ - виведений ланцюжок;
- означена граматикою мова складається з виведених ланцюжків, що містять тільки термінальні символи.

Приклади:

а) $S \to e$
$S \to 0S1$

б) $S \to e$
$S \to (S)$
$S \to SS$

Тут e означає порожній ланцюжок (довжини 0). Для скорочення запису прийнято використовувати символ "|" (читають "або") . Коротка форма запису попередніх прикладів:

а) $S \rightarrow e \mid 0S1$ б) $S \rightarrow e \mid (S) \mid SS$

Граматики є метамовами. Вище була описана "академічна" форма запису метамови. На практиці застосовується також інша форма запису, яку за традицією називають нормальними формами Бекуса-Наура (БНФ).

У метамові БНФ прийняті певні угоди.

Будь-яке поняття мови програмування зображується своїм найменуванням, укладеним у кутові дужки: <...> . Наприклад, речення БНФ, що подає одне визначення деякого поняття Паскаля через інші, має такий вигляд:
поняття Паскаля знак "::=" визначення цього поняття.
У складі визначення можуть використовуватися інші поняття мови програмування, символи алфавіту (термінальні символи) і ключові слова мови програмування, а також спеціальні символи мови БНФ (метасимволи), що мають визначене значення. У якості таких символи використовуються вертикальна риса, квадратні і фігурні дужки. У перші роки використання БНФ множина метасимволів обмежувалась знаком "::=" та вертикальною рисою, квадратні і фігурні дужки (а і потім і круглі) додались у розширених версіях БНФ.

Побудова конструкцій БНФ підкоряється наступним правилам:

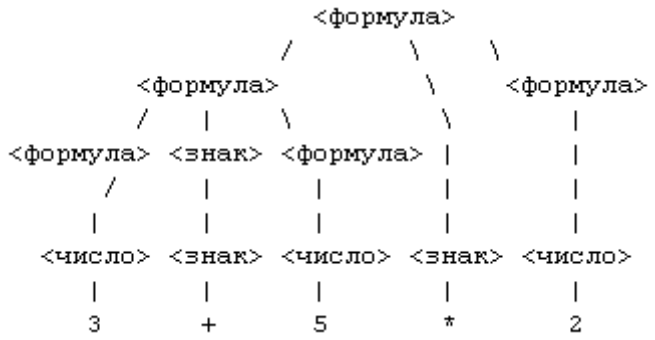
- запис <*поняття1*> *::=* <*поняття2*> <*поняття3*> *і т.д.* означає, що перше поняття в Паскалі являє собою послідовний запис інших понять;
- запис <*поняття1*> *::=* <*поняття2*> | <*поняття3*> | *і т.д.* означає, що перше поняття збігається з одним з інших понять;
- круглі дужки використовуються для угруповання складних конструкцій БНФ усередині простих;
- частина визначення, укладена в квадратні дужки, не обов'язкова;
- частина визначення, укладена у фігурні дужки, може бути повторена довільне число раз (у тому числі жодного разу);
- у якості неозначуваних елементів у правій частині БНФ можуть бути термінальні символи (символи основного алфавіту та ключові слова означуваної мови); для того, щоб відрізняти їх від метасимволів БНФ (наприклад, дужок) у друкованих текстах практикують виділення символів означуваної мови програмування і ключових слів (напр., жирним шрифтом, курсивом, підкресленням або ж, як це прийнято в окремих описах мови С, укладення термінальних символів в одинарні лапки).

Ще раз нагадаємо, що термінали в БНФ записуються як звичайні символи алфавіту, а нетермінали - як імена в кутових дужках < та >.
Наприклад, граматику для множини цілих чисел без знаку можна записати у вигляді:
<число> ::= <цифра> | <цифра> <число>
<цифра> : := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |9

Розглянемо мову найпростіших арифметичних формул:

<формула> ::= (<формула>) |<число> | <формула><знак><формула>
<знак> : := + | *

Наведемо послідовність перетворень ланцюжків (так званий quot;розбір" або "вивід") для ланцюжка 3+5*2. Зобразимо виконувані заміни ланцюжків у вигляді дерева розбору (або дерева виводу). За традицією дерево зображується "догори ногами":



Як і кожне дерево, дерево виводу для виразу з прикладу має ребра і вузли (позначені терміналами та нетерміналами), з яких ростуть гілки. Кінцеві вузли (термінали) називаються листами. Одне й те ж дерево розбору може описувати різні виводи (у дереві не фіксується порядок застосування правил). Якщо для одного й того ж ланцюжка можна побудувати два різних дерева розбору (або, що то ж саме, побудувати, два різних правих виводи), граматика називається неоднозначною. Описана вище граматика є неоднозначною. Ту ж мову можна описати й однозначною граматикою:

<формула> ::= <терм> | <терм><знак> <формула>

<терм> ::= (<формула>) | <число>

<знак> : := + | *

Наведемо кілька прикладів. Непорожній список, що складається з довільної кількості елементів, розділених комою, описується так:
<список> ::= <елемент списку> {**,** <елемент списку>}

Якщо ж список може бути порожнім, то його опис буде виглядати так:

<список> ::= | <елемент списку> {**,** <елемент списку>}

Інший приклад. Ідентифікатором є послідовність букв і цифр, що починається з букви:

<ідентифікатор>::=<буква>{<буква>|<цифра>}

Той факт, що параметри в процедурі можуть бути відсутніми, відбивається за рахунок укладання списку параметрів у квадратні дужки:

<заголовок процедури> ::= ***procedure*** <ім'я процедури> [***(***<параметри>***)***]***;***

Використання круглих дужок ілюструється на прикладі опису процедури або функції, що є або описом процедури, або описом функції, після чого іде крапка з комою:

< процедура чи функція>::=(<процедура>|<функція>)***;***

Далі наведемо кілька фрагментів опису синтаксису мови Паскаль . Так подається загальна структура програми на Паскалі в термінах мови БНФ:

<опис програми> ::=

<заголовок програми>

<блок оголошень>

<блок процедур і функцій>

<операторна частина> **.**

<заголовок програми> ::=

program <ім'я програми> [***(***<список параметрів>***)***]***;***

<блок оголошень> ::={<розділ оголошень>***;*** }

<розділ оголошень> ::= <розділ констант> | <розділ типів> |
<розділ змінних> | <розділ міток> | <розділ модулів>

<розділ констант> ::=
const <опис константи> {***;*** <опис константи>}

<опис константи> ::= <ім'я константи> [***:*** <тип>] **=** <вираз>

<розділ типів> ::= ***type*** <опис типу> {***;*** <опис типу>}

<опис типу> ::= <ім'я типу> **=** <тип>

<розділ змінних> ::=

var <оголошення змінних> {***;*** <оголошення змінних>}

<оголошення змінних> ::=
<ім'я змінної> {**,** <ім'я змінної> }***:*** <тип>

<розділ міток> ::= ***label*** <мітка> {**,** <мітка>}

<мітка> ::= <ціле без знака> | <ідентифікатор>

<розділ модулів> ::= ***uses*** <ім'я модуля> {**,** <ім'я модуля>}

<блок процедур і функцій> ::= {<опис процедури чи функції> }

<опис процедури чи функції>::= (<опис процедури> | <опис функції>) ***;***

<опис процедури> ::=

<заголовок процедури>

<блок оголошень>

<заголовок процедури> ::=

procedure <ім'я процедури> [***(***<список формальних параметрів>***)***]***;***

<список формальних параметрів> ::=
<формальний параметр> { ***;*** <формальний параметр> }

<формальний параметр> ::=
[***var***] <ім'я параметра> {**,** <ім'я параметра> }***:*** <ім'я типу>

<опис функції> ::=

<заголовок функції>

<блок оголошень>

<операторна частина>

<заголовок функції> ::= ***function*** <ім'я функції>
[***(***<список формальних параметрів>***)***] ***:*** <ім'я типу>***;***

<операторна частина> ::=
begin (| <оператор>{***;*** <оператор> }) ***end***

Зазначимо, що імена констант, типів, змінних, модулів, функцій і процедур є ідентифікаторами

Як інший приклад фрагмента синтаксису Паскаля наведемо опис специфікації типу:

<тип> ::= <ім'я стандартного типу> | <ім'я користувацького типу> |
<перелічимий тип> | <діапазон> | <тип масиву> | <тип запису> |
<тип множини> | <тип файлу>

<перелічимий тип> ::= ***(*** <ідентифікатор> {**,** <ідентифікатор> } ***)***

<діапазон> ::= <значення> **..** <значення>

<тип масиву> ::= ***array*** [<ім'я типу> | <діапазон>] ***of*** <тип>

<тип запису> ::= ***record*** <поле запису> {***;*** <поле запису> } ***end***

<поле запису> ::= <ім'я поля> ***:*** <тип>

<тип множини> ::= ***set of*** (<ім'я типу> | <діапазон>)

<тип файлу> ::= ***file of*** <тип>