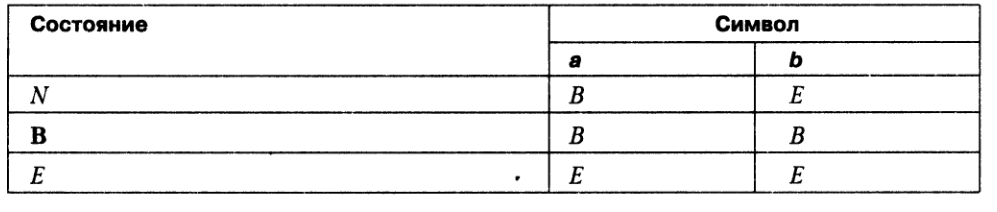
**Таблиця переходів детермінованого скінченого автомата**

Поруч із поданням графом функція переходів ДСА може бути задана таблицею, що, безумовно, більше підходить програмної реалізації скінченого автомата (табл. 1).

Розглянемо таблицю переходів ДСА А12, що розпізнає мову ідентифікаторів.

Таблиця 1. Таблиця переходів кінцевого автомата А12;



У таблиці записано стан, в який переходить автомат, перебуваючи у стані, відповідному даному рядку таблиці, і отримавши вхідний символ, позначений у відповідному стовпці. Кінцевий стан автомата позначено в таблиці жирним шрифтом.

Поряд із станами N та В передбачено додатковий стан Е - стан помилки. Це зроблено для того, щоб функцію переходів було визначено для всіх можливих пар символ-стан. Інакше перехід із стану N під час вступу на вхід символу b було б не визначено.

Потрапивши у стан Е, автомат залишається у ньому. Стан Е перестав бути кінцевим. На практиці при програмній реалізації, крім символів вхідного алфавіту, потрібно, швидше за все, визначити реакції автомата і будь-які інші символи, які, очевидно, повинні переводити автомат у стан Е.

**Програмна реалізація автоматного розпізнавача**

У лістингу 1 наведено ескіз програми, що моделює роботу скінченого детермінованого автомата. Ця програма є універсальним розпізнавачем (синтаксичним аналізатором) автоматних мов.

У ній є лише деякі умовності: передбачається, що стани позначаються великими латинськими літерами (S - початковий стан), а вхідний алфавіт - малі латинські літери. Не конкретизовано також способи зчитування символів та перевірки їх наявності на вході, а також те, як автомат реагує на прийняття чи неприйняття вхідного ланцюжка – ці частини програми записані російською мовою.

Лістинг 1. Універсальний розпізнавач автоматних мов

**type**

tCondition = (S, А, В, С, ..., Е, ...) ; { Стани }

tAlpha = 'а'..'z'; { Алфавіт }

tJump = array [tCondition, tAlpha] of tCondition;

{ Таблиця переходів }

tFinish = set of tCondition;

/ Тип множини кінцевих станів }

**var**

Cond : tCondition; { Поточний стан }

Ch : tAlpha; { Вхідний символ }

Р : tJump; { Функція переходів }

Fin : tFinish; { Кінцеві стани }

{ Тут задаються значення Р и Fin }

Cond := S;

**while** Є символи **do begin**

Читати(Ch);

Cond := P[Cond, Ch]

**end;**

**if** Cond **in** Fin **then**

Ланцюжок прийнятий

**else**

Ланцюжок не прийнятий

**Дерево розбору в автоматній граматиці**

Говорячи про завдання синтаксичного аналізу, ми зводили його до побудови дерева розбору. Тим часом у попередніх розділах на роль розпізнавача автоматних мов запропоновано кінцевий автомат, який дерево не будує. Чи немає тут протиріччя?

Ні. Дерево розбору ланцюжка в автоматній граматиці може бути побудовано, якщо відома послідовність переходів звичайного автоматного розпізнавача. Тобто автомат, що розпізнає, не тільки дає відповідь на питання про належність ланцюжка мови, але і дозволяє виявити структуру ланцюжка.

Структура при цьому представлена послідовністю переходів автомата. Розглянемо, який вид має дерево аналізу термінального ланцюжка в автоматній граматиці. З трьох видів правил автоматної граматики правила виду

**A →a**

та

**А → ε**

можуть бути використані в процесі породження ланцюжка рівно один раз, після чого закінчується процес породження. Усі інші підстановки виконуються за правилами виду

**A→аВ**

Кожна така підстановка призводить до появи у дереві нової внутрішньої вершини, позначеної нетерміналом. Її ліва дочірня вершина позначається терміналом (рис. 12).

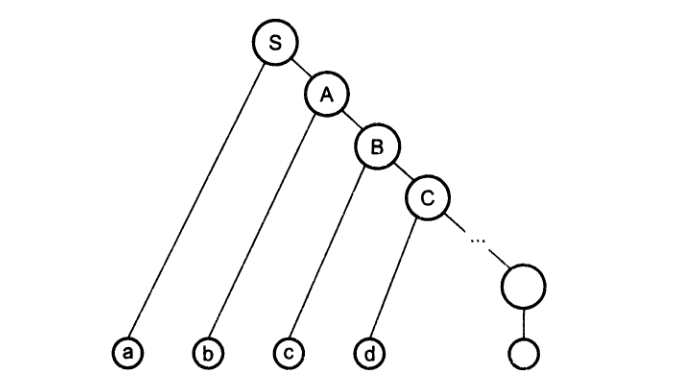


Рис. 12. Дерево розбору в автоматній граматиці

Якщо для граматики типу 3 побудований ДСА, то вхідний ланцюжок однозначно визначає послідовність станів, що проходять автоматом, і дерево виведення.

**Приклад автоматної мови**

Розглянемо мову цілих чисел зі знаком. Приклади правильно записаних чисел:

177 +22 -1 0 02

Побудуємо скінчений автомат, який розпізнає цю мову. Задаємо цей автомат за допомогою діаграми переходів. Ця діаграма буде також формальним визначенням самої мови.

Початковий стан автомата позначимо S (рис. 13). Перебуваючи у цьому стані, автомат очікує символ, з якого може розпочинатися запис числа.

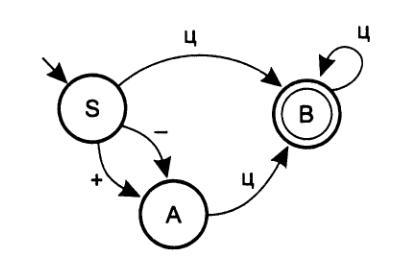


Рис.13. Скінчений автомат, що розпізнає цілі числа зі знаком

Це знаки "+", "-" та цифри. Відповідно, зі стану S повинні виходити дуги, позначені цими символами. Десять дуг, помічених цифрами від 0 до 9, замінимо на одну, позначивши її символом «ц». Після того, як прийнято знак числа, автомат повинен перейти в стан, в якому він очікує першу цифру.

Позначимо такий стан А. Отже, переходи за символами «+» і «-» ведуть зі стану S стан А. Якщо, перебуваючи в початковому стані S, автомат отримав цифру, він повинен перейти в стан (позначимо його В), в якому можуть бути прийняті наступні цифри, якщо вони є.

Зі стану А при отриманні цифри автомат також переходить у стан В. Стан В слід помітити як кінцеве, оскільки перехід у цей стан означає, що на вхід автомата надійшов правильний запис цілого числа.

Дуга, позначена символом «ц», яка веде зі стану В до нього самого, дозволяє автомату прийняти другу і наступні цифри числа, якщо вони є.

За діаграмою переходів можна записати і граматику, що породжує мову цілих чисел зі знаком. Кожній дузі відповідає правило. Кінцеві стани породжують правила з порожнім ланцюжком у правій частині.

S→+A|-A| *ц* B

А→ *ц* В

В→ *ц* В|ε

Насправді, як говорилося, доводиться враховувати можливість надходження на вхід автомата як символів вхідного алфавіту, а й будь-яких інших символів.

У цій ситуації можна припускати, що з будь-якого стану виходить дуга, що веде до стану помилки Е (рис. 14). Крім того, зручно вважати, що вхідний ланцюжок завжди завершується спеціальним символом «кінець тексту», який позначають ⊥

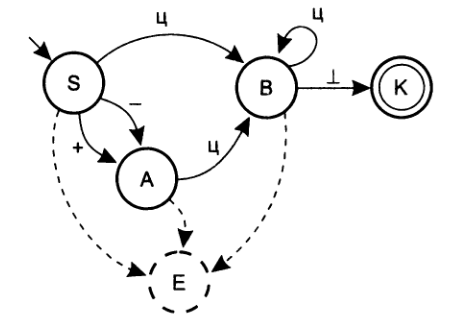


Рис. 14. Скінчений автомат зі станом помилки та додатковим кінцевим станом

При використанні символу ⊥ до автомата слід додати стан К, який буде єдиним кінцевим, і в який із «колишніх» кінцевих станів будуть направлені дуги, позначені ⊥.