Не важко було б заповнити для отриманого автомата таблицю переходів і використовувати універсальний розпізнавач (див. листинг 1), моделюючий поведінку ДСА. Але тут ми розглянемо іншу нагоду.

Користуючись діаграмою переходів (див. рис. 13 і 14), напишемо програму, яка веде себе подібно до скінченого автомата, але не моделює його поведінку безпосередньо. На початку роботи, тобто перебуваючи у вихідному стані, програма зчитує перший символ вхідного ланцюжка та перевіряє його

Читати(Символ);

if not (Символ in ['+', '-', '0'.. '9']) then

Помилка

Вважатимемо, що звернення до процедури Помилка зупиняє роботу розпізнавача.

Далі (як і раніше залишаючись у початковому стані) перевіряємо, який саме з допустимих символів надійшов на вхід.

Якщо це знак, читаємо наступний символ, переходячи до стану А та передбачаючи дії, які автомат виконує, перебуваючи у цьому стані:

{ Стан А }

**if** Символ **in** ['0'..'9'] **then**

Читати(Символ)

**else**

Помилка

**end**

**else** { Символ - цифра (в состоянии S) }

Читати(Символ);

{ Стан В }

Якщо в стані S надійшов символ цифри, зчитується наступний символ і відбувається перехід до стану В.

Після виконання наведеного фрагмента програма або зупиняється через помилку, або приходить у стан, аналогічний стану кінцевого автомата. Перебуваючи в стані В, автомат повинен приймати всі цифри, що надійшли на вхід, залишаючись при цьому в стані В:

{ Стан В }

**while** Символ **in** [ '0'.. '9'] **do**

Читати(Символ);

Вихід із циклу відбувається, якщо черговий лічений символ – не цифра.

Якщо це символ ⊥ — «кінець тексту», то вхідний ланцюжок закінчено і автомат перетворюється на стан К (див. рис. 14), приймаючи вхідний ланцюжок.

Якщо цикл припинено через надходження символу, відмінного від цифри та ⊥, автомат переходить у стан помилки:

**if** Символ = ‘⊥’ **then**

Ланцюжок прийнятий

**else**

Помилка;

Як бачимо, розпізнавач автоматної мови, якою є наша програма, можна написати, не вдаючись до прямого моделювання поведінки кінцевого автомата з використанням таблиці переходів.

Такий підхід може мати переваги. Технологія програмування розпізнавача виявляється досить простою: програма пишеться за діаграмою переходів ДСА, яка виконує роль схеми алгоритму.

Синтаксичні діаграми автоматної мови

У побудованій програмі-розпізнавачі стану скінченого автомата не фігурували явно. Вони просто відповідали деяким точкам програми. На противагу станам, символи, якими позначені дуги діаграми переходів, явно використовуються у розпізнавачі. Усунемо з діаграми переходів позначення станів (великі літери та гуртки). Ті місця діаграми, де були стани автомата, перетворюються на точки розгалуження та з'єднання дуг.

Термінальні символи, що відзначають переходи - дуги, навпаки, розмістимо в гуртках на цих дугах. Результат такого перетворення для діаграми переходів автомата, що розпізнає цілі числа, або, що те саме, для графа автоматної граматики, що породжує цілі числа, показано на рис. 15 а. Отриманий граф має назву синтаксичної діаграми автоматної мови або автоматної граматики.

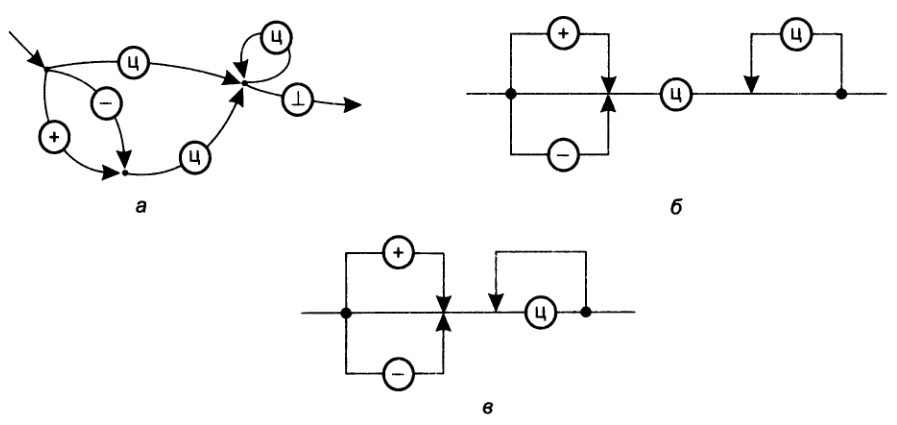


Рис.15 Синтаксичні діаграми граматики цілих чисел

Синтаксична діаграма може використовуватися для тих же цілей, що граматика та граф переходів автомата, тобто для породження та розпізнавання речень мови.

Будь-який шлях від входу діаграми (відповідає початковому стану автомата) до її виходу (кінцевому стану К) породжує ланцюжок символів, що є правильною пропозицією мови (сентенцією граматики). Розв'язання завдання розпізнавання зводиться до пошуку такого шляху від входу до виходу діаграми, який відповідає заданому ланцюжку.

Синтаксична діаграма може використовуватися для тих же цілей, що граматика та граф переходів автомата, тобто для породження та розпізнавання речень мови.

Порівняно з граматикою, що породжує, і скінченим автоматом синтаксичні діаграми набагато наочніші і краще підходять для специфікації мови при його конструюванні. Дуже зручна діаграма у ролі схеми алгоритму під час написання синтаксичного аналізатора.

Використовуючи іншу манеру накреслення дуг, перемістивши символ «ц» на дугу, що веде в колишній стан і відмовившись від явного зображення кінця тексту, отримаємо простішу діаграму, що задає синтаксис цілих зі знаком (рис. 15, б).

Спрощення діаграми можна продовжити і надалі (рис. 16, в), зберігаючи її еквівалентність вихідної (рис.15, а). Однак, при програмуванні аналізатора на Паскалі ця остання діаграма не зручніша за попередню.

Перепишемо аналізатор цілих чисел, користуючись синтаксичною діаграмою, показаною на рис. 15 б, як схемою алгоритму (листинг 2).

Щоб позбавитися умовностей, будемо вважати, що програма має доступ до глобальної змінної **ch**, яка зберігає поточний символ. Читання наступного вхідного символу виконує процедуру **Nextch**, яка поміщає прочитане значення у змінну **ch**.

Вважається, що константа EOT (End Of Text) позначає кінець тексту. Реакція на помилку покладена на процедуру Error, яка видає повідомлення про помилку та зупиняє роботу програми-розпізнавача. У разі прийняття вхідного ланцюжка жодного спеціального повідомлення не передбачається.

*Лістинг 2. Розпізнавач цілих зі знаком*

NextCh; { Прочитати перший символ }

**if** Ch **in** ['+', '-' ] **then**

NextCh;

**if** Ch **in** ['0'..'9'] **then**

NextCh

**else**

Error;

**while** Ch **in** [' 0 ' . . ' 9 ' ] **do**

NextCh;

i**f** Ch <> EOT **then**

Error;

Слід зазначити ряд важливих рис програми, що вийшла.

Вона складається з кількох частин, розділених у лістингу порожніми рядками.

Перша і остання частини, як неважко зрозуміти, повинні бути завжди: перед початком аналізу треба отримати перший символ, а після завершення — переконатися, що в момент, що відповідає виходу з діаграми, вхідний ланцюжок вичерпано.

Три інші частини суворо відповідають структурі синтаксичної діаграми (див. рис. 15, б). На діаграмі виділяються три послідовно з'єднаних ділянки, і програма містить три послідовно записаних і виконуваних фрагментів.

Перший (if-then) перевіряє наявність (необов'язкового) знаку. Другий (if-then-else) – наявність обов'язкової цифри. Цикл while (який, як відомо, може не виконатись жодного разу) відповідає циклу на діаграмі, що задає послідовність з нуля або більше цифр.

Програма-розпізнавач може бути написана за синтаксичною діаграмою автоматної граматики з використанням формальних прийомів.