УДК 681.3.06

В.Г. Левицький, аспір.
Житомирський інженерно-технологічний інститут

СПОСІБ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ДЕТЕРМІНОВАНИХ СКІНЧЕНИХ АВТОМАТІВ ЗА ТЕКСТОВИМ ОПИСОМ БЕЗ ПОПЕРЕДНЬОЇ ОБРОБКИ РЕГУЛЯРНИХ ВИРАЗІВ

(Науковий керівник – к.т.н., доц. М.М. Колодницький)

Запропоновано шлях вирішення проблеми швидкої побудови детермінованих скінчених автоматів за текстовим описом у вигляді регулярних виразів. Застосований спосіб орієнтовано на використання в синтаксично орієнтованому однопрохідному компіляторі текстових описів мов програмування і проблемно-орієнтованих мов та не потребує попередньої обробки текстового опису автомата. Розглянуто приклад застосування даного методу для генерації простого детермінованого скінченого автомата.

Вступ

Пошук певних текстових шаблонів є досить поширеною задачею в багатьох областях комп'ютерних технологій (лексичний аналіз, редагування текстів, обробка запросів тощо). Одним зі шляхів вирішення цієї проблеми є специфікація текстового шаблона за допомогою регулярного виразу і подальша побудова скінченого автомата (детермінованого або недетермінованого), який власне і має обробляти вхідний рядок та шукати в ньому текст за зразком. Такий підхід здобув популярність завдяки зручному способу запису шаблона і дуже високій ефективності процедури обробки вхідного тексту та широко застосовувався в багатьох інструментальних засобах генерації лексичних аналізаторів (наприклад, AT&T Lex [1]).

Задача генерації скінчених автоматів з регулярних виразів є добре відомою. Одним із популярних методів її вирішення є класичний алгоритм, описаний McNaughton та Yamada [2], який обробляє позначені регулярні вирази. В літературі зустрічаються також інші підходи до розв'язання цієї проблеми [3, 4, 5], які є іноді адаптацією відомого алгоритму [2] до таких спеціальних випадків, як, наприклад, "ледача" (lazy and incremental) генерація лексичних аналізаторів [6].

Однією з подібних спеціальних задач можна вважати проблему побудови скінченого автомата з регулярних виразів "на літу" – формування автомата одночасно з обробкою вхідного текстового опису. Дана задача виникла під час розробки програмного середовища генерації мовних процесорів *FancyCC* [7] та потребувала такого способу аналізу регулярних виразів, який був би зручним для використання в однопрохідному компіляторі вхідних описів мов програмування та проблемно-орієнтованих мов, що використовує для синтаксичного розбору метод SLR(1).

Генерація детермінованих мінімізованих скінчених автоматів

Представлений нижче метод генерації скінчених автоматів, розроблений для вирішення поставленої проблеми, не потребує попередньої обробки регулярних виразів (на відміну від класичного алгоритму [2]) та формує детерміновані мінімізовані скінчені автомати під час дій по згортці синтаксичного аналізатора.

Викладемо спочатку деякі відомі визначення, що необхідні для розгляду процедури генерації скінчених автоматів в програмному середовищі генерації мовних процесорів.

Першою фазою процесу компіляції є лексичний аналіз. Незалежно від того, чи складає він окремий прохід компілятора, чи працює під керівництвом синтаксичного аналізатора, зручно уявляти лексичний аналіз як окрему фазу компіляції. На вхід лексичного аналізатора (сканера) надходить ланцюжок символів деякого алфавіту, що складають текст програми; сканер пропускає порожні рядки та коментарі, а символи програми групує в лексеми. Слід відмітити, що синтаксис лексем є в більшості випадків порівняно простим, що дозволяє будувати для сканера більш ефективні методи розбору. Звичайно конструкції, що будуть виділятися як лексеми, вибирають таким чином, щоб синтаксис лексем визначався в рамках порівняно простого і добре дослідженого класу мов – регулярних множин. В цьому випадку лексичний аналіз можна проводити за допомогою скінчених автоматів, що дозволяє будувати

дуже ефективні процедури розпізнавання лексем. Тому, під час розробки програмного середовища генерації мовних процесорів, виправданим є підхід, що пропонує проводити визначення лексичного аналізатора за допомогою множини правил – регулярних виразів.

Множина в алфавіті Σ регулярна тоді і лише тоді, коли вона або ∅, або {ε}, або {a} для деякого a ∈ Σ, або її можна отримати з цих множин, застосувавши кінцеве число операцій об'єднання, конкатенації та ітерації (символ ε використовуємо для позначення порожнього ланцюжка).

Регулярні вирази в алфавіті Σ тоді визначають рекурсивно таким чином:

1) ∅ — регулярний вираз, що позначає регулярну множину ∅;
2) ε — регулярний вираз, що позначає регулярну множину ε;
3) якщо a ∈ Σ, то a – регулярний вираз, що позначає регулярну множину {a};
4) якщо p та q – регулярні вирази, що позначають регулярні множини P та Q відповідно, то
   - ✓ (p | q) – регулярний вираз, що позначає P ∪ Q;
   - ✓ (pq) – регулярний вираз, що позначає PQ;
   - ✓ p* — регулярний вираз, що позначає P*;
5) ніщо інше не є регулярним виразом в алфавіті Σ.

Інший поширений метод, що забезпечує завдання мови лексичного аналізатора кінцевими засобами, полягає у використанні найпростішого розпізнавача – скінченого автомата. Ми визначимо скінчений автомат, задавши кінцеву множину його керуючих станів (Q), допустимі вхідні символи (множина Σ), початковий стан та множину кінцевих станів, тобто станів, що вказують на допустимість вхідного ланцюжка символів. Задається також функція переходів, яка, використовуючи дані про поточний стан автомата та поточний вхідний символ, визначає наступний керуючий стан (δ: Q × Σ → Q). Ми далі будемо користуватися більш зручними засобами представлення функції переходів скінченого автомата – таблицею переходів та графом переходів.

Інформація в таблиці переходів вміщується згідно з такими правилами:

- ✓ стовпці позначені символами вхідного алфавіту;
- ✓ рядки позначені керуючими станами автомата;
- ✓ елементами таблиці є наступні керуючі стани, що відповідають вхідним символам стовпців та поточним станам рядків;
- ✓ перший рядок відповідає початковому стану;
- ✓ рядки, що відповідають заключним станам, позначені одиницями, всі інші рядки позначені нулями.

Графом переходів автомата M називають невпорядкований позначений граф, вершини якого позначені іменами станів і в якому є дуга (p, q), якщо існує такий вхідний символ a ∈ Σ, що δ(p, a) = q (така дуга позначається символом a). Заключні стани на графі переходів обводять подвійним кружком.

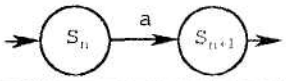
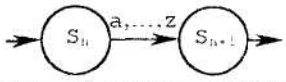
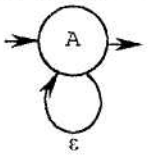
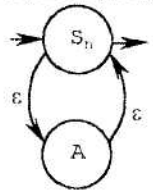
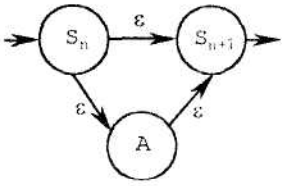
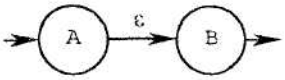
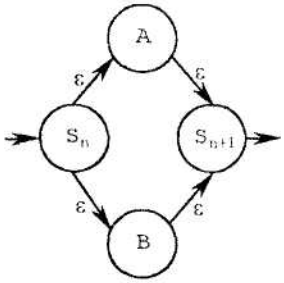
Процедура побудови результуючого автомата складається з трьох етапів: формування педетермінованого скінченого автомата, перетворення його на детермінований автомат та мінімізація отриманого автомата. Перший етап проходить до тих пір, поки регулярний вираз не буде проаналізовано повністю, та поділяється на ряд процедур, кожна з яких викликається в момент виконання згортки за одним з правил підстановки граматики мови регулярних виразів (рис. 1). В найбільш загальному вигляді спосіб формування педетермінованого скінченого автомата представлено схематично в табл. 1, що визначає фрагменти графа переходів автомата, які відповідають складовим частинам регулярних виразів.

При цьому використовуються такі правила:

- ✓ маленькі латинські літери використовуємо для позначення символів алфавіту;
- ✓ прописні латинські літери використовуємо для позначення фрагментів регулярних виразів;
- ✓ вершини графа, які позначені літерами A та B, відповідають не окремому стану скінченого автомата, а певному фрагменту графа переходів, що складається щонайменше з двох станів, і має один вхідний та один вихідний стан;
- ✓ вершини графа, позначені як S_n та S_{n+1}, відповідають новим станам скінченого автомата;
- ✓ новий стан з максимальним номером та відповідні ε-досяжні стани є заключними.

Таблиця 1

Правила формування графа переходів недетермінованого скінченого автомата

Правило №	Регулярний вираз	Фрагмент графа переходів
1	a (атом)	
2	[a-z] (множина атомів)	
3	A+ (1 або більше разів)	
4	A* (0 або більше разів)	
5	[A] (0 або 1 раз)	
6	AB (A та B)	
7	A|B (A або B)	

Таким чином, "атомарні" частини недетермінованого автомата, що генерується, отримуємо за першими двома рядками таблиці, а весь граф переходів будується як композиція "атомарних" частин за рештою правил.

Подальша обробка недетермінованого автомата проводиться за добре відомими алгоритмами [3, 4]. Детермінізація скінченого автомата відбувається шляхом додавання нових станів автомата, а остання фаза генерації складається з об'єднання еквівалентних станів та видалення недосяжних станів. При цьому стани s та t вважаються еквівалентними тоді і тільки тоді, коли виконуються такі умови:

1) умова подібності — стани s та t повинні бути обидва або заключними, або ні;
2) умова спадкоємності — для всіх вхідних символів стани s та t повинні переходити в еквівалентні стани.

Таким чином, застосовуючи розглянутий метод, отримаємо в результаті детермінований мінімізований скінчений автомат, у якого функція переходів, алфавіт, множина керуючих станів та кінцеві стани були сформовані одночасно з обробкою вхідного текстового опису регулярних виразів.

```
DFA      → LEFT   EXPR
LEFT     → identifier   =
EXPR     → EXPR   or   AND
EXPR     →             AND
AND      → AND   TERM
AND      →       TERM
TERM     → OPERAND   REPEAT
TERM     → OPERAND
REPEAT   → *
REPEAT   → +
OPERAND  → ~   [   ATOMS   ]
OPERAND  →     [   ATOMS   ]
OPERAND  →     (   EXPR    )
OPERAND  →     {   EXPR    }
OPERAND  → identifier
OPERAND  → atom
OPERAND  → string
ATOMS    → ATOMS   atom
ATOMS    →         atom
ATOMS    → ATOMS   atom   -   atom   ATOMS
ATOMS    →         atom   -   atom   ATOMS
ATOMS    → ATOMS   atom   -   atom
```

Рис. 1. Правила підстановки для граматики мови регулярних виразів

Приклад

Розглянемо, яким чином працює запропонований метод генерації детермінованих скінчених автоматів, на прикладі обробки регулярного виразу `letter(letter|digit)*`, що визначає традиційний синтаксис ідентифікатора для мови програмування.

Застосувавши правила 1 (a) для кожного входження `letter` та `digit` та правила 7 (A|B) і 4 (A*) з табл. 1, отримаємо недетермінований скінчений автомат M_1 (рис. 2 та табл. 2).

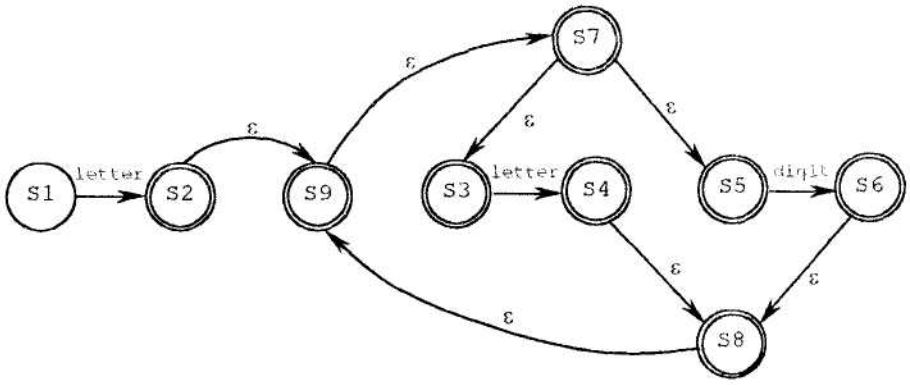


Рис. 2. Граф переходів недетермінованого скінченого автомата для розбору ідентифікаторів

Таблиця 2

Таблиця переходів автомата M_1

		letter	digit
0	1	{2,3,5,7,9}	
1	2	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	3	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	4	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	5	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	6	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	7	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	8	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	9	{3,4,5,7,8,9}	{3,5,6,7,8,9}

Застосувавши до автомата M_1 процедуру детермінізації, отримаємо детермінований скінчений автомат M_2 (таб. 3).

Таблиця 3

Таблиця переходів автомата M_2

		letter	digit
0	1	{2,3,5,7,9}	
1	2	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	3	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	4	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	5	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	6	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	7	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	8	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	9	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	{2,3,5,7,9}	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	{3,4,5,7,8,9}	{3,4,5,7,8,9}	{3,5,6,7,8,9}
1	{3,5,6,7,8,9}	{3,4,5,7,8,9}	{3,5,6,7,8,9}

Після об'єднання еквівалентних станів отримаємо результуючий детермінований мінімальний скінчений автомат M_3 (рис. 3 та табл. 4).

Таблиця 4

Таблиця переходів автомата M_3

		letter	digit
0	1	2	
1	2	2	2

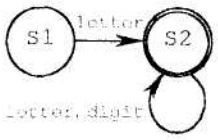


Рис. 3. Граф переходів мінімізованого детермінованого скінченого автомата для розбору ідентифікаторів

ЛІТЕРАТУРА:

1. *Lesk M.E.* Lex – lexical analyzer generator. Comp. Sci. tech. rep. № 39. Bell Laboratries: Murray Hill, New Jersey, 1975.
2. *McNaughton R., Yamada H.* Regular expressions and state graphs for automata. IRE Trans. Electron. Comput. EC-9 (1960). – Pp. 38–47.
3. *Aho A.V., Sethi R., Ullman J.D.* Compilers: Principles, Techniques and Tools. Addison-Wesley, Reading, Mass., 1986.
4. *Dick Grune, Ceriel Jacobs.* Parsing Techniques: A Practical Guide, Ellis Horwood Limited, Chichester, 1990. – 323 p.
5. *Parr T.J.* Language Translation Using PCCTS and C++. A Reference Guide. Automata Publishing Company, 1993. – 310 p.
6. *Heering J., Klint P., Rekers J.* Incremental generation of lexical scanners // ACM Trans. Program. Lang. Syst. 14, 4 (Oct. 1992). – Pp. 490–520.
7. *Колодницький М.М., Левицький В.Г.* Сучасні інструментальні засоби автоматизованої побудови мовних процесорів // Праці 2-ї міжн. наук.-практ. конф. з програмування "УкрПрог'00", 23–26 травня 2000 р. – Київ, 2000. (в друці)

ЛЕВИЦЬКИЙ В'ячеслав Георгійович – аспірант Житомирського інженерно-технологічного інституту.

Наукові інтереси:
– комп'ютерні інформаційні технології;
– моделювання і розв'язок задач за допомогою обчислювальної техніки;
– побудова компіляторів.

Подано 14.12.1999.