Използване на минимални автомати за кодиране на граматически речници (по [1] и [2])

I. Увод

1. Крайни автомати

Крайните автомати (КА) са ефективни устройства за разпознаване на голям и важен клас от формални езици. Сложността за разпознаване при детерминиран КА е ***O(n),*** където ***n*** е дължината на думата, подадена за разпознаване. За повече информация относно езиците, разпознавани от КА, вижте [3]. Във всеки случай това са езици с достатъчно изразни средства.

Теорема на Клини: Класът формални езици, разпознавани чрез КА, е еквивалентен на езиците, които се описват чрез регулярни изрази.

За всеки регулярен език съществува единствен минимален детерминиран КА (ДКА), който го разпознава (с точност до преименуване на състоянията).

2. КА за кодиране на граматически речници

Видяхме начин за използване на КА при кодиране на морфологичен речник (файл **fst_dict.pdf,** стр. 32-34). Показан е достатъчно голям пример. Накратко, разделяме думите от речника на краен брой класове, като всеки клас кодира определена граматическа информация:

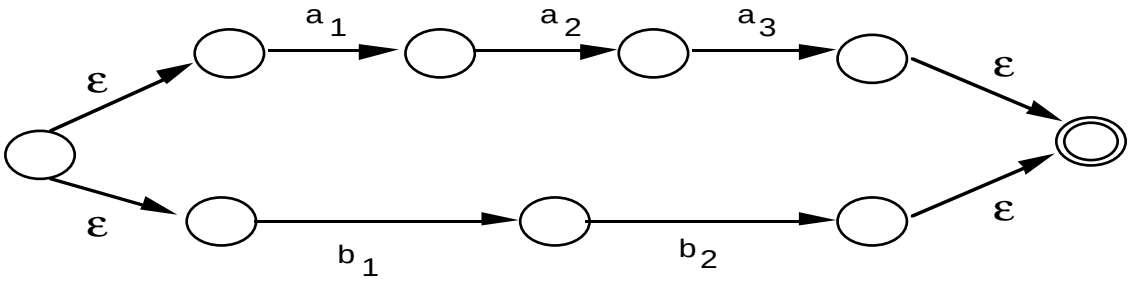
(а) как думата се получава от основната форма и

(б) какви са граматическите характеристики на думата.

Класовете се кодират във финалните състояния на автомата-речник. С такъв автомат успешно се решава следната задача: *по дадена дума да се разпознае дали е от речника и ако да, в кой граматически клас влиза тя*. Към многозначните думи (*води* на примера в **fst_dict.pdf**) се асоциират няколко вида граматическа информация, според броя на класовете, към които принадлежат. “Разпознаване на думата” означава да се достигне крайно (финално) състояние по дъгите на автомата, при което граматическата информация се извежда от крайното състояние, към което е “закрепена”. Показаният на примера автомат се нарича “автомат с етикети на крайните състояния”.

II. Построяване на КА, кодиращ зададен граматически речник

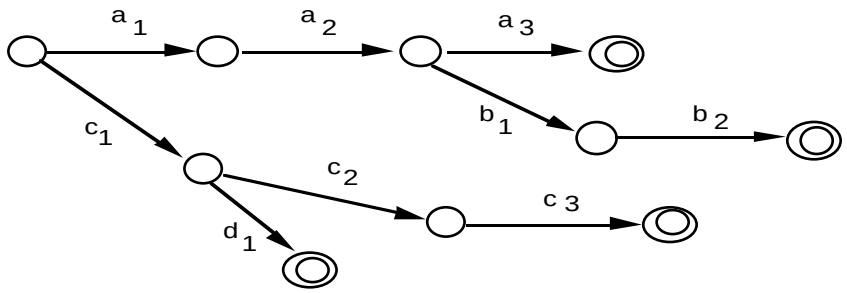
Възниква въпросът как ефективно да се строят такива автомати? Теоретичният отговор е, че винаги може да се построи някакъв КА и после да се минимизира (по известни алгоритми). Например: нека е даден речник от две думи { $a_1a_2a_3$, b_1b_2 }. Един КА, който го разпознава, е показан на Фиг. 1.



Фиг. 1. Недетерминиран КА за разпознаване на { $a_1a_2a_3$, b_1b_2 }.

Тривиалният КА на фиг. 1 е недетерминиран (поради епсилон-преходите). Построяване на еквивалентния ДКА е скъпа операция, а броят на състоянията нараства от **n** за не-ДКА до $\mathbf{2^n}$ за еквивалентния ДКА.

Друга идея е да се използват "дървовидни автомати" (те са детерминирани и ациклични). Строят се във вид на дърво *по зададени в лексикографска подредба* думи на речника. Пример за езика: { $a_1a_2a_3$, $a_1a_2b_1b_2$, $c_1c_2c_3$, c_1d_1 }



Фиг. 2. Дървовиден КА, построен по списък в лексикографска наредба

Тъй като лексикографската сортировка е в нарастващ ред, при директно построяване на КА като дърво се гарантира, че "няма връщане назад (нагоре) с цел добавяне на нови преходи". Този автомат е детерминиран и осигурява разпознаване за **n** прехода, където **n** е дължината на думата. Но такива КА са огромни; размерите им са съизмерими с броя на буквите в началния списък (макар началните букви да съвпадат както е показано на фиг. 2). При милион думи със средна дължина 6-7 символа се получават милиони състояния в КА и това е твърде голяма конструкция. Самата минимизация е с квадратична сложност [3]. Има друг, по-ефективен начин.

II.1. Директно построяване на минимален ацикличен КА

Интуитивно, думите често имат съвпадащи окончания. Значи има смисъл да се търси не само общо начало на съседни думи в лексикографската

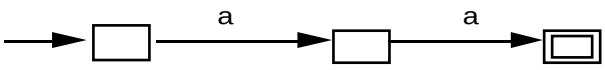
наредба (както на фиг. 2), но и общ край - очевидно при несъседни думи (сортировката засяга началото). Окончанията са фиксирани и заради това минимизацията е толкова ефективна, че полученият минимален ацикличен ДКА е около 100 пъти по-малък.

От дървовидния КА на фиг. 2 се вижда, че във всеки момент (в определена точка от списъка-речник) част от състоянията на текущия КА са "стабилни" (това са тези "над" текущата дума), докато друга част е в процес на внасяне на изменения. Това наблюдение се използва и при изложената по-долу идея за директно построяване на минимален КА [1].

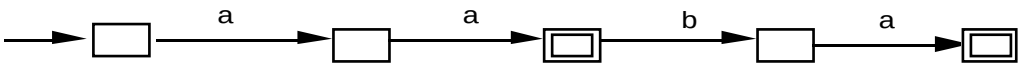
Освен понятията *обикновено* и *крайно* състояние (означавани както е прието с едно или съответно две кръгчета), при построяването се въвеждат две нови понятия: част от състоянията на построения минимален КА са *"временни"* (означени с едно или две квадратчета), а други "*постоянни*" (означени с кръгчета). Решение кое от временните състояния се трансформира в постоянно се взема на ***i***-тата стъпка, в зависимост от големината на общия префикс между ***i***-тата дума и предишната в изходната лексикографска наредба. Това става в контекста на общата идея за построяване на дървовиден КА "отгоре надолу", илюстрирана на фиг. 2.

Построяването на минимален ацикл. КА се демонстрира чрез пример. Нека е даден речник {aa, aaba, aabbb, abaa, ababb, abbab, abc, acd, ... }.

На стъпка 1 се разглежда думата "aa" и се строи дървовиден ацикличен автомат (детерминиран и минимален) от временни състояния - вж. фиг. 3а.



Фиг. 3а



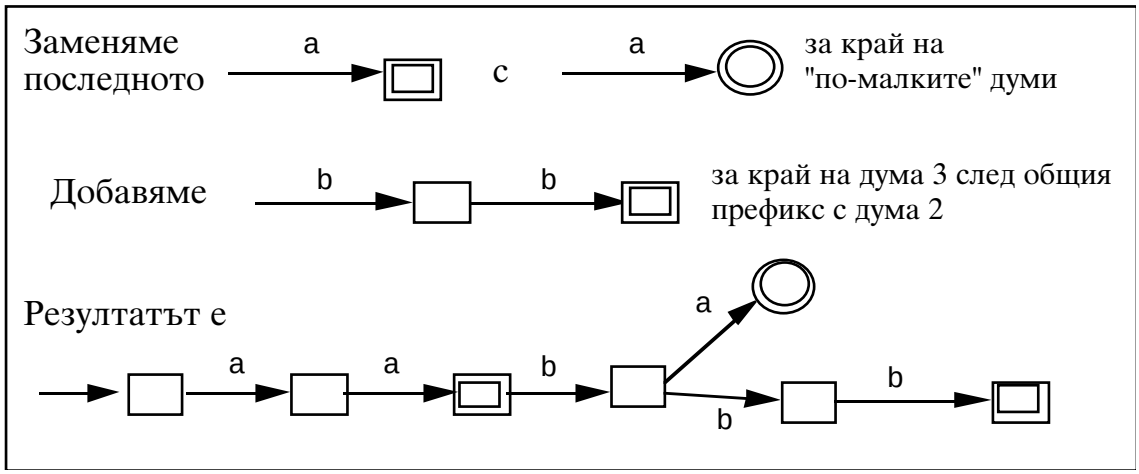
Фиг. 3б

Фиг. 3а,б. КА за езика {aa} и КА за езика {aa, aaba}.

Следващата дума, която идва на стъпка 2, е "aaba". Тя съдържа думата от предишната стъпка. Продължаваме КА от фиг. 3а с нови временни състояния, както е показано на фиг. 3б.

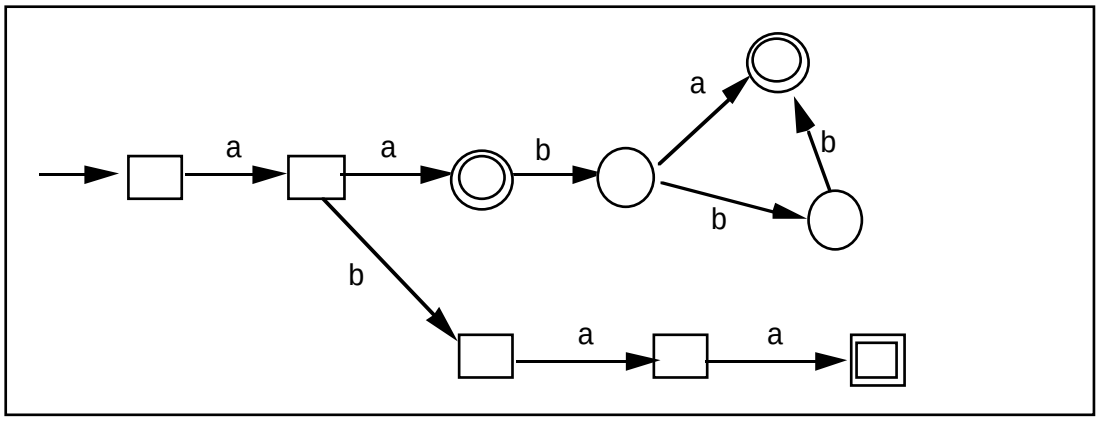
На стъпка 3 идва думата "aabbb" и се налага да се правят "разклонения" в дървото на КА. По-малка дума няма да дойде (поради сортировката в нарастващ ред). Значи е възможно някои състояния вече да се минимизират като постоянни, докато други са още в процес на промяна като временни. Разглежда се най-големият общ префикс на текущата и предишната думи,

т.е. на дума 3 (aabbb) и дума 2 (aaba). Този префикс в случая е PR=**aab**. С КА от фиг. 3б се постъпва както следва: разглеждат се думите, които започват с префикс по-малък от PR , и “техните” временни състояния се превръщат в постоянни, като се наблюдава дали това става по “минимален” начин. За думите с префикс равен или по-голям от PR се строят съответни временни състояния. На стъпка 3 се получава автоматът от фиг. 4.



Фиг. 4. КА за {aa, aaba, aabbb}.

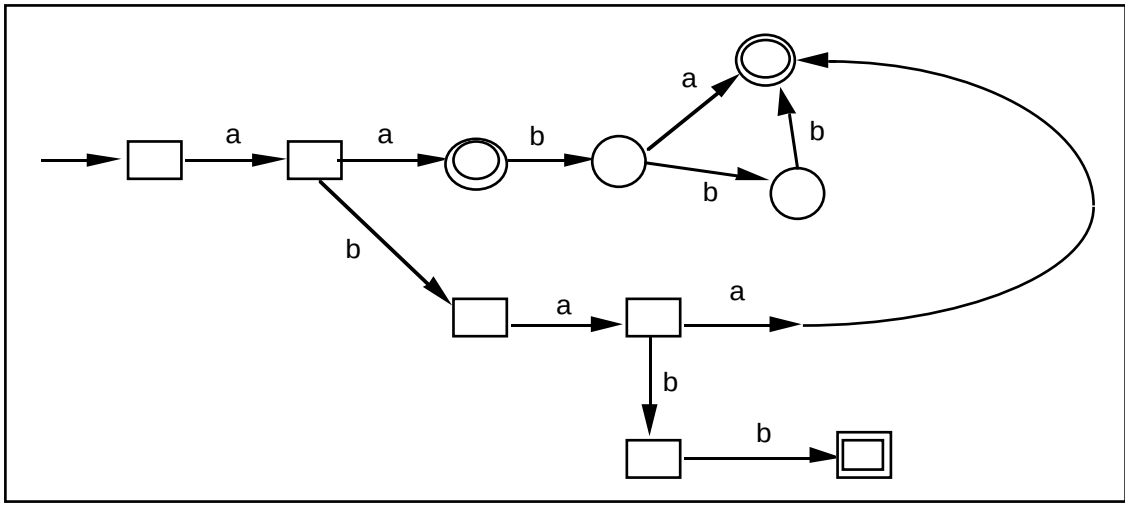
На стъпка 4 идва думата abaa, която има префикс **<u>a</u>** с предишната дума. Всички състояния след първото временно ще се превърнат в постоянни. Еквивалентните състояния се кодират в едно и така се получава “автомат, минимален освен в последната дума [1]”. Това е автоматът от фиг. 5.



Фиг. 5. КА за {aa, aaba, aabbb, abaa}.

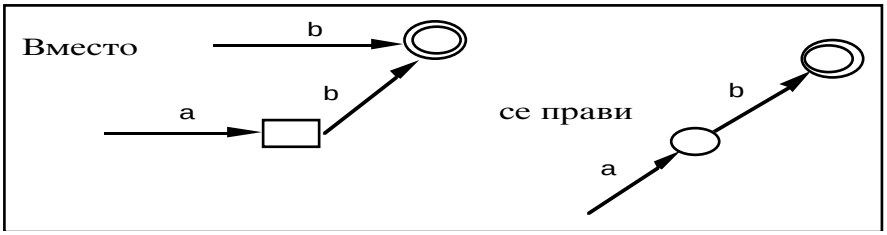
На следващата стъпка 5 идва думата ababb, общият й префикс с дума 4 е aba. Получава се автоматът от фиг. 6. При премахване на временното

крайно състояние ->a (получено на предишната стъпка) се строи минимален автомат, като вместо да се въведе ново постоянно състояние, се използва подходящо вече съществуващо.



Фиг. 6. КА за {aa, aaba, aabbb, abaa, ababb}.

И така нататък. Между постоянните състояния няма еквивалентни, т.е. те са част от минималния автомат. Временните състояния са необходими за последната дума, която се обработва. При определяне еквивалентността на постоянните крайни състояния се пренасочва дъгата или се създава нова с цел минимизиране броя на състоянията. (Пример е даден на Фиг. 7). В [1] е доказано по индукция, че тръгвайки от празния автомат, с по една дума на стъпка, получаваме “минимален освен в последната дума КА”. На края на списъка се получава минимален ацикличен КА освен в празната дума.



Фиг. 7. Минимизиране броя на еквивалентни състояния “отзад-напред”

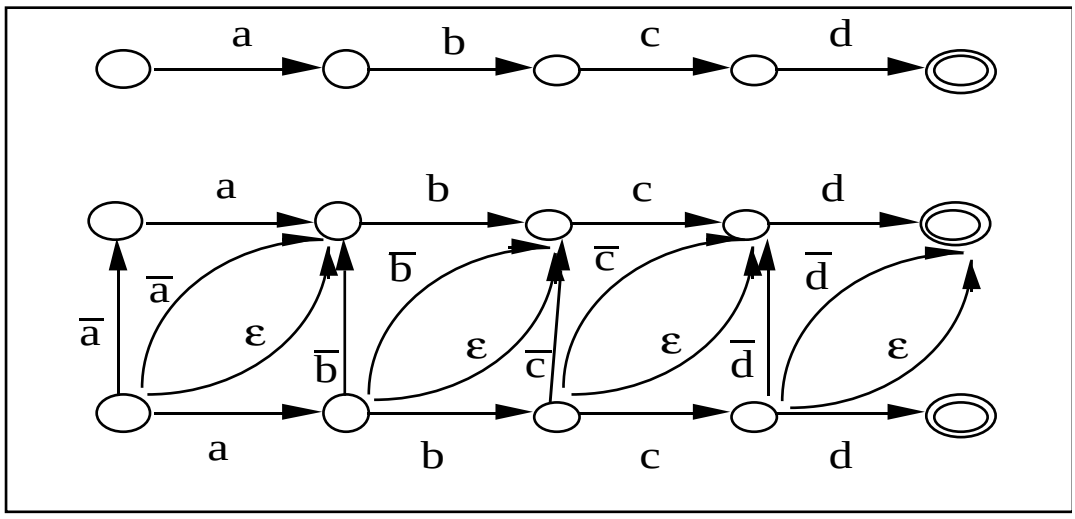
Сложността на този алгоритъм за минимизация е **n.log(m)**, където **n** е броят на символите в речника, а **m** е броят на състоянията в резултатния минимален автомат. Компресията на данните при граматически речници е забележителна: милиони думи от български/ руски граматически речник се кодират като минимален КА до 1 MB (при изходен текстов файл на речника от 35 MB).

III. Съвременни приложения на автоматите - речници

По даден КА, речникът се възстановява много бързо при обхождане в дълбочина по лексикографска наредба. Обединение/сечение/разлика на речници се получават лесно чрез обхождане на техните КА по лексикографска наредба и сравняване дума по дума, при което се добавят/запазват/махат състояния.

Речници-КА се прилагат при разработка на **spell-checkers**, за бързо намиране на дума от речника, която е близко до "неправилната" в текста (тази неправилна дума липсва в речника). Подобен проблем възниква и при сканирането, в **OCR софтуер,** когато трябва да се реши коя е входната дума (в случай, че не се разпознава дума от речника).

И при двете приложения се изисква да се намери дума, "близка" до даден низ според дадена метрика. При корекцията на правописни грешки близостта е "минимален брой елементарни редакции" (изтриване, замяна, вмъкване на символ или съседна транспозиция, т.е. разместване). При сканирането честите грешки са свързани със "смесване и разделяне на символи (merge & split)", напр. 2 символа са разпознати като един или един символ трябва да бъде разделен на два (cl и d). Задачата е да се намерят всички думи от речника, които са на дадено разстояние от текущия низ. Тривиално, но неефективно решение е да се генерират всички низове на дадено разстояние от текущия и да се проверява дали са думи от речника.



Фиг. 8. Горе - КА за разпознаване на езика {abcd}, долу - КА за разпознаване на всички думи на разстояние 1 от abcd. "Не-а" (а-черта) като етикет на прехода означава символ, различен от а; епсилон-преход означава изтриване на последния символ

Идеята е да се използват динамично генерирани КА, за да се строят всички низове на дадено разстояние от текущия низ, и после да се гледа кои от тези низове са думи в основния речник (чрез сечение на автоматите). Пример: да построим всички низове на разстояние 1 от думата abcd? (вж. фиг. 8).

Автоматът на фиг. 8 е недетерминиран. В [2] е доказано, че съществува ефективен начин да се построи за линейно време минимален КА, разпознаващ езика на всички думи, които са на дадено разстояние от дадена дума. Значи приложенията за spell-checker и OCR софтуер използват сечение на речника на системата с текущо-построения минимален КА. Двата автомата се обхождат паралелно и се извеждат думите от (естествения) език, които са на дадено разстояние от разглежданата дума. Късите думи практически се разпознават на разстояние 1, другите обикновено на 2 и рядко се налагат пресмятания на растояние 3-4.

Литература:

1. *Докторска дисертацията* на Стоян Михов, http://www.lml.bas.bg/~stoyan/

2. Klaus U. Schulz and Stoyan Mihov, *Fast String Correction with Levenshtein-Automata*. CIS-Bericht-01-127, Centrum fur Informations-und Sprachverarbeit-ung, Universitaet Muenchen, 2001. http://www.lml.bas.bg/~stoyan/publications.html)

3. Манев, Кр. *Дискретна математика.* Учебник за СУ, ФМИ.