Формално определение на езиците за програмиране. Граматики.Класификация. Бакус-Наур форма

Азбука

- *Азбука* се нарича крайно множество от символи (букви). Примери за символи от азбука са букви, цифри, пунктуационни знаци.

Примери:

  - Множеството {0,1} представлява *двоична азбука.*
  - *ASCII* е пример за азбука, използвана в множество приложения и системи.
  - *Unicode* включва около 100,000 символи от различни езици.

- *Низ* или *дума* (*string*) над дадена азбука се нар. крайна последователност от символи от азбуката. Дължина на низ *s,* отбелязвана с |s|, се нар. броят на символите в s.

  Примери:

  - **banana** е низ с дължина 6.
  - *Празен низ* (*empty string)* се отбелязва с **ε** е низ с дължина 0.

Формален език

- Ако **x** и **y** са низове, конкатенацията на **x** и **y**, означена като **xy**, е низ, получен чрез съединяване на **x** и **y**. Например, ако **x** = dog и **y** = house, тогава **xy** = doghouse. За празния стринг εs = sε = s.
- ***Език*** се нар. крайно множество от низове (думи) над дадена азбука. Това определение е много широко.

Формална граматика

- ***Граматика*** е средство за описание и анализ на формални езици. Състои се от ***правила***, които описват как се получават валидни изречения на езика.
- Пример от граматиката в английския език:

  изречение → <подлог><глаголна_фраза><допълнение>

  подлог → This | Computers | I

  глаголна_фраза → <наречие> <сказуемо> | <сказуемо>

  наречие → never

  сказуемо → is | run | am | tell

  допълнение → the <същ._име> | a <същ._име> | <същ._име>

  същ._име → university | world | cheese | lies

Правилата се нар. още *продукции.* Стрелката в правило се чете като „поражда“.

Чрез прилагане на правилата *се* получават изречения:

This is a university.

Computers run the world.

I am the cheese.

I never tell lies.

Пример за прилагане на правилата за получаване на горното изречение.

sentence → <subject> <verb-phrase> <object>

→ This <verb-phrase> <object>

→ This <verb> <object>

→ This is <object>

→ This is a <noun>

→ This is a university

Пример 2: Операторът **if-else** в Java има следната форма

if (израз) оператор **else** оператор

Тази форма означава, че операторът **if-else** се получава чрез конкатенация (съединяване) на ключовата дума **if** с отваряща скоба, израз, затваряща скоба, оператор, ключовата дума **else** и друг оператор. С използване на променливата *expr*, означаваща израз и променливата *stmt*, означаваща оператор, горното правило може да се запише по следния начин:

stmt → **if** (expr) stmt **else** stmt

В правилата елементи като ключовата дума **if** и скобите се нар. ***терминални символи (терминали)***. Променливи като *expr* и *stmt* представляват последователности от терминални символи и се нар. ***нетерминални символи*** (***nonterminals)***.

Дефиниции

- *граматика* - множество от правила за получаване на валидни изречения от езика.
- *нетерминал* – символи от граматиката, които могат да бъдат заместени (разширени) с други символи.
- *терминал* - елемент от езика; **не мога да бъдат заместени с никакви други символи**. След получаване на терминален символ не може да бъде приложено следващо правило.
- *Правило (продукция)* – правило от граматиката за заместване на символи. Общата форма за прилагане на правило с нетерминали е:

$$X \rightarrow Y_1 Y_2 Y_3 \ldots Y_n$$

Нетерминалът X е еквивалентен на конкатенацията от символи $Y_1Y_2Y_3...Y_n$. Правилото означава, че при всяко срещане на X, то може да се замести с последователност $Y_1Y_2Y_3...Y_n$. Когато се получи последователност от символи, в която никой от символите не може да се замести с друг, това означава, че тези символи са терминали. Такъв низ се нар. *изречение.* **В термините на програмните езици изречението представлява синтактично коректна, завършена програма.**

- *извеждане (derivation)* – такава последователност от прилагане на правилата, при която се получава крайна последователност от терминали. *Ляво извеждане (left-most derivation)* имаме, когато винаги заместваме най-левия нетерминал при прилагане на правилата. Възможно е и *дясно извеждане (right-most derivation)*.

- *стартов символ* - нетерминален символ, от който започва прилагането на правилата (извеждането):

$$S \rightarrow X_1X_2X_3...X_n$$

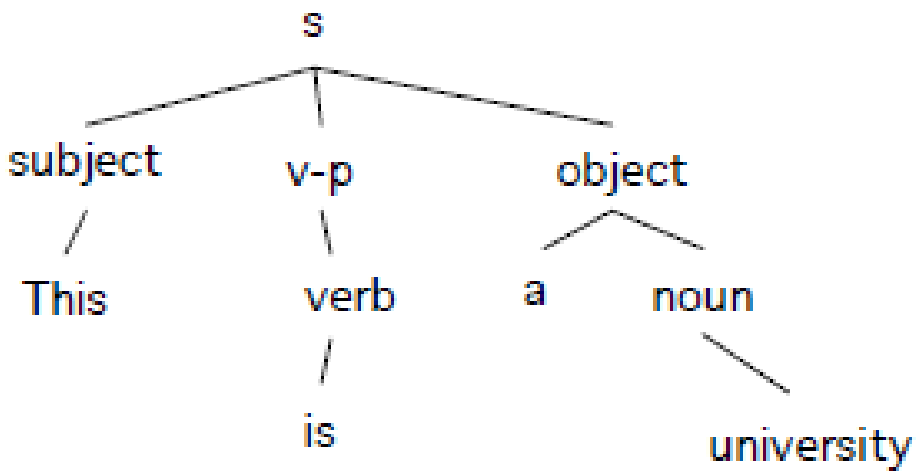
Всички изречения на езика се извеждат от S чрез последователно заместване на символи в съответствие с правилата.

- *празен символ ε* - понякога се налага даден символ да бъде заменен с празния символ. Например: $A \rightarrow B \mid \varepsilon$.
- *БНФ (Бакус-Наур форма)* - начин за описание на програмни езици чрез формални граматики и правила.

Представяне на извеждането

Прилагането на правилата за извеждане на изречение може да се представи по два начина:

- Първият начин е извеждане, при което правилата се прилагат стъпка по стъпка и на всяка стъпка се замества нетерминал.
- Вторият начин е чрез дърво. То показва как се заместват символи в йерархичен вид. Пример за извеждане на изречението "This is a university":



Първият начин показва реда, в който са приложени правилата за извеждане на изречение. Дадено изречение може да бъде изведено с различни последователности от прилагане на правилата (напр. ляво или дясно извеждане).

Дървовидното представяне на показва реда на прилагане на правилата за получаване на изречение. На дадено изречение съответства единствено дърво.

Формално определение за граматика

Граматика $G = (V_T, V_N, P, S)$ се състои от четири компонента:

1. Множество от терминални символи (V_T)
2. Множество от нетерминални символи (V_N)
3. Множество от правила (P)
4. Стартов символ (S).

Зависимости

$V_T \cap V_N = \emptyset$

$V = V_T \cup V_N$

Ако имаме двойка символи (α,β), където

$\alpha \in V^+$, $\beta \in V^*$, тогава

$\alpha \rightarrow \beta$ е правило от P

Стартов символ $S \in V_N$

където

V^* - множество от всички низове **плюс** празния низ ε.

V^+ - множество от всички низове **без** ε.

Обобщения

Граматиките се задават чрез *правила (продукции)*, като първото правило е за стартовия символ.

Цифрите, символи като < , <=, while, if и др. са терминали. Обикновено с главна буква се означава нетерминал, а с малки букви се означават терминали.

Правилата могат да се обединяват чрез специалния символ “**|**”, който означава „ИЛИ„.

Пример:

Граматика, която поражда низове от език $\{a^m b^n \mid m,n >= 0\}$ се дефинира по следния начин:

$G_1 = (\{a, b\}, \{S, A, B\}, P, S\})$

$S \rightarrow AB$

$A \rightarrow aA$

$A \rightarrow \varepsilon$

$B \rightarrow bB$

$B \rightarrow \varepsilon$

Низ **aaabb** се поражда от тази граматика чрез прилагане на следните правила:

$S \rightarrow AB \rightarrow aAB \rightarrow aaAB \rightarrow aaaB \rightarrow aaabB \rightarrow aaabb$

Еквивалентност

Език L(G), описан чрез граматика G, е множество от изречения, породени от граматиката G.

Изреченията на даден език могат да бъдат породени от повече от една граматика.

Две граматики G и G' са *еквивалентни,* ако езиците, L(G) и L(G'), които пораждат, са еднакви.

Пример: Граматика, която е еквивалентна на граматиката G_1:

Еквивалентност

$G_2 = (\{a, b\}, \{S, Y\}, P, S\})$

$S \rightarrow aS$ $\quad\quad$ $Y \rightarrow b$

$S \rightarrow a$ $\quad\quad$ $Y \rightarrow bY$

$S \rightarrow b$ $\quad\quad$ $Y \rightarrow \varepsilon$

$S \rightarrow bY$

$S \rightarrow Y$

Извеждане на низ aaabb:

$S \rightarrow aS \rightarrow aaS \rightarrow aaaS \rightarrow aaabY \rightarrow aaabb$

$G_2 \quad \Xi \quad G_1$

Горните правила могат да бъдат обединен с използване на символ ИЛИ - “|” :

$S \rightarrow aS \mid a \mid b \mid bY$

$Y \rightarrow b \mid bY \mid \varepsilon$

Йерархия на граматики

- Американският лингвист Ноам Хомски е извършил солидни изследвания върху формални граматики.
- В йерархията (класификацията) на Хомски съществуват четири типа формални граматики, започващи от тип 0 (най-обща) до тип 3 (най-рестриктивна).
- Ако граматиката има повече ограничения към правилата, тя е по-лесна за описание и реализация, но е с по-малка мощност (т.е. поражда по-малко низове в езика).

Йерархия на Хомски

- Тип 0: ***без ограничения***. Този тип граматики са най-общи. Правилата са във вида u->v, където u и v са произволни низове от V. Няма ограничения към правилата, освен че в лявата страна на правило не може да има празен низ.
- Тип 1: ***контекстно-зависими граматики (context-sensitive).*** Правилата са във вида uXw –> uvw, където u, v и w са произволни низове от символи от V, където v не е празен низ, а X е единствен нетерминал. Това означава, че X може да бъде заместен от v, само ако е обграден от символи u и w. (т.е. в конкретен контекст).

Йерархия на Хомски

- Тип 2: ***контекстно-свободни граматики (context-free)***. Правилата са във вида X–> v, където v е произволен низ от символи от **V,** а X е единствен нетерминал. Навсякъде, където се среща X, той може да бъде заместен с v (независимо от контекста).
- Тип 3: ***регулярни граматики.*** Правилата са във вида X–> a, X–> aY, or X–>ε, където X и Y са нетерминали, а е терминал. Т.е., лявата страна трябва да е единствен нетерминал, а дясната страна трябва да е празен низ, единствен терминал или единствен нетерминал. Тези граматики са най-ограничени по отношение на брой породени символни низове.

Йерархия на Хомски

Граматики от тип 3 (G3) са най-лесни за обработване поради липса на рекурсивни правила.

За някои класове граматики от тип 2 (G2) съществуват ефективни анализатори.

Въпреки че граматики от тип 1 (G1) и тип 0 (G0) са по-мощни от G2 и G3, те имат по-малко практическо приложение поради невъзможността на създаване на ефективни анализатори за тях.

При създаване на транслатори на програмни езици най-често се използват контекстно-свободни граматики (G2), които обикновено се означават като CFG (Context-Free Grammars).

Регулярни изрази

Регулярни изрази – правила за точно дефиниране на множества от низове, които са валидни в даден формален език. Правилата се задават със следните три оператора:

- Съединяване (конкатенация) xy
- Алтернатива x|y x или y
- Повторение x* x се повтаря 0 или повече пъти

Формално регулярните изрази се дефинират чрез следните рекурсивни правила:

1) Всеки символ от азбуката на формален език е регулярен израз
2) **ε** е регулярен израз
3) Ако **r1** и **r2** са регулярни изрази, то **(r1)**, **r1r2**, **r1 | r2, r1*** са също регулярни изрази
4) Никакъв друг израз не е регулярен

Регулярни изрази

Чрез регулярни изрази мога да бъдат дефинирани лексеми (думи) в програмните езици.

Пример за регулярен израз за дефиниране на цяло число, съставено от една или повече цифри:

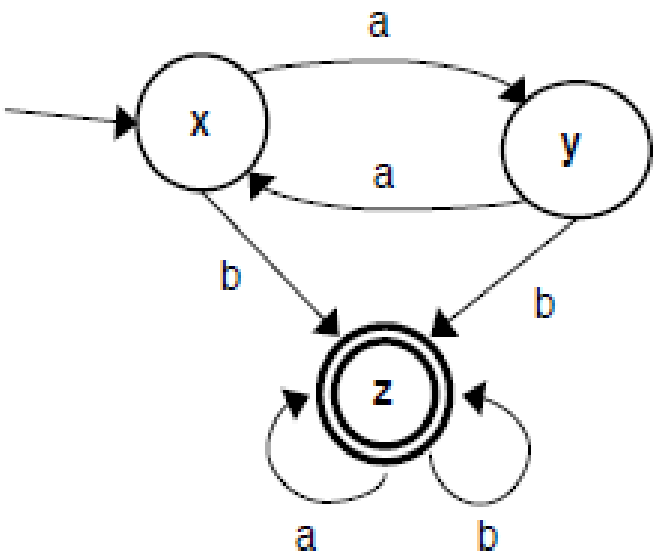
(+ е специален символ в регулярния израз, който означава 1 или повече повторения)

(0|1|2|3|4|5|6|7|8|9)+

Крайни автомати

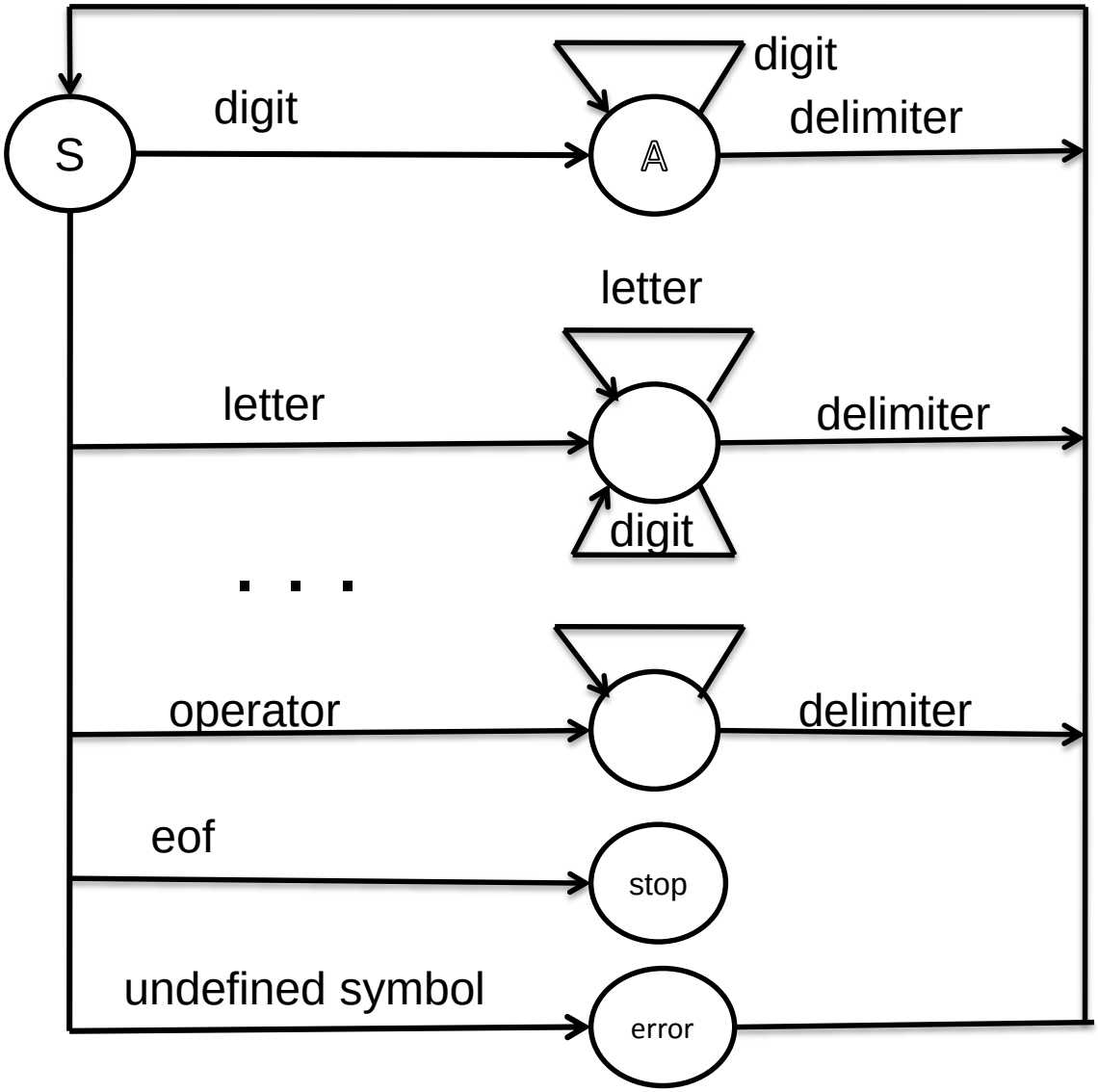
След като лексемите са дефинирани чрез регулярни изрази, можем да създадем краен автомат за тяхното разпознаване. *Крайният автомат (finite automata)* се състои от:

1) Краен брой състояния, едно от които е начално, а някои състояния (или нито едно) са крайни.
2) Азбука Σ от възможни входни символи.
3) Краен брой преходи, които дефинират за всяко състояние при даден символ от абуката кое е следващото състояние.



		a	b
(start)	x:	y	z
	y:	x	z
(final)	z:	z	z

Краен автомат за разпознаване на лексеми в Pascal



A in more details