Jazyk

Jakákoli množina řetězců , konečná nebo nekonečná, je formální jazyk. Nezáleží na tom, pomocí jakých znaků jsou jednotlivé řetězce jazyka napsány, podstatné ale je, aby jich byl konečný počet. Prvek jazyka, tedy jeden z řetězců, z nichž se jazyk skládá, se nazývá slovo daného jazyka.
Ve formálních jazycích neexistují mnohé další jevy, na které jsme zvyklí z přirozených jazyků, jimiž mluvíme. Například slovní druhy, věty složené z jednotlivých slov apod.

Příklad 1

L_1 = {toto, je, jazyk, obsahující, 14, slov, a, slouží, pouze, demonstraci, pojmů, nemá, jiný, význam}.
Na pořadí prvků, které jsou uvedeny v tomto výčtu jazyka, nezáleží. To, že výčet prvků připomíná českou větu, je shoda okolností, nic víc.
Nejdelší slovo L_1 má délku 11 znaků, nejkratší má délku jeden znak.
Abeceda, nad níž je jazyk L_1 vytvořen, tj. množina všech znaků, které se objeví alespoň v jednom slově jazyka L_1, je A_1 = {a, b, c, d, e, i, í, j, k, l, m, n, o, p, r, s, t, u, ů, v, y, ý, z, ž, 1, 4}.
Je zbytečné z hlediska teorie formálních jayzků uvažovat větší abecedu, např. celou českou abecedu včetně číslic spod.

Příklad 2

L_2 = {10, 101, 1011, 10111, 101111, 1011111, …}.
Tři tečky v zápisu L_2 naznačují, že jazyk L_2 je nekonečný a spoléhají na přitom na čtenářův důvtip, který má napovědět, jak všechna další slova jazyka vypadají. Jmenovitě jsou to posloupnosti (řetězce skládající se z) nul a jedniček, z nichž každý začíná dvojicí znaků 10 a pak následuje libovolný počet (i nulový) znaků 1.

Příklad 3

```
#include <stdio.h>
int main(int argc, char *argv[]) {
  printf("Ahoj");
}
```

Uvedený kód můžeme chápat jako jediný řetězec (obsahující znaky konce řádku), a tudíž jako jedno slovo určitého jazyka L_3 . V tomto případě by to mohl být jazyk všech syntakticky správných programů v jazyce C. Každé slovo L_3 pak představuje jeden kompletní program v C.

Gramatika

Některé jazyky jsou natolik přehledné a jednoduše strukturované, že je možno sestavit formální mechanizmus na výrobu libovolného slova daného jazyka. Tento mechanizmus se nazývá gramatika.
Gramatika pracuje tak, že z jediného počátečního symbolu (který je stejný pro gramatiky všech jazyků) postupně pomocí odvozovacích pravidel vytváří slovo jazyka. K tomu využívá pomocné, tzv. neterminální symboly. Neterminální symboly se ve vytvářeném slově nahrazují podle pravidel jinými podřetězci tak dlouho, dokud nevznikne slovo daného jazyka. To je zapsáno pomocí znaků abecedy daného jazyka a pro gramatiku jsou to tzv. terminální znaky.

Příklad 4

Uvedeme pravidla gramatiky G_2 generující jazyk L_2 z příkladu 2

S → 10
S → 10A
A → 1A
A → 1

Při generování slova postupujeme takto:
Nejprve napíšeme startovní symbol S. To je pokaždé povinný úkon. Dále již cyklicky opakujeme neustále stejnou akci:
V napsaném řetězci zvolíme libovolný neterminální symbol (naše gramatika má jen dva, *S* a *A*) a zvolíme libovolné pravidlo, které má tento symbol na levé straně. Nyní tento neterminální symbol v napsaném řetězci nahradíme řetězcem na pravé straně pravidla. Při této akci nahrazujeme pouze jediný výskyt neterminálního symbolu, nikdy více najednou. V dalším kroku můžeme opět volit libovolný neterminální symbol, jeho výskyt i pravidlo.
Cyklus ukončíme, když v napsaném řetězci nezbývá žádný neterminální symbol, který bychom mohli nahradit.

Vygenerujeme slovo 10111 jazyka L_2.

1. Píšeme *S*.
2. Použijeme pravidlo $S \rightarrow 10A$, získáme řetězec 10A.
3. Použijeme pravidlo $A \rightarrow 1A$, tj. v řetězci 10*A* nahradíme symbol *A* řetězcem 1*A*, získáme řetězec 101*A*.
4. Použijeme pravidlo $A \rightarrow 1A$, tj. v řetězci 101*A* nahradíme symbol *A* řetězcem 1*A*, získáme řetězec 1011*A*.
5. Použijeme pravidlo $A \rightarrow 1$, tj. v řetězci 101*A* nahradíme symbol *A* řetězcem 1, získáme řetězec 10111, generování slova 10111 je u konce.

Důležitou vlastností každé gramatiky je to, že více pravidel může mít na levé straně stejný neterminální symbol. Když tento symbol nahrazujeme ve vytvářeném řetězci, můžeme si vybrat pravidlo a tím i ovlivnit tvar výsledného řetězce.
Pro přehlednost užíváme často zkrácené značení, kdy všechna pravidla se stejnou levou stranou (stejným symbolem na levé straně) píšeme na jeden řádek: Vlevo napíšeme společný neterminální symbol, vpravo pak odděleně všechny příslušné pravé strany. V našem příkladě je to:

$S \rightarrow 10 \mid 10A$
$A \rightarrow 1A \mid 1.$

Shrnutí

Gramatika G_2 generující jazyk L_2 obsahuje
1. Abecedu terminálních symbolů {0, 1}.
2. Abecedu neterminálních symbolů {*S*, *A*}.
3. Startovní symbol *S*.
4. Pravidla
   $S \rightarrow 10 \mid 10A$
   $A \rightarrow 1A \mid 1.$

Z těchto čtyř částí – terminální symboly, neterminální symboly, startovní symbol, pravidla – se skládá každá gramatika.

Příklad 5

Chceme automatizovat tvorbu zápisů přirozených čísel. Připomeňme si, že samotné číslo nula mezi přirozená čísla nepatří a že se zápis přirozeného čísla skládá pouze z číslic 0,1, ..., 9, přičemž na prvním místě zleva (nejvyšší řád) nesmí stát nula. Budeme proto rozlišovat mezi nejlevější číslicí (která nesmí být nula) a ostatními číslicemi v zápisu čísla.
Označme písmenem *N* nejlevější (nenulovou) číslici a řetězec představující ostatní číslice v zápisu označme písmenem *Z* (jako zbytek). Zbytek může být i prázdný řetězec, který značíme symbolem ε. Všechna přirozená čísla pak mají stejný tvar *NZ*. Pravidla naší gramatiky pak mohou být:

1. $S \rightarrow NZ$
2. $N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
3. $Z \rightarrow 0Z \mid 1Z \mid 2Z \mid 3Z \mid 4Z \mid 5Z \mid 6Z \mid 7Z \mid 8Z \mid 9Z \mid \varepsilon$

Zápis čísla 807 vytvoříme ve čtyřech krocích. Pro jednoduchost budeme zaznamenávat postupně pouze řetězec vznikající ze startovního symbolu a jednotlivé kroky označíme číslem řádku, na němž leží použité pravidlo:

$$S \xrightarrow{1.} NZ \xrightarrow{2.} 8Z \xrightarrow{3.} 80Z \xrightarrow{3.} 807Z \xrightarrow{3.} 807 .$$

Pravidla můžeme používat v různém pořadí:

$$S \xrightarrow{1.} NZ \xrightarrow{3.} N0Z \xrightarrow{3.} N07Z \xrightarrow{3.} N07 \xrightarrow{2.} 807 .$$

Soustava našich pravidel je význačná tím, že jejich libovolnou aplikací získáme vždy zápis přirozeného čísla. Soustava je "fool-proof", jejím správným použitím můžeme získat kterékoli přirozené číslo a zároveň nemůžeme získat nic jiného než přirozené číslo. To jeden z důvodů, proč se gramatiky používají.

Příklad 6

Rozšíříme příklad 5 a vytvořme pravidla pro generování zápisu celých čísel. Nejprve považme krátký rozbor: Celá čísla lze rozdělit na záporná, nulu, a kladná. Kladná splývají s přirozenými a ty už vytvářet umíme. Zapsat nulu separátně také nebude problém a záporná se zapisují stejně jako kladná, jen mají navíc na začátku znaménko mínus. Bude nám stačit jen úprava prvního pravidla:

1. $S \rightarrow 0 \mid -NZ \mid NZ$
2. $N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
3. $Z \rightarrow 0Z \mid 1Z \mid 2Z \mid 3Z \mid 4Z \mid 5Z \mid 6Z \mid 7Z \mid 8Z \mid 9Z \mid \varepsilon$

Nesmíme zapomenout, že se nyní rozšířila terminální abeceda o znak '–' (mínus).

Příklad 7

Na digitálním časovém displeji se mění čas po jedné sekundě. Zápis okamžitého času má formát *HH*:*MM*:*SS*. Vytvoříme gramatiku, která popisuje všechny řetězce tohoto formátu. (Laskavý čtenář snadno zdůvodní, proč je všech těchto řetězců právě 86400.)

Zřejmě minuty i sekundy mají zcela identický formát, jsou to řetězce 00, 01, ..., 59, označme tento formát neterminálem *B*. Hodiny jsou retězce 00, 01, ..., 23, označme tento formát neterminálem *A*.

Terminálními symboly budou číslice 0, …, 9 a znak ‘:’ (dvojtečka). První pravidlo gramatiky tedy můžeme zapsat:

1. $S \rightarrow A{:}B{:}B.$

Zbývá určit, jak expandovat *A* a *B*.

Expanze *A* (hodiny): je-li první číslicí 0 nebo 1 může být druhá číslice libovolná, je-li první číslicí 2, může druhá nabývat hodnot pouze 0, ... 3. Číslice 0 nebo 1 označme společným symbolem C_{01} číslice 0, ..., 9 označme společným symbolem C_{09} a číslice 0, ..., 3 symbolem C_{03} . Potřebná pravidla pak budou:

2. $A \rightarrow C_{01}C_{09} \mid 2C_{03}$
3. $C_{01} \rightarrow 0 \mid 1$
4. $C_{03} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid$
5. $C_{09} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Expanze *B* (minuty nebo sekundy): *B* se skládá ze dvou číslic, přičemž první číslice může nabývat hodnot jen 0, ..., 5, označme ji symbolem C_{05} a druhá číslice může nabývat hodnot 0, ..., 9, pro ně již máme označení C_{09} i příslušné pravidlo 5.

6. $B \rightarrow C_{05}C_{09}$
7. $C_{05} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5$

Naše odvození pravidel gramatiky pro tuto úlohu bylo úmyslně pozvolné, čtenář toužící po zhuštěnějším výrazu snadno ověří, že bychom stejně dobře mohli použít následující sadu pravidel:

1. $S \rightarrow C_{01}C_{09} : C_{05}C_{09} : C_{05}C_{09} \mid 2C_{03} : C_{05}C_{09} : C_{05}C_{09}$
2. $C_{01} \rightarrow 0 \mid 1$
3. $C_{03} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid$
4. $C_{05} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5$
5. $C_{09} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9.$

Je vidět, že stejnou množinu řetězců (stejný jazyk) můžeme generovat pomocí rozličných (méně nebo více přehledných) gramatik.

Širší perspektiva

V dosavadních příkladech jsme pracovali s gramatickými pravidly poměrně speciálního tvaru: V levé části se vždy nacházel jediný neterminální symbol. Ve všeobecném případě (tzv. neomezené gramatiky v Chomského hierarchii) může levá část pravidla obsahovat libovolnou kombinaci terminálních a neterminálních symbolů obsahující alespoň jeden neterminální symbol. Toto pravidlo se pak interpretuje takto: Pokud ve vytvářeném řetězci existuje podřetězec shodný s levou částí pravidla, je možno jej nahradit pravou částí pravidla.
Uvádíme tuto poznámku pro úplnost, gramatiky, se kterými se setkáme, takzvané regulární a bezkontextové, mají na levé straně vždy jen jediný neterminální symbol