Karel Pala

Automatická syntaktická analýza českého textu (Experiment)

Kybernetika, Vol. 4 (1968), No. 4, (318)--330

Persistent URL: http://dml.cz/dmlcz/124405

KYBERNETIKA ČÍSLO 4, ROČNÍK 4/1968

Automatická syntaktická analýza českého textu*

(Experiment)

KAREL PALA

V článku se referuje o experimentech s automatickou syntaktickou analýzou několika úryvků českého textu. Experimenty byly provedeny v květnu a červnu 1966 na samočinném počítači Elliott 4100 a v říjnu 1966 na samočinném počítači SAAB D 21. Algoritmus analýzy se opírá o jeden způsob syntaktické analýzy nekontextových jazyků a používá se v něm zásobníkové paměti. Program analýzy je uveden v jazyce Algol.

1. ÚVOD

Cílem článku je podat stručný přehled problematiky syntaktické analýzy přirozeného jazyka – češtiny a ukázat, jak lze takovou analýzu provádět prostřednictvím samočinného počítače. Zatím byly na samočinném počítači analyzovány úryvky textů v rozsahu 10–35 vět a jednotlivé věty z českých vědecko-technických textů (celkem 111 vět a souvětí).

Aby bylo možno provádět automatickou syntaktickou analýzu vět přirozeného jazyka, je nutno mít k dispozici: 1. *vhodnou formalizovanou gramatiku*, která popisuje systém příslušného přirozeného jazyka; 2. *vlastní proceduru* (algoritmus), která na základě formalizované gramatiky vložené do počítače analyzuje syntakticky věty přirozeného jazyka [11].

2. SYNTAKTICKÁ ANALÝZA

Syntaktické popisy jazyků (přirozených i umělých) jsou v posledním desetiletí (1956–1966) konstruovány na základě nových typů gramatik, které se nazývají *generativní* gramatiky (dále GG) nebo *přepisovací systémy* [5], [2], [7].

* Předneseno na II. kybernetické konferenci, která se konala v Praze ve dnech 16.—19. listopadu 1965.

Jazyk L je pak definován jako množina vět (konečná nebo nekonečná), které jsou vytvořeny sřetězením konečné množiny prvků tvořících univerzální slovník *V*. Tato definice zahrnuje jak přirozené, tak umělé logické a programovací jazyky.

Při použití těchto gramatik vyvstávají dva základní požadavky: (i) *vymezit* (generovat) množinu gramaticky správných vět (jazyk); (ii) každé generované větě přiřadit *strukturní popis*, jímž se rozumí specifikování prvků, z nichž se věta skládá, specifikování jejich pořadí a vzájemného uspořádání, vztahů a jakékoli další informace o tom, jak je věta užívána a chápána v syntaktickém slova smyslu [4], [16]. Přitom je nutno poznamenat, že ve skutečnosti ani požadavek (i) nebyl ve vztahu k přirozeným jazykům dosud nikdy splněn [4], [15].

Na formální vlastnosti gramatik (tvar přepisovacích pravidel) lze klást postupně se zesilující omezení, tak je možno získat jednotlivé typy gramatik, které se od sebe liší svou *generativní silou*, tj. schopností generovat řetězy symbolů s rozdílnými formálními vlastnostmi (různé typy jazyků). Z tohoto hlediska existuje hierarchie GG a jimi generovaných jazyků [5].

Na základě některých typů GG můžeme formulovat dva základní druhy procedur: 1. *procedury generativní*, tvořené příslušnou GG a přesnými pravidly jejího fungování, které umožňují generovat jazyk *L*, tj. množinu všech gramaticky správných vět jazyka *L* a současně jim automaticky přiřadit odpovídající strukturní popisy; 2. *procedury rekognoskativní*, skládající se podobně z dané GG a pravidel jejího fungování, které dovolují rozhodnout, zda předložený řetěz je gramaticky správnou větou jazyka *L*, a jestliže ano, přiřadit mu odpovídající strukturní popis.

Rozdíl mezi procedurami 1 a 2 je i v tom, že generativní procedury vymezují množiny rekurzívně spočetné, kdežto rekognoskativní procedury vymezují množiny rekurzívní [5].

Procedura 1 nebo 2 umožňuje tedy ve spojení s vhodnou GG vymezit jazyk *L* a poskytnout jeho syntaktický popis. To však platí v plném rozsahu jen o umělých jazycích, jakými jsou např. programovací jazyky. U přirozených jazyků je situace podstatně jiná: daný přirozený jazyk, např. čeština, existuje nezávisle na naší vůli a lingvisté se snaží popsat jej tak, aby k němu mohli sestavit vhodnou GG, která by pak byla schopna generovat všechny gramaticky správné české věty a přiřazovat jim současně adekvátní strukturní popisy. Jinými slovy, jazyk *L* (čeština) je dán a lingvisté se snaží definovat k němu vhodnou GG a tím i explicitně popsat jeho syntaktickou strukturu a ovšem i celý jazykový systém, tj. fonologii, morfologii, syntax a sémantiku [6]. Procedur 1 nebo 2 lze tedy použít k experimentálnímu ověření lingvistických koncepcí popisu jazykového systému např. tak, že je zaprogramujeme pro samočinný počítač, který pak v případě 1 generuje věty přirozeného jazyka s jejich strukturními popisy nebo v případě 2 věty analyzuje a poskytuje jejich strukturní popisy. Tento postup umožňuje současně ověřovat *adekvátnost* konstruovaných GG: jestliže věty generované počítačem jsou z větší části *gramaticky nesprávné* (to lze zjistit dotazem u informátorů), lze usuzovat na nízký stupeň adekvátnosti dané GG, podobně, jestliže v případě 2 počítač není schopen analyzovat část vstup-

ních vět nebo analyzovaným větám přiřazuje z hlediska lingvistických požadavků chybné strukturní popisy, lze z toho vyvodit závěr, že použitá GG není dostatečně empiricky adekvátní. V některých případech se sice může stát, že informátoři budou váhat nebo odmítnou odpovědět, ve většině případů však jsou schopni bezpečně rozhodnout, že daná věta je gramaticky správná nebo nesprávná. Mezi těmito dvěma zřetelnými póly je pásmo přechodných případů [1].

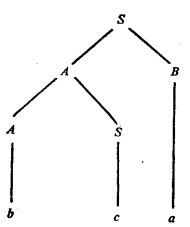
2.1. Bezkontextové gramatiky

Pro popis syntaxe přirozených i umělých jazyků se v poslední době užívá nejčastěji GG, které se nazývají *bezkontextové frázové gramatiky* (context-free, constituent-structure grammars) a které jsou podrobně popsány v pracích Chomského, Bar-Hillela, Ginsburga aj. [4], [5], [2], [7], proto nepokládáme za nutné podrobně o nich mluvit a uvádět základní pojmy.

Přejdeme přímo ke konkrétní problematice. Je dána bezkontextová (dále BK) gramatika $G_1 = (V_T, V_N, R, S)$, kde $V_N = \{S, A, B\}$, $V_T = \{a, b, c\}$ a R obsahuje následující přepisovací pravidla:

1. $S \rightarrow AB$
2. $A \rightarrow AS$
3. $A \rightarrow b$
4. $B \rightarrow a$
5. $S \rightarrow c$

Mějme ukončenou S-derivaci řetězu „*bca*" generovaného G_1. Takové derivaci lze jednoznačně přiřadit ohodnocený orientovaný graf-strom, který je frázovým ukazatelem řetězu „*bca*" (obr. 1). (Termínu „frázový ukazatel" užíváme vždy v souvislosti s konkrétní BK gramatikou, jinak lze užívat termínu „strukturní popis", viz [16].)



Obr. 1.

Analýzou terminálního řetězu vzhledem k dané BK gramatice G rozumíme přiřazení frázového ukazatele tomuto řetězu, existuje-li v gramatice G S-derivace tohoto řetězu.

V souhlasu s tím, co jsme řekli v odd. 2 o rekognoskativních procedurách, lze tedy tvrdit, že cílem většiny navržených a zaprogramovaných algoritmů je provést analýzu vstupních řetězů v právě formulovaném smyslu. Přiřadí-li počítač vstupnímu řetězu (větě) žádaný frázový ukazatel (dále FU), rozhodne tím současně, že analyzovaný řetěz symbolů (věta) náleží do jazyka L.

Jak ukazují ve svém článku Griffiths a Petrick [8], vlastní analýza vstupního řetězu, tj. získávání frázového ukazatele může postupovat několika způsoby: shora dolů, zdola nahoru, zleva doprava, zprava doleva. Některé algoritmy užívají zásobníkové paměti (pushdown store) a selektivních testů omezujících falešné kroky během analýzy, jiné pracují bez nich, dále některé algoritmy konstruují všechny možné FU vstupního řetězu, jiné konstruují jen jeden, který je pak z hlediska syntaktického popisu jazyka pokládán za správný. Rozdíly mezi jednotlivými algoritmy jsou i v tom, že se v nich užívá různě definovaných speciálních typů BK gramatik, např. tzv. *normálních* gramatik majících pravidla tvaru $A \to BC$, $A \to E$ nebo $D \to a$ nebo *standardních* BK gramatik s pravidly tvaru $A \to cB_1 \ldots B_n$ pro $n \leqq 1$ nebo *sekvenčních* gramatik.

Poměrně podrobný přehled algoritmů analýzy, jejich předběžnou klasifikaci a srovnání efektivnosti lze nalézt v již zmíněném článku Griffithse a Petricka. Částečný přehled algoritmů analýzy konkrétního přirozeného jazyka (angličtiny) je uveden v práci S. Kuna [10]. Přehled jednotlivých typů algoritmů nebudeme tu uvádět, protože by to přesahovalo rámec článku.

3. REKOGNOSKATIVNÍ PROCEDURA PRO ČEŠTINU

V této části článku uvedeme gramatiku definovanou pro jistý úsek češtiny a strukturu algoritmu analýzy. Algoritmus analýzy je zapsán v programovacím jazyce ALGOL v té podobě, v jaké byl ověřován na samočinném počítači.

3.1. Struktura gramatiky pro češtinu

V naší rekognoskativní proceduře je použito tzv. nespojité sekvenční gramatiky G_2 odlišné od normální BK gramatiky G_1 definované v oddíle 2.1. (gramatiky jako G_2 nejsou BK, jsou však s nimi slabě ekvivalentní). Přesněji $G_2 = (V_T, V_N, R', S)$ a platí podmínka sekvenčnosti, která říká, že neterminální slovník gramatiky G lze uspořádat do posloupnosti $A_1, \ldots, A_n$ tak, že je-li $A_i \overset{*}{\Rightarrow} \phi A_j \psi$, pak $j \geqq i$ pro každé i, j (Chomsky [5]).

V_T interpretujeme jako *terminální slovník*, tj. jako množinu symbolů, z nichž jsou sestaveny vstupní české věty. V_N interpretujeme jako *neterminální slovník*, tj. jako množinu symbolů reprezentujících gramatické kategorie pro syntaktický

popis vstupních českých vět. R' je konečná množina bezkontextových *přepisovacích pravidel* tvaru $B_1B_2 \rightarrow A$ nebo $B \rightarrow A$ a tzv. *nespojitých* pravidel $D_1 \ldots D_2 \rightarrow C$ (těchto pravidel použil poprvé V. Yngve v práci [13]). Lze říci, že pravidla v R' jsou vzhledem k pravidlům R z G_1 inversní, protože jsou v nich zaměněny pravé a levé strany, např. pravidlo $B_1B_2 \rightarrow A \in R'$ má v R tvar $A \rightarrow B_1B_2$ (formálně to však není podstatný rozdíl).

Dále platí, že $R' = T2A \cup T3A \cup T2B \cup T3B \cup T2C$, kde $T2A$, $T3A$, $T2B$, $T3B$, $T2C$ jsou disjunktní podmnožiny pravidel v R'. $T2A$ je podmnožina dvoučlenných pravidel ve tvaru $CB \rightarrow A$, $T3A$ rovněž, od $T2A$ se však liší tím, že pravidla v $T3A$ mají na pravé straně symbol S, tedy např. $DE \rightarrow S$. $T2B$ je podmnožina jednočlenných pravidel ve tvaru $E \rightarrow A$, $T3B$ rovněž, pravidla v $T3B$ mají však na pravé straně symbol S, tj. např. $F \rightarrow S$. $T2C$ jsou dvoučlenná pravidla ve tvaru $K \ldots J \rightarrow M$ a jsou určena k popisu tzv. *nespojitých složek*.

Množina R' byla rozdělena na jednotlivé podmnožiny proto, aby se zrychlil průběh analýzy a aby podmnožina $T2C$ byla procházena jen po prohledání všech předchozích podmnožin pravidel. Algoritmus analýzy je konstruován tak, že nejprve jsou procházena pravidla z podmnožiny $T2B$ a $T3B$, není-li hledání úspěšné, přechází se k pravidlům $T3A$ a $T2A$, jestliže ani pak není nalezeno hledané pravidlo, přechází se k pravidlům $T2C$ za předpokladu, že jsou splněny některé další doplňující podmínky (v zásobníkové paměti musí být minimálně tři symboly, z nichž jeden musí mít prioritu – viz podrobněji v závěru odd. 4). Toto uspořádání pravidel v R' zaručuje, že v daném okamžiku se prochází jen potřebná skupina pravidel, nikoliv celá množina R'.

V pravidlech $T3B$ a $T3A$ je použito více vyznačených výsledných symbolů (v G_1 je takový symbol jen jeden – S) a jsou to symboly $S0$, $S1$, $S2$, ..., $S9$, přičemž poslední pravidlo

$$\left\{\begin{matrix} S0 \\ S1 \\ \vdots \\ S9 \end{matrix}\right\} \rightarrow S$$

je vypuštěno. Dospěje-li se během analýzy vstupní věty k některému z uvedených symbolů, analýza věty je skončena a začíná se analyzovat další věta.

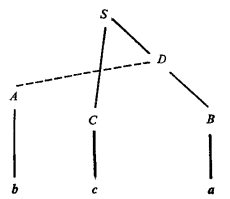
Cílem analýzy je pouze získat popis syntaktických struktur českých vět, proto současná verze gramatiky G_2 neobsahuje žádná lexikální pravidla tvaru $a_1a_2 \rightarrow A$. Vstupní věty jsou na začátku analýzy v takové podobě, jakou by měly po provedení morfologické analýzy. To znamená, že slova dané věty jsou nahrazena symboly označujícími jejich slovní druhy. Např. věta „Můj přítel píše disertační práci." je přepsána jako posloupnost symbolů $A1\ N1\ V3\ A4\ N4\ 999$ (999 je znak označující konec věty a znamená totéž, co v normální interpunkci tečka), která představuje vstup do samočinného počítače. Na takto upravenou větu aplikuje počítač pravidla z R' a konstruuje její frázový ukazatel. Důvodem k této úpravě byla skutečnost, že pří-

prava samostatného algoritmu pro morfologickou analýzu by byla velmi pracná a že se zaměřujeme především na syntaktickou stránku problému.

3.2. Lingvistická charakteristika množiny R'

Z lingvistického hlediska obsahuje množina R' jednak bezkontextová pravidla, která umožňují popisovat syntaktické struktury českých vět podle koncepce *bezprostředních složek*, jednak tzv. nespojitá pravidla. Podmnožina $T2A$ čítá nyní 1260 pravidel a zahrnuje pravidla popisující: a) slovesa a větné členy rozvíjející sloveso, tj. podle běžné terminologie objekt a příslovečné určení; b) substantiva a jejich rozvíjející adjektiva, tj. podmětové části vět, přívlastky a předložkové pády; c) koordinaci větných členů a vět a subordinaci vět. Podmnožina $T2B$ je tvořena 24 pravidly, která popisují přísudkové části vět. Podmnožina $T3A$ obsahuje 90 a $T3B$ 14 pravidel se symboly $S0, S1, \ldots, S9$ na pravé straně, např. $VP7\ NP \rightarrow S7$ nebo $VP6 \rightarrow S6$. Tato pravidla popisují větné typy, např. zatím se rozlišují věty oznamovací ($S7$) a uvnitř nich věty dvojčlenné s vyjádřeným i nevyjádřeným podmětem, některé typy vět jednočlenných, souvětí souřadná (slučovací a odporovací), souvětí podřadicí s různými typy vět vedlejších, zejména předmětových, přívlastkových a příslovečných.

Podmnožina $T2C$ se skládá z 51 speciálních tzv. nespojitých pravidel určených k popisu *nespojitých složek* (discontinuous constituents). Jde tu o případy, kdy dvě větné složky (slova nebo větné členy) patřící k sobě a tvořící spolu vyšší jednotku (větný člen) jsou od sebe odděleny nějakou jinou složkou, která je vložena mezi ně. Proto je třeba během analýzy použít pravidlo z podmnožiny $T2C$, které nám umožní zpracovat nejprve nespojité složky a teprve pak složku, která je vložena mezi ně. Tato situace je zachycena ve frázovém ukazateli na obr. 2.



Obr. 2.

V tomto případě jsou nespojitými složkami A a B, proto je třeba aplikovat pravidlo $A \ldots B \rightarrow D \in T2C$. Dále již analýza pokračuje s použitím ostatních skupin pravidel.

Použití pravidel $T2C$ je empiricky velmi výhodné, protože dovoluje pracovat s binárními pravidly i tam, kde by jinak bylo nezbytné použít pravidel typu $ABC \rightarrow D$

nebo dokonce $ABCF \rightarrow E$. Z lingvistického hlediska pravidla $T2C$ zachycují dosti časté případy, kdy mezi sloveso a jeho rozvíjející větný člen je vložen jiný větný člen, obvykle podmět, nebo případy, kdy příklonka „se" patřící ke svému slovesu je od něho oddělena a přesunuta za první přízvučné slovo ve větě, jak to vyžaduje rytmický princip slovosledu v češtině.

4. ALGORITMUS ANALÝZY

Algoritmus je založen na využití zásobníkové paměti (dále ZP) a FU vstupní věty je konstruován ve směru zdola nahoru a zleva doprava (ve smyslu Griffithsovy a Petrickovy klasifikace). Vstupní věta je analyzována na jeden průchod a výsledkem analýzy je pouze jeden frázový ukazatel. Použitím nespojité sekvenční BK gramatiky dostává celý algoritmus selektivní charakter, takže během analýzy nejsou nastupovány falešné cesty a není nutno se vracet. V tomto smyslu lze předpokládat, že náš algoritmus odpovídá algoritmu *selektivní přímé substituce* podle klasifikace, kterou uvádějí Griffiths a Petrick [8]. Práci algoritmu ukážeme na příkladě a použijeme k tomu nové BK gramatiky $G_3 = (V_T, V_N, R'', S)$, která je až na množinu R'' stejná jako G_1 definovaná v oddíle 2.1.

R'':

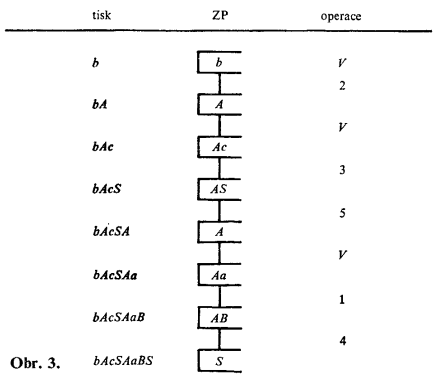
1. $a \rightarrow B$
2. $b \rightarrow A$
3. $c \rightarrow S$
4. $AB \rightarrow S$
5. $AS \rightarrow A$.

Analýza řetězu „*bca*" začíná tím, že do zásobníkové paměti se zapíše první terminální symbol „*b*". V tomto okamžiku lze aplikovat pravidlo 2 a do ZP se zapíše A. Nyní není možno aplikovat žádné z pravidel, proto se do ZP zapíše další terminální symbol „*c*", který se tím stává nejhořejším symbolem v ZP. Je vidět, že nyní lze aplikovat pravidlo 3, takže v ZP dostaneme AS a dále lze použít pravidla 5 atd. Jakmile je zásobníková paměť vyprázdněna a obsahuje jen výsledný symbol S, analýza věty končí a přechází se k další větě. Po každém zapsání vstupního terminálního symbolu do ZP nebo po každém úspěšném nahrazení následuje vždy bezprostřední tisk, takže na výstupu dostáváme FU analyzované věty v bezzávorkové notaci. Na obr. 3 je ukázána analýza řetězu „*bca*", čísla označují aplikovaná pravidla, $\sqsubset$ značí zásobníkovou paměť a V zápis terminálního symbolu do ZP.

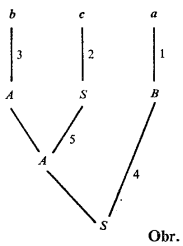
Jako výsledek analýzy vstupního řetězu „*bca*" dostaneme posloupnost symbolů *bAcSAaBC* (obr. 3), která odpovídá převrácenému FU na obr. 3a nebo ohodnocenému uzávorkování v zápisu

$$S[\ A[\ A[b]\ S[c]]\ B[a]]\ .$$

Poněkud odlišná je situace při použití pravidel *T2C*. Aplikování těchto pravidel vede nutně k porušení principu ZP (poslední dovnitř – první ven), ale tento problém je v algoritmu vyřešen poměrně jednoduše zavedením priority (vlastnost dovolující porušit princip ZP) pro vložené symboly, jako je např. symbol *C* na obr. 2. Priorita je vloženým symbolům připsána předem a tyto symboly jsou v paměti počítače uloženy odděleně v samostatném poli *PRIOR*.



Obr. 3.



Obr. 3a.

Stane-li se, že při analýze některé věty chybí v R' potřebné pravidlo (R' samozřejmě v současné době představuje jen fragment toho, co by měla obsahovat úplná gramatika češtiny), počítač v daném místě nahrazení příslušného symbolu neprovede a snaží se analyzovat větu dále. Nepodaří-li se mu větu kompletně analyzovat, tj. nedospěje-li po přečtení všech vstupních symbolů a prohledání celé množiny R' až k symbolu *S*, vytiskne neúplný FU a přejde k analýze další věty. Neúplný FU se od úplného liší tím, že v něm chybí některé vrcholy a hrany, lze však podle něho velmi snadno zjistit, které pravidlo v R' chybí, a je pak možno chybějící pravidla do množiny R' pohodlně doplnit.

4.1. Program analýzy v Algolu*

Vysvětlení k programu: Znaky 999 a 9999 označují konec vět, identifikátor *STACK* označuje zásobníkovou paměť, která je programu definována jako pole o rozsahu 1 : 20, význam identifikátorů *T2B*, *T3B*, *T3A*, *T2A*, *T2C* byl již vysvětlen v oddíle 3.1, identifikátor *PRIOR* označuje pole obsahující symboly s prioritou.

* Děkuji doc. J. Kopřivovi z LPS VUT Brno za převedení programu analýzy do Algolu a B. Kyliánové rovněž z LPS VUT za pomoc při konkretizaci programu. Dále děkuji M. Šimkovi z Kancelářských strojů v Praze, který program ladil.

```
326  begin
       integer i, j, count; integer array T2B [1 : 24], T3B [1 : 14], T3A [1 : 90],
         T2A [1 : 1260], T2C [1 : 51], PRIOR [1 : 10], STACK [1 : 20];
           switch SSS := L3, L5;
               for i := 1 step 1 until 24 do read T2B [i];
               for i := 1 step 1 until 14 do read T3B [i];
               for i := 1 step 1 until 90 do read T3A [i];
               for i := 1 step 1 until 1260 do read T2A [i];
               for i := 1 step 1 until 51 do read T2C [i];
               for i := 1 step 1 until 7 do read PRIOR [i];

     L1: count := 1;
     L2: read STACK [count]; j := 1;
         if STACK [count] = 999 then goto L10 else
           if STACK [count] = 9999 then begin nový řádek; goto L1 end;
         print STACK [count];
     L3: for i := 1 step 2 until 23 do
         if STACK [count] = T2B [i] then
             begin STACK [count] := T2B [i + 1]; print STACK [count]; goto L4
             end;
     L4: for i := 1 step 2 until 13 do
         if STACK [count] = T3B [i] then
             begin STACK [count] := T3B [i + 1]; print STACK [count]; goto L6
             end;
     L6: if count = 1 then
         begin count := count + 1; goto if j = 1 then L2 else L12
         end
           else goto L7;
    L12: j := 1;
     L7: for i := 1 step 3 until 88 do
         if STACK [count − 1] = T3A [i] ∧ STACK [count] = T3A [i + 1] then
          begin STACK [count − 1] := T3A [i + 2]; count := count − 1;
          print STACK [count]; goto SSS [j]
         end;
     for i := 1 step 3 until 1258 do
       if STACK [count − 1] := T2A [i] ∧ STACK [count] = T2A [i + 1] then
        begin STACK [count − 1] := T2A [i + 2]; count := count − 1;
              print STACK [count]; goto SSS [j]
       end;
     if j = 2 then begin count := count + 1; j := 1; goto L7
                   end
                     else goto L11;
     L5: count := count + 1; STACK [count] := STACK [count + 1]; j := 1; goto L7;
    L11: if count <3 then goto L9;
           for i := 1 step 1 until 10 do
             if STACK [count − 1] = PRIOR [i] then goto L8;
                                                  goto L9;
```

```
L8: for i := 1 step 1 until 10 do
      if STACK [count] = PRIOR [i] ∨ STACK [count − 2] = PRIOR [i]
            then goto L9;
    for i := 1 step 3 until 49 do
      if STACK [count − 2] = T2C [i] ∧ STACK [count] = T2C [i + 1] then
      begin STACK [count − 2] := T2C [i + 2]; count := = count − 2;
            print STACK [count]; j := 2; goto L3
      end;
L9: count := count + 1; goto L2;
L10: end programu
```

5. VÝSLEDKY EXPERIMENTU

Vlastní syntaktická analýza 75 českých vět byla provedena na samočinném počítači Elliott 4100 v květnu 1966 během výstavy INCOMEX 66. Uvedených 75 vět počítač analyzoval asi 12–14 min (včetně vstupu a výstupu). Věty byly vybrány převážně z vědecko-technických textů. Bez chyby bylo analyzováno 35 vět, které představovaly výchozí materiál pro sestavení G_2. Ostatních 40 vět bylo analyzováno většinou neúplně, protože v G_2 chyběla příslušná pravidla. Po doplnění chybějících pravidel byl soubor vstupních vět rozšířen na 111 vět a analýza provedena znovu na počítači SAAB D 21 v říjnu 1966*. Z uvedených 111 vět bylo 60 analyzováno správně a 51 neúplně.

Ze získaných výsledků vyplývá, že stupeň empirické adekvátnosti gramatiky G_2 je v současné verzi nízký, protože G_2 je vzhledem k složitosti českého jazykového systému velmi neúplná. V současném stadiu výzkumů však šlo jen o ověření základních lingvistických a programovacích postupů, tj. ukázali jsme, že není příliš obtížné sestavit gramatiku, která popisuje jistou omezenou část syntaktického systému češtiny, a řešili jsme některé problémy týkající se programování lingvistických úloh. Bylo použito jazyka Algol 60, který se ukazuje jako velmi vhodný k programování lingvistických úloh logického charakteru.

V dalším výzkumu půjde nyní o to, aby algoritmus analýzy se skutečně stal nástrojem umožňujícím hlubší a úplnější popis syntaktického systému češtiny na vyšší úrovni. Gramatiku G_2 bude nutno doplnit tak, aby byla dostatečně (i když nepředpokládáme, že zcela) adekvátní. V této souvislosti je třeba upozornit i na to, že je to záležitost pracná a empiricky místy značně nevyjasněná. Při sestavování gramatiky pro češtinu vznikají totiž problémy, které nebyly v dosavadních gramatikách [3], [9], [12] prakticky vůbec řešeny (protože tyto gramatiky jsou určeny lidem, nikoli pro samočinný počítač) a které jsou spojeny s otázkou vztahu sémantiky a syntaxe v přirozeném jazyce. Syntaktické jednotky a vztahy jsou v existujících gramatikách obvykle vymezovány nejen na základě svých formálních vlastností, ale

* Děkuji vedení LPS VUT v Brně a firmě SAAB za poskytnutí strojového času na počítači D 21.

také na základě vlastností sémantických. Zatímco formální vlastnosti syntaktických jednotek a vztahů jsou poměrně snadno dostupné formalizovanému popisu, nelze zdaleka říci totéž o jejich sémantické stránce, jejíž formalizované popisy bude teprve nutno vytvořit.

V současné době se často poukazuje na to, že BK gramatiky samy o sobě nejsou adekvátním modelem pro popis jazykových systémů přirozených jazyků. Jsou proto budovány složitější formální modely, zejména tzv. transformační gramatiky Chomského [4], [6] (dosud prakticky jen pro angličtinu a němčinu), pro něž je však typické, že obsahují jako své základní složky částečně upravené BK gramatiky nebo systémy s nimi slabě ekvivalentní. Tato skutečnost nás podporuje v názoru, že má smysl pracovat na sestavení gramatiky (jako je G_2) popisující syntaktický systém češtiny, která je slabě ekvivalentní s BK gramatikami (a která je pravděpodobně co do silné ekvivalence předčí). „Nespojitost" G_2 je motivována čistě empiricky, soudíme např., že G_2 přiřazuje českým větám s nespojitými složkami adekvátnější strukturní popisy, než by přiřazovala slabě ekvivalentní BK gramatika. Jiné vlastnosti G_2 (sekvenčnost) vyplývají z požadavků kladených strojovým zpracováním (snadné programování apod.). Konečně sestavujeme-li gramatiku pro automatickou analýzu (popř. syntézu) jazyka L (češtiny), není v počátečních fázích práce možné už z technických a fyzických důvodů mít gramatiku popisující přímo jazyk L, vždy je možno začít gramatikou pro $L' \subset L$, která je teprve v průběhu dalšího výzkumu postupně doplňována a zdokonalována.

5.1. Závěry

Získané výsledky naznačují, jakými cestami se bude ubírat další výzkum. Po technické stránce bude třeba upravit program analýzy tak, aby počítač tiskl frázové ukazatele vět přímo v podobě grafů – stromů. Dále počítáme s tím, že program analýzy bude upraven tak, aby bylo možno získávat všechny možné frázové ukazatele dané vstupní věty v případě, že je víceznačná.

Po lingvistické stránce bychom pro G_2 chtěli dosáhnout co nejpřijatelnějšího stupně empirické adekvátnosti, tj. aby G_2 byla schopna postihnout všechny základní české větné (jednoduché i souvětné) typy. Zde se naskýtá možnost použít k doplňování gramatiky G_2 přímo samočinného počítače, tj. upravit program analýzy tak, aby počítač byl schopen ve spolupráci s člověkem převzít většinu práce vznikající při doplňování gramatiky.

(Došlo dne 30. března 1967.)

[1] E. Bach: An Introduction to Transformational Grammars. Holt, Rinehart and Winston Inc., New York 1964, 181—185.
[2] Y. Bar-Hillel: On Formal Properties of Simple Phrase Structure Grammars. Sb. Language and Information, Jerusalem 1964, 116—150.
[3] J. Bauer, M. Grepl: Skladba spisovné češtiny. SPN, Praha 1965 (skriptum).
[4] N. Chomsky: Introduction to the Formal Analysis of Natural Languages, Sb. Handbook of Mathematical Psychology II, editoři Luce, Galanter, Bush, John Wiley, New York 1963, 269—321.
[5] N. Chomsky: Formal Properties of Grammars. Sb. Handbook of Mathematical Psychology II, editoři Luce, Galanter, Bush, John Wiley, New York 1963, 323—418.
[6] N. Chomsky: Aspects of the Theory of Syntax, MIT Press, Cambridge Mass. 1965, 3—62.
[7] S. Ginsburg: Mathematical Theory of Context-free Languages. Materiály z Letní školy programovacích jazyků, Pisa 1965.
[8] T. Griffiths, S. R. Petrick: On the Relative Efficiencies of Context-free Grammars Recognizers. Communications of the ACM *8* (1965), 5, 289—300.
[9] B. Havránek, A. Jedlička: Česká mluvnice. SPN, Praha 1960.
[10] S. Kuno: Computer Analysis of Natural Languages. Proceedings of the Symposium on Mathematical Aspects of Computer Science, April 5—7, New York 1966, 6—21 (v tisku).
[11] P. Novák: Některé otázky syntaktické analýzy (z hlediska SP). Slovo a slovesnost *23* (1962), 1, 9—20.
[12] V. Šmilauer: Novočeská skladba. SPN, Praha 1966.
[13] V. Yngve: A Model and an Hypothesis for Language Structure. Proceedings of the American Philosophical Society *104* (1960), 5, 444—466.
[14] G. H. Mathews: Discontinuity and Asymmetry in Phrase Structure Grammars. Information and Control *6*, 137—146.
[15] P. Novák: Doslov k českému překladu Syntaktických struktur N. Chomského. Academia, Praha 1966, 191—199.
[16] K. Čulík: On Some Transformations in Context-free Grammars and Languages. Czechoslovak Mathematical Journal *17* (*92*) (1967), 278—311.

SUMMARY

Automatic Syntactic Analysis of Czech Text
(An experiment)

KAREL PALA

The aim of our experiments is to conduct an automatic analysis of syntactic structures of Czech sentences chosen from complex scientific texts.

We use the recognition procedure which proceeds from bottom to top. The input string are given: the task of the computer is to decide whether the given string is a grammatical sentence of a particular language (i. e. Czech), and to print its syn-

tactic analysis (phrase marker) on the output. The grammar G_2, employed in this research, is the *sequential discontinuous grammar* and is defined in the following way: $G_2 = (V_T, V_N, R', S)$, where V_T is the set of *terminal symbols* from which the input strings are composed, V_N is the set of *nonterminal symbols*, i. e. the set of grammatical categories, R' is the set of *inverse rewriting rules* in the form $B \rightarrow A$, $CB \rightarrow D$, $E \ldots F \rightarrow H$, S is the set of *designed final symbols*. The above mentioned sets of symbols and rewriting rules constitute a certain part (fragment) of the description of Czech syntax. The algorithm of the analysis makes use of the pushdown store and most probably corresponds to the algorithm of selective direct substitution (SDS) described by Griffiths and Petrick [8]. The input Czech sentences are analyzed during only one scanning.

The analysis of all sentences was performed on the computer ELLIOTT 4100 (75 senteces) in May and June 1966, and on the computer SAAB D 21 (111 sentences) in October 1966. The program of the analysis has been written in ALGOL language.

Further possibilities of research point in the direction of heuristic procedure: we should like to make the computer construct and complete the grammar G_2 either by itself or in combination of man and machine.

In this way we shall be able to arrive at grammar of Czech which has sufficient degree of empirical adequacy. This would help us to pass from the sphere of basic research to an actual description of the syntactic system of the Czech language.

Dr. Karel Pala, katedra českého jazyka, filosofická fakulta UJEP, Arna Nováka 1, Brno.