prace naukowo-badawcze

Instytutu Maszyn Matematycznych

A. ŁUKASIEWICZ: Metoda automatycznej konstrukcji analizatora składniowego dla języków generowanych przez gramatykę afiksową

J. M. KLIMOWICZ: Logical values in fault test generation

Zjednoczenie Przemysłu Automatyki i Aparatury Pomiarowej "MERA"

Instytut Maszyn Matematycznych

prace naukowo-badawcze Instytutu Maszyn Matematycznych

Nr [1]

w zeszycie zamieszczono:

Warszawa 1980

Adres Redakcji: Instytut Maszyn Matematycznych
Branżowy Ośrodek INTE
ul. Krzywickiego 34, 02-078 Warszawa
tel. 28-37-29 lub 21-84-41 w. 244

Prace naukowo-badawcze Instytutu Maszyn Matematycznych

METODA AUTOMATYCZNEJ KONSTRUKCJI ANALIZATORA SKŁADNIOWEGO DLA JĘZYKÓW GENEROWANYCH PRZEZ GRAMATYKĘ AFIKSOWĄ

Andrzej ŁUKASIEWICZ

Praca doktorska wykonana
pod kierunkiem prof.dr hab.
Władysława M. Turskiego

Wprowadzone w 1970 r. przez C.H.A. Kostera gramatyki afiksowe są wygodnym aparatem do definiowania języków programowania. Ze względu na dużą moc gramatyk afiksowych /w ogólności można za ich pomocą wygenerować dowolny język rekurencyjnie przeliczalny/ stosowane są one do opisu nie tylko składni bezkontekstowej, ale również i semantyki definiowanego języka. W związku z dużą atrakcyjnością gramatyk afiksowych jako narzędzia do opisu języków, w niniejszej pracy podjęto próbę podania metody automatycznej konstrukcji analizatora składniowego dla języków generowanych przez gramatyki afiksowe. Metoda ta dotyczy szerokiej podklasy klasy gramatyk afiksowych, tzw."dobrze zdefiniowanych gramatyk afiksowych", /wcześniej proponowane metody Kostera i Watta obejmowały stosunkowo wąskie podklasy/. Podano algorytm konstrukcji analizatora składniowego dla danej /dowolnej/ gramatyki dobrze zdefiniowanej. Algorytm ten opiera się na wprowadzonym w pracy pojęciu "dodawania form zdaniowych". Aby ułatwić czytelnikowi zapoznanie się z proponowaną metodą, pracę podzielono na trzy części: przedstawienie metody, dowody podawanych twierdzeń oraz przykłady ilustrujące ważniejsze z wprowadzonych pojęć.

SPIS TREŚCI

WSTĘP

Od kilkunastu lat składnię języków programowania opisuje się za pomocą gramatyk bezkontekstowych. Wiadomo jednak, że gramatyki te są zbyt słabym narzędziem opisu: definiują one bowiem nie sam język, lecz pewien jego nadzbiór. Wynika to stąd, że większość języków programowania wykracza poza klasę języków bezkontekstowych. Dlatego też opis składni języka programowania uzupełnia się zestawem (na ogół nieformalnych) reguł, mówiących o tym, które słowa z nadzbioru wyznaczonego składnią bezkontekstową należą do języka, a które nie należą (por. na przykład opis języka ALGOL 60). Taka sytuacja nie stwarza problemów użytkownikom języka programowania, natomiast wyklucza bezpośrednie użycie opisu języka do automatycznej konstrukcji jego analizatora syntaktycznego. Stąd też zaczęto prowadzić prace nad stworzeniem środków opisowych, które umożliwiałyby opis składni języków szerszej klasy niż bezkontekstowa. Pojawiły się gramatyki atrybutowe (Knuth [11]).

Jedną z propozycji były wprowadzone przez van Vijngaardena gramatyki dwupoziomowe (por. opis języka ALGOL 68 (van Vijngaarden [15])). Jak udowodnił Sintzoff /Sintzoff [1]/, gramatyki te generują wszystkie języki rekurencyjnie przeliczalne, są więc dostatecznie "mocnym" aparatem do opisu składni języków programowania. Jednakże konstrukcja analizatora syntaktycznego (parsera) jest, dla języka zadanego za pomocą gramatyki dwupoziomowej, zagadnieniem trudnym i mało zbadanym (jedyną szerzej dostępną publikacją na ten temat jest praca M.Grzymkowskiego (Grzymkowski [7])). Dlatego też gramatyki dwupoziomowe, chociaż stosunkowo wygodne dla użytkownika języka, są raczej mało przydatne dla osób piszących jego translator.

W celu ułatwienia konstrukcji parsera C.H.A. Koster zaproponował pewną modyfikację gramatyk dwupoziomowych - tzw. gramatyki afiksowe (Koster [9]). Gramatyki te mają moc równą gramatykom van Vijngaardena (tzn. generują wszystkie języki rekurencyjnie przeliczalne), a przy tym algorytmy konstrukcji parsera można oprzeć na znanych metodach analizy składniowej dla gramatyk bezkontekstowych.

Problem automatycznej generacji parsera dla języków opisanych za pomocą gramatyki afiksowej doczekał się, jak dotychczas, dwóch konkretnych rozwiązań.

Autor pierwszego z nich - C.Koster - wyodrębnia w klasie gramatyk afiksowych pewną podklasę, dla której podaje metodę automatycznej konstrukcji parsera /Koster [9]/. Podklasa ta, aczkolwiek dostatecznie szeroka, aby generować wszystkie języki ogólnie rekurencyjne, nastręcza jednak (przez określające ją ograniczenia nałożone na gramatyki afiksowe) duże trudności dla posługującego się tą metodą definiowania języków.

Inny sposób automatycznej generacji parsera opisany został przez D.Watta (Watt [16] i [17], implementacja - Franzen [5] i [6]). Wprowadzone przez Watta "dobrze sformułowane gramatyki afiksowe typu LR(k)" /well formed affix grammars AF-LR (k)/ są już znacznie wygodniejszym, w porównaniu z gramatykami używanymi przez C.Kostera, sposobem zadawania języków programowania.

Jednakże obydwa proponowane rozwiązania posiadają dwie zasadnicze wady:

(1) Parser jest generowany tylko dla języków, które się dają tłumaczyć bezpośrednio od lewej do prawej (translator jest jednoprzebiegowy). Tylko takie języki można bowiem opisać zarówno za pomocą gramatyk stosowanych przez Watta jak i za pomocą gramatyk stosowanych przez Kostera.

(2) Druga wada jest związana z charakterem gramatyk afiksowych w ogóle. Istotną cechą gramatyk afiksowych jest m.in. fakt, że definiujący język musi sam określać każdy tzw. typ afikspozycji (typ afikspozycji jest pojęciem intuicyjnie bliskim Knuthowskiemu typowi atrybutu - tzn. informacji czy dany atrybut jest dziedziczony czy syntetyzowany).
Zmusza to opisującego składnię języka do przybliżonego przynajmniej wyobrażenia sobie jak będzie wyglądał translator tego języka.

Celem niniejszej pracy jest wyeliminowanie powyższych wad. Praca została podzielona na trzy części.

W części pierwszej wprowadzono tzw. "dobrze zdefiniowane gramatyki afiksowe". Nie mają one wady (1) , a gramatyki używane przez Watta zawierają jako swoją właściwą podklasę. Następnie podano metodę automatycznej konstrukcji parsera dla języków generowanych przez dobrze zdefiniowane gramatyki afiksowe oraz metodę zastosowania tzw. zmiennych globalnych wzorowanych na Knuthowskich atrybutach globalnych (Knuth [11]).

W części drugiej zostały zdefiniowane "niezorientowane gramatyki afiksowe". Gramatyki te są pewną modyfikacją gramatyk afiksowych, nie posiadającą wady (2) . Dla języków generowanych przez pewną podklasę niezorientowanych gramatyk afiksowych (podklasa ta nie posiada również wady (1)) podano algorytm automatycznej konstrukcji parsera (okazuje się jednak, że usunięcie wady (2) wpływa niekorzystnie na czas pracy parsera, dlatego też pozostawiono definiującemu język możliwość korzystania z gramatyk, które nie posiadają tylko wady (1)).

Na zakończenie (część trzecia) opisana została metoda automatycznej konstrukcji parsera dla języków definiowanych za pomocą, wprowadzonych przez Watta (Watt [16]), "rozszerzonych gramatyk afiksowych".

Nie wdając się w tej chwili w szczegóły, powiemy tylko, że różnica między gramatykami afiksowymi a rozszerzonymi gramatykami afiksowymi sprowadza się do tego, że funkcje związane z predykatami (występujące w gramatykach afiksowych) zostały opisane (w rozszerzonych gramatykach afiksowych) za pomocą tzw. hiperprodukcji predykatowych. Różnica ta nie ma żadnego związku z omówionymi wcześniej wadami rozwiązań Kostera i Watta dla gramatyk afiksowych, i dlatego podane wyżej uwagi zachowują swoją moc również dla rozszerzonych gramatyk afiksowych.

Ponieważ z punktu widzenia konstrukcji parsera, jedynym następstwem różnicy między gramatykami afiksowymi a rozszerzonymi gramatykami afiksowymi jest inna realizacja (używanej w parserze) procedury PREDYKAT, więc opis parsera dla rozszerzonych gramatyk afiksowych sprowadza się w niniejszej pracy do opisu procedury PREDYKAT.
Dodamy jeszcze, że na ogół do opisu języków nie używa się gramatyk afiksowych, lecz rozszerzonych gramatyk afiksowych. W związku z tym do realizacji procedury PREDYKAT dla gramatyk afiksowych (dotyczy to również niezorientowanych gramatyk afiksowych) nie przykładano w pracy specjalnej wagi.

Praca zawiera trzy dodatki. W celu zwiększenia przejrzystości wszystkie przykłady zgrupowano w dodatku A, a wszystkie dowody twierdzeń /oraz definicje pojęć pomocniczych/ w dodatku B. Natomiast w dodatku C podano sposób w jaki będą reprezentowane w maszynie tzw. wartości afikspozycji (jest to pewne pojęcie charakterystyczne dla gramatyk afiksowych) oraz związane z nimi tzw. suma form zdaniowych.

Praca ma charakter teoretyczny, jednak starano się wszystkie proponowane tutaj rozwiązania podawać w formie umożliwiającej łatwe ich zaprogramowanie. Zakłada się znajomość teorii gramatyk bezkontekstowych, ze szczególnym uwzględnieniem gramatyk typu LR (k) . Przyjęto, że metoda konstrukcji parsera dla gramatyk typu LR (k) jest znana i w związku z tym nie ma potrzeby omawiać jej w tej pracy.

II. OZNACZENIA I POJĘCIA WPROWADZAJĄCE

1. Symbolem ϕ oznaczać będziemy zbiór pusty.
2. Symbolem ε oznaczać będziemy napis pusty.
3. Jeżeli A i B oznaczają dowolne zbiory symboli to przez $A \cdot B$ lub AB będziemy oznaczać konkatenację tych zbiorów.
   Napis A^1 oznaczać będzie zbiór A
   Napis A^n oznaczać będzie zbiór $A\,A^{n-1}$ (gdzie $n \geq 2$)
   Napis A^+ oznaczać będzie zbiór $\bigcup_{n=1}^{\infty} A$
   Napis A^* oznaczać będzie zbiór $A^+ \cup \{\varepsilon\}$
4. Niech V będzie dowolnym zbiorem symboli. Przyjmijmy, że napis x jest elementem zbioru V^+. Mówiąc, że $y \in V^+$ występuje w x (lub x zawiera y) będziemy mieli na myśli, że istnieją $\alpha, \beta \in V^*$ takie, iż $x = \alpha y \beta$. Napis $\alpha y \beta$ będziemy często nazywali wystąpieniem y w x.
5. Przez gramatykę bezkontekstową będziemy rozumieć w tej pracy czwórkę uporządkowaną. Pierwszy element tej czwórki będzie zawsze oznaczał zbiór symboli nieterminalnych, drugi - zbiór symboli terminalnych, trzeci - zbiór produkcji, a czwarty - aksjomat.
6. Załóżmy, że $G = (V_N, V_T, P, S)$ jest dowolną gramatyką bezkontekstową. Niech $M = \alpha A \beta$, $N = \alpha x \beta$ (gdzie $\alpha, \beta, x \in (V_N \cup V_T)^*$; $A \in V_N$). Wtedy:
   (1) Jeżeli $A \to x \in P$ to napiszemy $M \underset{G}{\Rightarrow} N$ lub $M \underset{P}{\Rightarrow} N$
   (2) Jeżeli $\beta \in V_T^*$ i $M \underset{G}{\Rightarrow} N$ (lub $M \underset{P}{\Rightarrow} N$) to napiszemy
   $M \underset{G}{\Rightarrow}_R N$ (lub $M \underset{P}{\Rightarrow}_R N$).
   Przypuśćmy teraz, że M i N są dowolnymi elementami zbioru $(V_N \cup V_T)^*$. Powiemy, że M wyprowadza N (M wyprowadza prawostronnie N) i napiszemy $M \overset{*}{\underset{G}{\Rightarrow}} N$ lub $M \overset{*}{\underset{P}{\Rightarrow}} N$ ($M \overset{*}{\underset{G}{\Rightarrow}}_R N$ lub $M \overset{*}{\underset{P}{\Rightarrow}}_R N$) jeżeli:
   Istnieje ciąg $M_0, M_1, \ldots, M_n$ (n może być zero) o własnościach:
   (a) $M_0 = M$, $M_n = N$
   (b) dla każdego $i = 1, 2, \ldots, n$
   $M_{i-1} \underset{G}{\Rightarrow} M_i$ ($M_{i-1} \underset{G}{\Rightarrow}_R N$)
   Jeżeli $n > 0$ to będziemy czasem pisać $M \overset{+}{\underset{G}{\Rightarrow}} N$ ($M \overset{+}{\underset{P}{\Rightarrow}} N$)
7. Przez maszynę Turinga rozumieć będziemy pojęcie zdefiniowane w pracy J.Hoperofta (Hoperoft [8]).
8. W niniejszej pracy będziemy używać operację := w ogólnie przyjętym znaczeniu (por. ALGOL 60), a więc zwrot x := x+1 oznacza: do dotychczasowej wartości x dodaj 1 - wynik dodawania będzie nową wartością x.

Część pierwsza - GRAMATYKI AFIKSOWE

Rozdział 1. DEFINICJA

Definicja 1.1.

Gramatyką afiksową nazwiemy układ:

$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$, gdzie:

V_N jest skończonym alfabetem symboli nieterminalnych,

V_T jest skończonym alfabetem symboli terminalnych,

A_N jest skończonym alfabetem afiksów nieterminalnych,

A_T jest skończonym alfabetem afiksów terminalnych,

Q jest skończonym zbiorem symboli nazywanych predykatami,

R jest skończonym zbiorem produkcji afiksowych; $R \subset A_N \times (A_N \cup A_T)^*$

B jest skończonym zbiorem symboli nazywanych zmiennymi afiksowymi $B \cap A_T = \phi$.
$\mathrm{card}(B) \gg \mathrm{card}(A_N)$.

D jest funkcją ze zbioru $(B \cup A_T)^*$ na zbiór $(A_N \cup A_T)^*$ spełniającą warunki:
(i) $D(ax) = D(a)\,D(x)$ dla każdego $a, x \in (B \cup A_T)$
(ii) $D(\varepsilon) = \varepsilon$
(iii) dla każdego $x \in A_T^*$: $D(x) = x$
(iv) dla każdego $x \in B$: $D(x) \in A_N$,

S jest zbiorem piątek takim, że dla każdego $x \in V_N \cup Q$ istnieje dokładnie jedna piątka:
$S_X = (X, N_X, A_N^{N_X}, T^{N_X}, F_X)$
należąca do S, przy czym:

- N_X jest liczbą naturalną, nazywaną liczbą afikspozycji elementu x,
- $A_N^{N_X}$ jest N_X-elementowym ciągiem o wyrazach ze zbioru A_N, i-ty wyraz tego ciągu będziemy nazywać dziedziną i-tej afikspozycji symbolu nieterminalnego bądź predykatu X,
- T^{N_X} jest N_X-elementowym ciągiem o wyrazach ze zbioru $\{\tau, \sigma\}$.
Jeżeli i-ty element tego ciągu ma wartość τ to będziemy mówili, że i-ta afikspozycja X jest dziedziczona, w przeciwnym razie będziemy mówili, że jest syntetyzowana.
Jeżeli $a \in A_N$ i przez L_a oznaczamy język generowany przez gramatykę bezkontekstową $G_a = (A_N, A_T, R, a)$, to
- F_X oznacza funkcję rekurencyjnie przeliczalną związaną z symbolem X:
$F_X : L_{a_1} \times L_{a_2} \times \ldots \times L_{a_{N_X}} \longrightarrow \{\text{prawda}, \text{fałsz}\}$, gdzie
a_i $(i = 1, 2, \ldots, N_X)$ jest dziedziną i-tej afikspozycji symbolu X.
Funkcję F_X określać będziemy jedynie wtedy gdy $X \in Q$.

Zbiór S nazwiemy sterowaniem gramatyki G.

Zanim przejdziemy do definicji zbioru P zdefiniujemy pewne pojęcie pomocnicze:
Napis $X(f_1, f_2, \ldots, f_{N_X})$ będziemy nazywać hiperpojęciem jeżeli $X \in Q \cup V_N$ oraz dla każdego $i = 1, 2, \ldots, N_X$ ma miejsce:
(1) $f_i \in (B \cup A_T)^*$,
(2) $A_i \overset{*}{\underset{R}{\Longrightarrow}} D(f_i)$; gdzie A_i jest dziedziną i-tej afikspozycji x.
Jeżeli $X \in V_N$ to będziemy mówili o hiperpojęciu nieterminalnym, jeżeli $X \in Q$ - o hiperpojęciu predykatowym.

P jest skończonym zbiorem hiperprodukcji. Każda hiperprodukcja ma postać:

$$Z \to Z_1 Z_2 \ldots Z_n,$$

gdzie Z jest dowolnym hiperpojęciem nieterminalnym, Z_j $(j=1,2,\ldots,n)$ jest albo napisem terminalnym (tzn. $Z_j \in V_T^*$), albo dowolnym hiperpojęciem, $\sigma \in V_N$ jest aksjomatem gramatyki G.

Definicja 1.2.

Powiemy, że para $Y \to Y_1 Y_2 \ldots Y_n$ jest produkcją gramatyki afiksowej

$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ jeżeli:

albo (1) Istnieje hiperprodukcja $Z \to Z_1 Z_2 \ldots Z_n \in P$ taka, że para $Y \to Y_1 Y_2 \ldots Y_n$ powstała z tej hiperprodukcji w wyniku następującego postępowania: każde wystąpienie dowolnej zmiennej afiksowej b występującej w hiperprodukcji $Z \to Z_1 Z_2 \ldots Z_n$ zastępujemy dowolnym napisem $c \in A_T^*$ takim, że $D(b) \overset{*}{\underset{R}{\Rightarrow}} c$ (wszystkie wystąpienia jednej zmiennej zastępujemy tym samym napisem). Czynność tę wykonujemy dla każdej zmiennej afiksowej występującej w omawianej hiperprodukcji,

albo (2) Para $Y \to Y_1 Y_2 \ldots Y_n$ jest postaci $y\,(f_1, f_2, \ldots, f_{N_y}) \to \varepsilon$, gdzie $y \in Q$; $f_1, f_2, \ldots, f_{N_y} \in A_T^*$ oraz $F_y(f_1, f_2, \ldots, f_{N_y}) =$ prawda.

Definicja 1.3.

Napiszemy $\alpha Y \beta \overset{}{\underset{G}{\Rightarrow}} \alpha Y_1 Y_2 \ldots Y_n \beta$, (gdzie $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ jest gramatyką afiksową) jeżeli $Y \to Y_1 Y_2 \ldots Y_n$ jest produkcją gramatyki G.

Powiemy, że X wyprowadza Y w gramatyce afiksowej G i napiszemy $X \overset{*}{\underset{G}{\Rightarrow}} Y$ jeżeli istnieje ciąg $X_0, X_1, X_2, \ldots, X_k$ taki, że:

(1) $X_0 = X$

(2) $X_n = Y$

(3) dla każdego $i=1,2,\ldots,k$ jest: $X_{i-1} \overset{}{\underset{G}{\Rightarrow}} X_i$

Definicja 1.4.

Hiperpojęciem początkowym nazwiemy każde hiperpojęcie $\sigma(F_1, F_2, \ldots, F_{N_\sigma})$ takie, że σ jest aksjomatem gramatyki G, a $F_1, F_2, \ldots, F_{N_\sigma}$ są elementami zbioru A_T^*.

Definicja 1.5.

Niech $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ będzie dowolną gramatyką afiksową, wtedy zbiór:

$L(G) = \{ y \in V_T^* \mid$ istnieje hiperpojęcie początkowe X takie, że $X \overset{*}{\underset{G}{\Rightarrow}} y \}$

nazwiemy językiem generowanym przez gramatykę G.

Definicje 1.1. - 1.5 ilustruje przykład 1.1

Prawdziwe jest twierdzenie:

Twierdzenie 1.1.

Dla każdego języka rekurencyjnie przeliczalnego istnieje gramatyka afiksowa, która go generuje.

Podana tutaj definicja gramatyki afiksowej (definicja 1.1) różni się nieco od definicji zaproponowanej przez C.H.A. Kostera (Koster [9]). W oryginalnej definicji Kostera nie występował zbiór B i funkcja D oraz trochę inaczej zdefiniowane było hiperpojęcie. W pracy Kostera [9] zakładano ponadto, że liczba afikspozycji aksjomatu gramatyki jest równa zero. Wprowadzone zmiany są nieistotne, wydają się natomiast wpływać na zwiększenie przejrzystości niniejszej pracy.

Rozdział 2. DOBRZE ZDEFINIOWANE GRAMATYKI AFIKSOWE

Nie potrafimy dla dowolnej gramatyki afiksowej podać metody automatycznej konstrukcji parsera. Umiemy to zrobić dla pewnej podklasy gramatyk afiksowych. Podklasę tę określimy nakładając ograniczenia na gramatyki afiksowe. Będą one głównie zmierzały do "zdeterminizowania" wyprowadzeń.

2.1. OGRANICZENIA DOTYCZĄCE PRODUKCJI AFIKSOWYCH

Niech $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ będzie gramatyką afiksową oraz niech a będzie dowolnym elementem zbioru A_N. Przez G_a oznaczać będziemy gramatykę bezkontekstową $G_a = (A_N, A_T, R, a)$. Jak zobaczymy w dalszym ciągu niniejszej pracy, konstruowany przez nas parser będzie znajdował wywody w gramatykach G_a. W związku z wyborem metody LR(k) przyjmiemy definicję:

Definicja 2.1.1

Powiemy, że produkcje afiksowe R gramatyki $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ są dobrze zdefiniowane jeżeli dla każdego $a \in A_N$ istnieje k takie, że gramatyka G_a jest typu LR(k).

2.2. OGRANICZENIA DOTYCZĄCE HIPERPRODUKCJI

Przyjmiemy, że $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ jest dowolną gramatyką afiksową. Ze zbioru hiperprodukcji P utworzymy zbiór Z pewnych produkcji bezkontekstowych:
Jeżeli funkcja $f: (V_N \cup V_T \cup Q \cup B \cup A_T \cup \{(,),\rightarrow\})^* \rightarrow (V_N \cup V_T \cup \{\rightarrow\})^*$ jest homomorfizmem postaci:

$$f(a) = \begin{cases} a & \text{gdy } a \in V_N \cup V_T \cup \{\rightarrow\} \\ \varepsilon & \text{gdy } a \notin V_N \cup V_T \cup \{\rightarrow\} \end{cases}$$

to zbiór $Z = f(P)$ nazwiemy zbiorem produkcji podstawowych gramatyki G.

Gramatykę bezkontekstową $G_P = (V_N, V_T, Z, \sigma)$ będziemy nazywać gramatyką podstawową gramatyki G.

Podobnie jest dla gramatyk G_a wymagamy żeby gramatyka G_P była gramatyką typu LR(k). Przyjmiemy więc definicję:

Definicja 2.2.1

Powiemy, że gramatyka podstawowa G_p gramatyki G jest dobrze zdefiniowana jeżeli istnieje takie k, że G_p jest gramatyką typu LR (k).
Definicję 2.2.1 ilustruje przykład 2.

Definicja 2.2.2

Powiemy, że zmienna afiksowa b ma w hiperprodukcji p wystąpienie definiujące, jeżeli występuje ona na pozycji dziedziczonej po lewej stronie tej hiperprodukcji, bądź na pozycji syntezyzowanej po prawej stronie tej hiperprodukcji.

Definicja 2.2.3

Załóżmy, że p jest dowolną hiperprodukcją gramatyki afiksowej G. Powiemy, że hiperprodukcja p jest dobrze zdefiniowana, jeżeli wszystkie zmienne afiksowe występujące w hiperprodukcji p mają w niej wystąpienia definiujące.

Zbiór hiperprodukcji gramatyki G nazwiemy dobrze zdefiniowanym, jeżeli każda hiperprodukcja tej gramatyki jest dobrze zdefiniowana.

Zbiór hiperprodukcji gramatyki z przykładu 1 jest dobrze zdefiniowany.

Definicje 2.2.2 i 2.2.3 oraz podane niżej definicje 2.3.1, 2.3.2 i 2.3.3 pomogą nam w intuicyjnej interpretacji pojęć "afikspozycja dziedziczona" i "afikspozycja syntetyzowana".

2.3. OGRANICZENIA DOTYCZĄCE TYPÓW AFIKSPOZYCJI

Definicja 2.3.1

Niech hiperpojęcia $Z_1 = X_1 (F_1, F_2, \ldots, F_{N_{X_1}})$.
$Z_2 = X_2 (G_1, G_2, \ldots, G_{NX_2})$ występują w hiperprodukcji p gramatyki afiksowej
$G = (V_N, V_T, A_2 A_T, Q, R, B, D, S, P, \delta)$.
Powiemy, że i-ta afikspozycja X_1 blokuje j-tą afikspozycję X_2 w hiperprodukcji p, jeżeli istnieją $\alpha, \beta, \gamma, \delta \in (B \cup A_T)^*$ i $L \in B$ takie, że $F_i = \alpha L \beta$, $G_j = \gamma L \delta$ oraz zachodzi jeden z przypadków:

(1) Z_1 i Z_2 oznaczają to samo hiperpojęcie występujące po lewej stronie hiperprodukcji p i i-ta afikspozycja X_1 jest dziedziczona zaś j-ta afikspozycja X_2 ($X_2 = X_1$) jest syntetyzowana.

(2) Z_1 występuje po lewej stronie hiperprodukcji p, Z_2 po prawej oraz zarówno i-ta afikspozycja X_1 jak i j-ta afikspozycja X_2 są dziedziczone.

(3) Z_1 występuje po prawej stronie p, Z_2 po lewej stronie oraz zarówno i-ta afikspozycja X_1 jak i j-ta afikspozycja X_2 są syntetyzowane.

(4) Z_1 i Z_2 występują po prawej stronie hiperprodukcji p oraz i-ta afikspozycja X_1 jest syntetyzowana, j-ta afikspozycja X_2 jest dziedziczona.

Oprócz wymienionych wyżej przypadków powiemy, że i-ta afikspozycja X_1 blokuje j-tą afikspozycję X_2 w hiperprodukcji p, zawsze wtedy, gdy X_1 i X_2 oznaczają ten sam predykat oraz i-ta afikspozycja X_1 jest dziedziczona, j-ta afikspozycja X_2 ($X_2=X_1$) jest syntetyzowana.

Definicja 2.3.2

Załóżmy, że $\alpha_0, \alpha_1, \ldots, \alpha_n \in V_T^*$ oraz, że hiperprodukcja h gramatyki $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ jest postaci:

$$X_0(F_1^0, \ldots, F_{N_{X0}}^0)\alpha_0 X_1(F_1^1, \ldots, F_{N_{X1}}^1)\alpha_1 X_2(F_1^2, \ldots, F_{N_{X2}}^2)\alpha_2 \ldots X_n(F_1^n, \ldots, F_{N_{X_n}}^n)\alpha_n$$

Wtedy grafem hiperprodukcji h nazwiemy każdy z grafów zorientowanych D_h o własnościach:

(1) Wierzchołkami (węzłami) grafu D_h są trójki:

$(X_0, k_0, 1), (X_0, k_0, 2), \ldots, (X_0, k_0, N_{X0})$, $(X_1, k_1, 1)$, $(X_1, k_1, 2), \ldots, (X_1, k_1, N_{X1}) \ldots$
$\ldots (X_n, k_n, 1), (X_n, k_n, 2), \ldots, (X_n, k_n, N_{X_n})$

gdzie $k_0, k_1, \ldots, k_n$ są dowolnymi (ale ustalonymi dla danego grafu) liczbami naturalnymi takimi, że jeżeli $i \neq j$ to $k_i \neq k_j$.

(2) Od węzła (X_i, k_i, j) do węzła (X_l, k_l, m) prowadzi strzałka wtedy i tylko wtedy, gdy j-ta afikspozycja X_i blokuje w hiperprodukcji p m-tą afikspozycję X_l.

Definicję tę ilustruje przykład 3.

Ustalmy we wszystkich następnych rozważaniach dowolną gramatykę afiksową $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ taką, że jej gramatyka podstawowa G_p jest dobrze zdefiniowana. Zwróćmy uwagę, że przy takim ustaleniu każdej produkcji podstawowej odpowiada dokładnie jedna hiperprodukcja. Korzystając z tego faktu poniższa definicja określa nieformalnie pojęcie zorientowanego grafu wywodu $\sigma \overset{*}{\underset{R}{\Rightarrow}} x$ słowa x w gramatyce G_p.

Formalną definicję podajemy w dodatku B.

Definicję grafu wywodu ilustruje przykład 4.

Definicja 2.3.3

Przyjmijmy, że wywodowi prawostronnemu $\sigma \overset{*}{\underset{G_p R}{\Rightarrow}} M$ formy zdaniowej M odpowiada ciąg $p_1, p_2, \ldots, p_k$ kolejnych produkcji podstawowych zastosowanych w tym wywodzie.

Grafem wywodu $\sigma \overset{*}{\underset{G_p R}{\Rightarrow}} M$ nazwiemy każdy z grafów zorientowanych powstałych ze złożenia grafów $D_{h_1}, D_{h_2}, \ldots, D_{h_k}$ hiperprodukcji $h_1, h_2, \ldots, h_k$ odpowiadających produkcjom $p_1, p_2, \ldots, p_k$.

Pojęcia "złożenie grafów" nie będziemy tutaj formalnie definiować. Wydaje się ono jednak (w odniesieniu do wywodu) intuicyjnie zrozumiałe.

Ze względu na przyjęte założenie jednoznaczności gramatyki G_p wywód prawostronny w gramatyce G_p jest określony jednoznacznie.

Opierając się na pojęciu grafu wywodu przyjmiemy następującą definicję:

Definicja 2.3.4

Powiemy, że gramatyka afiksowa G o dobrze zdefiniowanej gramatyce podstawowej G_p nie posiada pętli, jeżeli żaden z grafów wywodów $\sigma \overset{*}{\underset{G_p R}{\Rightarrow}} x$ żadnego słowa $x \in L(G_p)$ nie posiada pętli.

Problem czy dana gramatyka afiksowa posiada pętlę jest problemem rozstrzygalnym. W dodatku B cz.I rozdz.2 znajduje się algorytm wzorowany na algorytmie D.Knutha (Knuth [10]) wykrywający ewentualną pętlę.

2.4. OGRANICZENIA DOTYCZĄCE PREDYKATÓW

Jak się przekonamy w dalszym ciągu pracy, znajdowanie wyprowadzeń w gramatyce afiksowej będzie się wiązało m.in. z obliczaniem wartości funkcji F_q związanych z poszczególnymi predykatami $q \in Q$. Aby proces ten był zawsze procesem skończonym musimy przyjąć następującą definicję:

Definicja 2.4.1

Powiemy, że zbiór predykatów Q gramatyki afiksowej G jest dobrze zdefiniowany, jeżeli dla każdego predykatu $q \in Q$, związana z nim funkcja F_q jest funkcją ogólnie rekurencyjną.

2.5. DEFINICJA

Gramatykę afiksową postaci:

$$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \delta)$$

nazwiemy dobrze zdefiniowaną, jeżeli spełnia ona warunki:

(1) Produkcje afiksowe R są dobrze zdefiniowane,

(2) Gramatyka podstawowa G_p gramatyki G jest dobrze zdefiniowana,

(3) Zbiór hiperprodukcji P jest dobrze zdefiniowany,

(4) Gramatyka G nie posiada pętli,

(5) Zbiór predykatów jest dobrze zdefiniowany.

Zanim przejdziemy do podania metody konstrukcji parsera dla dobrze zdefiniowanych gramatyk afiksowych, omówimy pokrótce sens wprowadzonych powyżej ograniczeń nałożonych na gramatyki afiksowe - ograniczeń definiujących klasę gramatyk afiksowych, dla których konstrukcja parsera za pomocą proponowanej metody jest możliwa.

(1) Dla każdego $a \in A_N$ gramatyki G_a muszą być typu LR(k) /patrz definicja 2.1.1/. Warunek ten nie jest warunkiem niezbędnym. Wynika on z potrzeby /w trakcie konstruowania parsera/ znajdowania wywodów w gramatykach G_a, przy czym dla samej metody znajdowania wyprowadzeń w gramatyce afiksowej, sposób w jaki konstruowane są wywody w gramatykach G_a jest zupełnie obojętny. Gramatyki te mogłyby więc być również np. typu LL(k), gramatykami z pierwszeństwem, lub innymi dowolnymi gramatykami bezkontekstowymi /w szczególności na gramatyki G_a mogłyby nie być nałożone żadne ograniczenia/. W pracy zdecydowano się na wybór metody LR(k), stąd warunek (1).

(2) Warunek ten /def. 2.2.1/ ma te same przyczyny jak warunek (1), a więc wszystkie uwagi podane wyżej zachowują również i tutaj swoją moc.

(3) Wystąpienie definiujące zmiennej afiksowej to /intuicyjnie mówiąc/ takie wystąpienie tej zmiennej, którego wartość jest podstawiana na inne wystąpienia danej zmiennej w tej samej hiperprodukcji. Gdyby więc jakaś zmienna nie miała żadnego wystąpienia definiującego, to mogłoby się zdażyć, że w tej hiperprodukcji owa zmienna miałaby nieokreśloną wartość dla jakiegoś wyprowadzenia.

(4) W trakcie konstruowania wywodu w gramatyce afiksowej, wartości poszczególnych afikspozycji są przekazywane wg "dróg" określonych przez graf tego wywodu. Pętla jednej z takich dróg oznaczałaby, że pewna afikspozycja będzie miała nieobliczoną wartość.

(5) Warunek ten ma na celu zagwarantowanie aby programy odpowiadające obliczaniu wartości funkcji związanych z predykatami były zawsze programami skończonymi.

Właściwość dobrej definiowalności dla dowolnej gramatyki afiksowej jest właściwością nierozstrzygalną. Wynika to z faktu, że dla niczym nie ograniczonej gramatyki bezkontekstowej G_{BK} nierozstrzygalny jest problem istnienia k takiego, że G_{BK} jest typu LR (k). Nierozstrzygalne są więc punkty (1) i (2) z wyżej podanych. Nierozstrzygalny jest również warunek (5). Jednak w praktycznej realizacji jest on na ogół łatwy do sprawdzenia.

Rozdział 3. PARSER

W rozdziale tym podamy metodę automatycznej konstrukcji parsera dla dowolnej dobrze zdefiniowanej gramatyki afiksowej:

$$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$$

Parser ten dla danych: gramatyki G i słowa $x \in V_T^*$, podaje wywód x w gramatyce G - jeżeli $x \in L(G)$, bądź też powoduje sygnalizację błędu - jeżeli $x \notin L(G)$.

Parser podzielimy na dwie części:

Pierwsza część - zwana konstruktorem - mając na wejściu gramatykę G podaje na wyjściu gramatykę G', będącą zmodyfikowaną postacią gramatyki G. Konstruktor jest oczywiście uruchamiany jeden raz dla danej gramatyki.

Druga część-zwana parserem właściwym - korzysta z wyników konstruktora i dla danego $x \in V_T^*$ znajduje wywód x w gramatyce G - gdy $x \in L(G)$, albo sygnalizuje błąd - gdy $x \notin L(G)$.

Czynności wykonywane przez parser właściwy będą realizowane w dwóch kolejno uruchamianych przebiegach. Mówiąc nieformalnie - przebieg pierwszy odpowiada analizie składniowej wraz z pewnymi początkowymi "akcjami semantycznymi", drugi - analizie semantycznej.

We wszystkich poniższych rozważaniach ustaliliśmy dobrze zdefiniowaną gramatykę afiksową: $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$

Punkty 3.2.1, 3.2.2 i 3.2.5 są ilustrowane przykładami - odpowiednio - 5, 6 i 7.

3.1. KONSTRUKTOR

Konstruktor ma do spełnienia dwa zadania:

(1) Jak wiemy, jeżeli zmienna afiksowa występuje w dowolnej hiperprodukcji h, to musi mieć w niej wystąpienie definiujące. Z różnych względów wygodnie jest aby zmienna nie mająca w hiperprodukcji h wystąpień innych niż definiujące, miała w h dokładnie jedno wystąpienie.

Załóżmy więc, że w hiperprodukcji h występuje zmienna L, która ma w niej kilka wystąpień definiujących, a nie ma wystąpień stosowanych (tzn. takich, które nie są definiujące). Konstruktor wykonuje wtedy dla każdej takiej hiperprodukcji h i zmiennej L następujące czynności:

Kolejne wystąpienia L w hiperprodukcji h są zastępowane nowymi zmiennymi afiksowymi $L_1, L_2, \ldots, L_n$: przy czym $D(L_1) = D(L_2) = \ldots = D(L_n) = D(L)$. Następnie na końcu hiperprodukcji h dopisywane jest hiperpojęcie predykatowe $equal_{L_h}(L_1, L_2, \ldots L_n)$ takie, że

$S_{equal_{Lh}} = (equal_{Lh}, n.[D(L_1), D(L_2), \ldots, D(L_n)] . [1,1,\ldots,1] . F_{equal_{Lh}})$, gdzie

$F_{equal_{Lh}}(x_1, x_2, \ldots, x_n) =$ prawda $\langle=\rangle x_1 = x_2 = \ldots = x_n$

Na końcu konstruktor uzupełnia zbiory Q,B,S o nowe wielkości.

(2) Załóżmy, że f jest dowolnym predykatem takim, że wszystkie jego afikspozycje są dziedziczone. Wtedy dla każdego takiego predykatu f konstruktor wykonuje czynności (już po zrealizowaniu czynności opisanych w (1)) :

Każde hiperpojęcie $f(F_1, F_2, \ldots, F_{N_f})$, występujące w jakiejś hiperprodukcji jest zastępowane hiperpojęciem $f(F_1, F_2, \ldots, F_{N_f}$, prawda), a następnie zamiast piątki $S_f = (f, N_f, A_N f, T^N f, F_f)$ umieszczamy w zbiorze S piątkę:

$S_f = (f, N_f+1, [M_1, \ldots, M_{N_f}, \text{PRAWDA}] . [1, \ldots, 1, \delta] . F'_f)$ taką, że $[M_1, \ldots, M_{Nf}] = A_N^{N_f}$, oraz $F'_f(x_1, \ldots, x_n, \text{prawda}) = F_f(x_1, \ldots, x_n)$.

Na zakończenie do zbioru R dodajemy produkcję: PRAWDA → prawda, do zbioru A_N dodajemy element PRAWDA, do zbioru A_T element prawda, do zbioru B element $\mathcal{T}$ i do zbioru D - element $(\mathcal{T}$, PRAWDA) .

Prawdziwe jest twierdzenie (jako oczywiste pozostawiamy je bez dowodu):

Twierdzenie 3.1.1

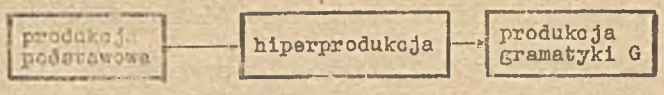
W wyniku działalności konstruktora, dobrze zdefiniowana gramatyka afiksowa G zostanie przekształcona w dobrze zdefiniowaną gramatykę afiksową G' taką, że:

$$L(G) = L(G')$$

Zmiany wprowadzone przez konstruktor są na tyle nieistotne, że mówiąc o wyprowadzeniu w G będziemy mieli zawsze na myśli wyprowadzenie w G'. Pojęć tych nie będziemy dalej rozróżniali.

3.2. PARSER WŁAŚCIWY

Parser właściwy zgodnie z uprzednimi ustaleniami znajduje wyprowadzenie w gramatyce G. Mówiąc ściślej: parser znajduje wywód w gramatyce podstawowej G_p. Następnie każdej produkcji tego wywodu zostaną przyporządkowane zawartości pamięci maszyny odpowiadające hiperpojęciom występującym w produkcji gramatyki afiksowej G (powstałej z hiperprodukcji odpowiadającej danej produkcji podstawowej). Będzie to więc schemat:



Ze względu na komunikatywność zdecydowano się każdy przebieg omówić dwa razy. Raz - omijając szczegóły i sposób realizacji pewnych czynności (punkty 3.2.1 i 3.2.2), a następnie drugi raz - już dokładnie.

3.2.1. PRZEBIEG PIERWSZY

Korzystać będziemy tutaj z faktu, że $L(G) \subseteq L(G_p)$.

Wejściem do przebiegu pierwszego będzie dowolne słowo $x \in V_T^*$. Wyjściem będzie wywód prawostronny x w gramatyce podstawowej G_p - jeżeli $x \in L(G_p)$, bądź sygnalizacja błędu składniowego - jeżeli $x \notin L(G_p)$. W trakcie wykonywania czynności przebiegu pierwszego, wywodowi $\sigma \overset{*}{\underset{G_p^R}{\Rightarrow}} x$ zostanie przyporządkowany jeden z grafów tego wywodu (definicja 2.5.3). Przyporządkowanie odbędzie się w ten sposób, że dla każdej produkcji z wywodu $\sigma \overset{*}{\underset{G_p^R}{\Rightarrow}} x$ będzie wiadomo, które węzły grafu jej odpowiadają.

Każdemu węzłowi (X,adr,i) ww grafu zostanie przydzielona pamięć o adresie równym wartości adr. Dla dowolnej wartości adr będzie więc N_X węzłów (N_X jest liczbą afikspozycji X) o adresie adr - w rzeczywistości pamięć o tym adresie będzie mieć n_X pozycji.

W pamięci przydzielonej węzłowi (X,adr,i) będziemy zapisywać dwie wielkości:

WARTOŚĆ (X,adr,i) oraz ILE (X,adr,i)

WARTOŚĆ (X,adr,i) będzie zawsze pewną formą zdaniową $M \in (A_N \cup A_T)^*$.

Formę tę nazwiemy wartością węzła (X,adr,i).

ILE (X,adr,i) będzie liczbą naturalną mówiącą ile strzałek omawianego grafu "wchodzi" w danej chwili do węzła (X,adr,i). Każda strzałka będzie oznaczona numerem hiperprodukcji, w której nastąpiło odpowiednie blokowanie.

Przebieg pierwszy nada wszystkim węzłom pewne wartości początkowe. Na wyjściu przebiegu pierwszego zostanie podany stos STOS (x). Stos ten zawierać będzie te węzły (X,adr,i), dla których ILE(X,adr,i) = 0. Ponieważ omawiana gramatyka nie zawiera pętli więc stos STOS(x) będzie niepusty.

Twierdzenie 3.2.1.1 (Dodatek B, cz.I, rozdz. 3)

Po wykonaniu wszystkich czynności przebiegu pierwszego stos STOS(x) zawiera tylko takie węzły (X,adr,1), że WARTOŚĆ $(X,adr,i) \in A_T^*$.

W trakcie wykonywania czynności parsera WARTOŚĆ(X,adr,i) będzie ulegać zmianom. Proces ten nazwiemy aktualizacją wartości węzła (X,adr,i). Zawsze jednak każda z wielkości WARTOŚĆ (X,adr,i) będzie taką formą M, że jeżeli A_i jest dziedziną i-tej afikspozycji X to $A_i \overset{*}{\underset{R}{\Rightarrow}} M$.

Celem parsera jest znalezienie wartości terminalnych wszystkich węzłów grafu wywodu $\sigma \overset{*}{\underset{G_p^R}{\Rightarrow}} x$ (tzn. takich wartości, że WARTOŚĆ $(X,adr,i) \in A_T^*$).

3.2.2. PRZEBIEG DRUGI

Czynności przebiegu drugiego, ogólnie mówiąc, polegają na trawersowaniu grafu wywodu $\sigma \overset{*}{\underset{G_p^R}{\Rightarrow}} x$ zgodnie ze wskazaniami strzałek oraz na wykonaniu operacji aktualizujących wartości węzłów.

Przebieg drugi używać będzie następujących procedur:

(1) AKTUAL $[(X,adr_1,i), (Y,adr_2,j)]$

Wejściem do tej procedury będą:

(a) Terminalna wartość węzła (X,adr_1,i).

(b) Aktualna wartość węzła (Y,adr_2,j) takiego, że istnieje strzałka od (X,adr_1,i) do (Y,adr_2,j).

(c) Numer hiperprodukcji p takiej, że nastąpiło w niej blokowanie określające powyższą strzałkę.

Wyjściem tej procedury będzie nowa "zaktualizowana" wartość węzła (Y,adr_2,j)

(2) PREDYKAT (X,adr)

Wejściem do tej procedury będą:

Terminalne wartości węzłów (X,adr,i_1), $(X,adr,i_2),\ldots,(X,adr,i_k)$ takich, że $X \in Q$ oraz $i_1,i_2,\ldots i_k$ są numerami wszystkich dziedziczonych afikspozycji predykatu X.

Wyjściem tej procedury będą:

Terminalne wartości węzłów (X,adr,j_1), $(X,adr,j_2)\ldots$, (X,adr,j_l) takich, że $j_1,j_2,\ldots,j_l$ są numerami wszystkich syntetyzowanych afikspozycji X.

(3) WYMAŻ $[(X,adr_1,i),\ (Y,adr_2,j)]$

Procedura ta "wyciera" strzałkę od węzła (X,adr_1,i) do węzła (Y,adr_2,j).

Wykonywać będziemy operacje na stosie STOS(x). Operacjom tym towarzyszyć będzie aktualizacja wartości odpowiednich węzłów.

Początkowa zawartość stosu STOS(x) obliczona została przez przebieg pierwszy.

Węzeł znajdujący się aktualnie na wierzchołku stosu STOS(x) oznaczymy niżej napisem WIRZCHOŁEK. Operację umieszczania na stosie STOS(x) elementu (X,adr,i) będziemy zapisywać w postaci:

STOS(x) : = STOS(x)+ (X,adr,i)

Operację zdjęcia ze stosu STOS(x) elementu, który aktualnie znajduje się na jego wierzchołku oznaczamy napisem:

STOS(x) : = ZDEJMIJ (STOS(x))

Zamiast "w przeciwnym przypadku" będziemy pisać "wpp".

ALGORYTM (przebieg drugi)

Krok 1 - sprawdzenie czy stos STOS jest pusty

Jeżeli STOS(x) = ϕ to wykonaj krok 10, wpp wykonaj krok 2.

Krok 2 - decyzja, która z procedur AKTUAL czy PREDYKAT będziemy wykonywać

Niech WIERZCHOŁEK = (X,adr,i)

Jeżeli $X \in V_N$ to wykonaj krok 3, wpp wykonaj krok 7 ($X \in Q$)

Krok 3 - sprawdzenie czy od węzła WIERZCHOŁEK prowadzi jakaś strzałka

Jeżeli (Y,adr_1,j) jest dowolnym węzłem takim, że prowadzi do niego strzałka od węzła WIERZCHOŁEK to wykonaj krok 4.

Jeżeli od węzła WIERZCHOŁEK nie prowadzi żadna strzałka to wykonaj krok 6.

Krok 4 - aktualizacja wartości węzła (Y,adr_1,j) oraz "wytarcie" odpowiedniej strzałki

Wykonaj procedurę AKTUAL $[$WIERZCHOŁEK, $(Y,adr_1,j)]$;

Wykonaj procedurę WYMAŻ $[$WIERZCHOŁEK, $(Y,adr_1,j)]$;

ILE $(Y,adr_1,j) := $ ILE $(Y,adr_1,j) - 1$.

Jeżeli ILE $(Y,adr_1,j) = 0$ to wykonaj krok 5, wpp wykonaj krok 3.

Krok 5 - umieszczenie węzła (do którego nie prowadzi żadna strzałka) na stosie STOS

STOS(x) := STOS(x) + (Y,adr_1,j) ; wykonaj krok 2

Krok 6 - zdjęcie ze stosu STOS węzła do którego nie prowadzi żadna strzałka

STOS(x) := ZDEJMIJ (STOS(x)); wykonaj krok 1

Krok 7 - sprawdzenie czy wszystkie argumenty procedury PREDYKAT są już obliczone

Niech węzły (X,adr,j_1), $(X,adr,j_2),\ldots,(X,adr,j_l)$ będą wszystkimi węzłami, do których prowadzi strzałka od węzła WIERZCHOŁEK.

Jeżeli ILE $(X,adr,j_1) = 1$ to wykonaj krok 8, wpp wykonaj krok 9

Krok 8 - wykonanie procedury PREDYKAT, "wytarcie" strzałek, "aktualizacja" stosu STOS

Wykonaj procedurę PREDYKAT (X,adr) ; gdzie WIERZCHOŁEK = (X,adr,i)

Wykonaj ciąg procedur: WYMAŻ $[$WIERZCHOŁEK,$(X,adr,j_1)]$; ...; WYMAŻ $[$WIERZCHOŁEK, $(X,adr,j_1)]$;

ILE(X,adr,j_1) := ILE (X,adr,j_2) := ... := ILE (X,adr,j_1) := 0;

STOS(x):= STOS(x) + (X,adr,j_1); ... ; STOS(x) := STOS(x) + (X,adr,j_1) ; wykonaj krok 3

Krok 9 - "wytarcie" strzałek, "aktualizacja" stosu STOS

Wykonaj ciąg procedur: WYMAŻ $[$WIERZCHOŁEK,$(X,adr,j_1)]$; ...; WYMAŻ $[$WIERZCHOŁEK, $(X,adr,j_l)]$;

ILE(X,adr,j_1) := ILE $(X,adr,j_1) - 1$; ...ILE (X,adr,j_l) :=ILE$(X,adr,j_l) -1$; wykonaj krok 3

Krok 10

STOP

Twierdzenie 3.2.2.1 (Dodatek B, cz.I,rozdz.3)

Po wykonaniu przebiegu drugiego, wartością każdego węzła (X,adr,i) grafu otrzymanego w przebiegu pierwszym jest napis M, taki, że $M \in A_T^*$.

Przebieg drugi jest algorytmem skończonym.

Zastanówmy się teraz nad wynikami otrzymanymi po zakończeniu działalności parsera: Załóżmy, że $p = x_0 \to \alpha_0 x_1 \alpha_1 x_2 \alpha_2 \ldots x_k \alpha_k$ ($\alpha_i \in V_T^*$, $X_i \in V_N$) jest dowolną produkcją podstawową zastosowaną w wywodzie $\sigma \underset{G_P}{\overset{*}{\Rightarrow}} x$.

Nie zmniejszając ogólności możemy przyjąć, że produkcji tej odpowiada hiperprodukcja

$h = X_0 (F_1^0,F_2^0,\ldots,F_{N_{X_0}}^0) \to \alpha_0 X_1 (F_1^1,\ldots,F_{N_{X_1}}^1)\alpha_1 \ldots x_k (F_1^k,\ldots,F_{N_{X_k}}^k)\alpha_k X_{k+1} (F_1^{k+1},\ldots,F_{N_{X_{k+1}}}^{k+1})\ldots$

$\ldots X_n (F_1^n,\ldots, F_{N_{X_n}}^n)$; gdzie $X_{k+1},\ldots,X_n \in Q$

Ponieważ dla każdej produkcji podstawowej, znalezionej przez przebieg pierwszy, wiemy jakie węzły grafu jej odpowiadają (omówione to było w punkcie 3.2.1) , więc jak wynika z twierdzenia 3.2.2.1, przebieg drugi przyporządkował produkcji p napis:

$X_0 (w_1^0,\ldots,w_{N_{X_0}}^0) \to \alpha_0 X_1 (w_1^1,\ldots,w_{N_{X_1}}^1)\alpha_1 \ldots X_k (w_1^k,\ldots,w_{N_{X_k}}^k)\alpha_k X_{k+1}(w_1^{k+1},\ldots,w_{N_{X_{k+1}}}^{k+1})\ldots X_n (w_1^n,\ldots,w_{N_{X_n}}^n)$

taki, że dla wszystkich i=0,1,...,n; $j=1,2,\ldots,N_X$

(1) w_j^i jest wartością węzła (X_i,adr_X,j)

(2) $w_j^i \in A_T^*$

Prawdziwe są następujące twierdzenia:

Twierdzenie 3.2.2.2 (Dodatek B, cz. I, rozdz.3)

Dla każdej produkcji podstawowej $p = x_0 \to \alpha_0 X_1 \alpha_1 X_2 \alpha_2 \ldots X_k \alpha_k$

napis $X_0 (W_1^0,\ldots,W_{N_{X_0}}^0) \to \alpha_0 X_1 (W_1^1,\ldots,W_{N_{X_1}}^1)\alpha_1 \ldots X_k (W_1^k,\ldots,W_{N_{X_k}}^k)\alpha_k \ldots X_n (W_1^n,\ldots,W_{N_{X_n}}^n)$

przyporządkowany jej przez przebieg drugi, jest produkcją gramatyki afiksowej G.

Twierdzenie 3.2.2.3 (Dodatek B, cz,III, rozdz.3)

Mając na wejściu ciąg napisów $A_{i_1},A_{i_2},\ldots,A_{i_m}$ taki, że $A_{i_j} \in A_T^*$ (j=1,2,...,m) oraz $i_1,i_2,\ldots,i_m$ są numerami wszystkich dziedziczonych afikspozycji predykatu X, procedura PREDYKAT (X,adr) poda na wyjściu ciąg napisów $W_{j_1},W_{j_2},\ldots,W_{j_l} \in A_T^*$ taki,że $j_1,j_2,\ldots,j_l$ są numerami wszystkich syntetyzowanych afikspozycji X oraz napis $X (F_1, F_2,\ldots,F_{N_X}) \to \varepsilon$ jest produkcją gramatyki G taką, że

$F_{i_1} = A_{i_1}, F_{i_2} = A_{i_2},\ldots,F_{i_m} = A_{i_m}, F_{j_1} = W_{j_1}, F_{j_2} = W_{j_2},\ldots,F_{j_l} = W_j$

Ogólnie więc, ciągowi produkcji podstawowych, znalezionych przez przebieg pierwszy, zostanie przez przebieg drugi przyporządkowany ciąg pewnych produkcji gramatyki G.

Twierdzenie 3.2.2.4 (Dodatek B, cz.I, rozdz.3)

Ciągowi produkcji podstawowych zastosowanych w wywodzie $\sigma \overset{*}{\underset{G_P}{\Rightarrow}} x$ zostanie, przez przebieg drugi, przyporządkowany ciąg produkcji gramatyki afiksowej G określających wyprowadzenie $\sigma \overset{*}{\underset{G}{\Rightarrow}} x$.

Zanim przejdziemy do dokładnego omówienia czynności wykonywanych przez parser wprowadzimy pewne pojęcie pomocnicze.

3.2.3. SUMA FORM ZDANIOWYCH

Załóżmy, że $G_a = (A_N, A_T, R, a)$ jest dowolną gramatyką bezkontekstową typu LR(k). Przypuśćmy dalej, że formy zdaniowe M, $N \in (A_N \cup A_T)^*$ spełniają warunki:

Istnieje $x \in (A_N \cup A_T)^*$ takie, że: (1) $a \overset{*}{\underset{G_a}{\Rightarrow}} M \overset{*}{\underset{G_a}{\Rightarrow}} x$

(2) $a \overset{*}{\underset{G_a}{\Rightarrow}} N \overset{*}{\underset{G_a}{\Rightarrow}} x$

Dla takich form (niekoniecznie prawostronnych) określimy ich sumę:

Definicja 3.2.3.1

Formę zdaniową $\sum \in (A_N \cup A_T)^*$ nazwiemy sumą form M i N jeżeli:

(1) $M \overset{*}{\underset{G_a}{\Rightarrow}} \sum$

(2) $N \overset{*}{\underset{G_a}{\Rightarrow}} \sum$

(3) dla każdej formy zdaniowej $\sum'$ spełniającej powyższe warunki (1) i (2) jest: $\sum \overset{*}{\underset{G_a}{\Rightarrow}} \sum'$

Jeżeli formy zdaniowe M i N nie spełniają warunków podanych przed definicją 3.2.3.1 to powiemy, że suma ich jest nieokreślona (dodawanie jest niewykonalne):

Sumę form M i N (jeżeli istnieje) będziemy oznaczać przez M+N.

Prawdziwe jest twierdzenie

Twierdzenie 3.2.3.1 (Dodatek B, część I, rozdz. 3)

Załóżmy, że $G_a = (A_N, A_T, R, a)$ jest gramatyką typu LR(k) oraz, M, N, F są formami zdaniowymi takimi, że istnieje $x \in (A_N \cup A_T)^*$ o własności: $a \overset{*}{\underset{G_a}{\Rightarrow}} F \overset{*}{\underset{G_a}{\Rightarrow}} x$, $a \overset{*}{\underset{G_a}{\Rightarrow}} M \overset{*}{\underset{G_a}{\Rightarrow}} x$, $a \overset{*}{\underset{G_a}{\Rightarrow}} N \overset{*}{\underset{G_a}{\Rightarrow}} x$. Wtedy istnieje dokładnie jedna forma zdaniowa $\sum$ taka, że $\sum = M + N$ oraz:

(1) $\sum \overset{*}{\underset{G_a}{\Rightarrow}} x$

(2) Jeżeli $M \in A_T^*$ to $M + N = N + M = M$

(3) $M + N = N + M$

(4) $F + (M+N) = (F + M) + N$

W dodatku C rozdz. 3 znajduje się algorytm tworzący dla zadanych form zdaniowych ich sumę, bądź sygnalizujący fakt niewykonalności dodawania.

3.2.4. DOKŁADNY OPIS PRZEBIEGU PIERWSZEGO

Przebieg pierwszy używać będzie parsera gramatyk typu LR(k) dla gramatyki podstawowej G_p. Zakładamy, że parser ten podaje najpierw ostatnią z produkcji zastosowanych w wywodzie prawostronnym analizowanego słowa $x \in V_T^*$, potem przedostatnią, itd.

W czasie realizacji czynności przebiegu pierwszego korzystać będziemy ze stosów STOS, STACK i WYWOD.

Przyjmijmy, że produkcja podstawowa p jest kolejną produkcją podaną przez parser gramatyki podstawowej. Niech odpowiada jej hiperprodukcja h postaci:

$$h = X_0 (F_1^0, \ldots, F_{H_{X_0}}^0) \to \alpha_0 X_1 (F_1^1, \ldots, F_{H_{X_1}}^1) \alpha_1 \ldots X_k (F_1^k, \ldots, F_{H_{X_k}}^k) \alpha_k \; X_{k+1} (F_1^{k+1}, \ldots, F_{H_{X_{k+1}}}^{k+1}) \ldots X_n (F_1^n, \ldots, F_{H_{X_n}}^n)$$

gdzie $\alpha_0, \alpha_1, \ldots, \alpha_k \in A_T^*$. Nie zmniejszając ogólności możemy założyć, że $X_0, X_1, \ldots, X_k \in V_N$; $X_{k+1}, \ldots, X_n \in Q$.

Dla hiperprodukcji h wykonywane będą następujące działania:

(A) Każdemu z symboli $X_0, X_{k+1}, \ldots, X_n$ zostanie przydzielona pamięć.

Pamięć przydzielona x_j $(j=0,k+1,\ldots,n)$ będzie się składać z N_{x_j} pozycji (N_{x_j} oznacza liczbę afikspozycji x_j). Na i-tej $(i=1,2,\ldots,N_{x_j})$ pozycji zapisywane będą trzy wielkości:

WARTOŚĆ (X_j, adr_j, i), S (X_j, adr_j, i) i ILE (x_j, adr_j, i), gdzie adr_j jest adresem pamięci przydzielonej X_j.

Następnie pamięć ta zostanie wypełniona:

WARTOŚĆ $(X_j, adr_j, i) := D(F_i^j)$

na stosie S (X_j, adr_j, i) zostaną umieszczone wszystkie trójki (adr_m, r, h) takie, że r-ta afikspozycja symbolu X_m (o adresie adr_m) jest blokowana w hiperprodukcji h przez i-tą afikspozycję x_j.

ILE $(X_j, adr_j, i) := t_j$ gdzie t jest liczbą afikspozycji blokujących w hiperprodukcji h i-tą afikspozycję symbolu X_j.

(B) Każdemu z symboli $X_1, X_2, \ldots, X_k$ pamięć została przydzielona i wypełniona już wcześniej.

Adres adr_j pamięci przydzielonej X_j $(j=1,2,\ldots,k)$ jest k-j+1-szym elementem stosu

STACK (licząc od wierzchołka). Teraz odbędzie się "aktualizacja" tej pamięci.
Dla każdego j=1,2,...,k zostaną wykonane instrukcje:

(a) dla wszystkich i=1,2,...,N_{xj}
WARTOŚĆ (X_j, adr_j, i) := WARTOŚĆ (X_j, adr_j, i) + D (F_i^j)

(b) Na każdym ze stosów S (X_j, adr_j, i) $(i=1,2,...,N_{xj})$ umieszczone zostaną wszystkie trójki (adr_m, r, h) takie, że r-ta afiksposycja symbolu X_m (o adresie adr_m) jest blokowana w hiperprodukcji h przez i-tą afiksposycję symbolu X_j.

(c) ILE (X_j, adr_j, i) := ILE (X_j, adr_j, i) + s ; gdzie s jest liczbą afiksposycji blokujących w hiperprodukcji h i-tą afiksposycję symbolu X_j.

(C) Po wykonaniu czynności opisanych w punktach (A) i (B), ze stosu STACK zostanie zdjętych k pierwszych elementów (będą to adresy symboli $X_1, X_2, ..., X_k$), a następnie będzie na tym stosie umieszczony adres symbolu X_0.

(D) Hiperprodukcji h zostanie przydzielona pamięć zawierająca adresy wszystkich występujących w niej symboli ze zbioru $V_N \cup Q$ (tzn. adresy odpowiadające $X_0, X_1, ..., X, X_{k+1}, ..., X_n$) oraz numer tej hiperprodukcji.

(E) Na stosie WYWOD umieszczony zostanie adres pamięci przydzielonej hiperprodukcji h.

Dodawanie opisane w podpunkcie (a) punktu (B) może się nie wykonać. Sygnalizujemy wtedy błąd semantyczny i przerywamy pracę parsera.

Twierdzenie 3.2.4.1 (Dodatek B, część I, rozdz.3)

Jeżeli analizowane słowo x należy do języka L (G) to czynności opisane w podpunkcie (a) punktu (B) wykonają się dla każdego węzła (X, adr, i).

Przebieg pierwszy tworzyć będzie stos STOS (x) umieszczając na nim te węzły (X, adr, i), dla których ILE(X, adr, i) = 0. Realizowane to będzie wtedy, gdy element X o adresie adr będzie zdejmowany ze stosu STACK.

3.2.5. DOKŁADNY OPIS CZYNNOŚCI PRZEBIEGU DRUGIEGO

Używać będziemy pewnych pomocniczych oznaczeń:

(1) Załóżmy, że M i T są dowolnymi (niekoniecznie prawostronnymi) formami zdaniowymi gramatyki $G_a = (A_N, A_T, R, a)$ typu LR(k), takimi że $a \overset{*}{\underset{G_a}{\Rightarrow}} M \overset{*}{\underset{G_a}{\Rightarrow}} T$. Przyjmijmy dalej, że $\alpha \in (A_N \cup A_T)^*$ jest napisem o własności: $M = \alpha L \beta$, gdzie $L \in A_N$, $\beta \in (A_N \cup A_T)^*$. Przez $\varphi(T, M, \alpha)$ będziemy wtedy oznaczać napis F spełniający warunki:

(i) istnieją $\alpha_1, \beta_1 \in (A_N \cup A_T)^*$ takie, że $T = \alpha_1 F \beta_1$

(ii) $\alpha \overset{*}{\underset{G_a}{\Rightarrow}} \alpha_1$, $\beta \overset{*}{\underset{G_a}{\Rightarrow}} \beta_1$, $L \overset{*}{\underset{G_a}{\Rightarrow}} F$ (α_1 i β_1 te same co w warunku (i)).

Fakt, że (dla zadanych M, T i α) taki napis F istnieje dokładnie jeden, wynika stąd, że gramatyka G_a jest typu LR(k).

$\varphi(T, M, \alpha)$ jest, intuicyjnie mówiąc, napisem, którym zastąpiliśmy afiks nieterminalny L, w trakcie konstrukcji wyprowadzenia $M = \alpha L \beta \overset{*}{\underset{G_a}{\Rightarrow}} T$. Zwróćmy uwagę, że lewy kontekst w jakim występuje afiks L (u nas α) wyznacza ten afiks jednoznacznie dla danej formy M.

(2) Pozostając przy oznaczeniach wprowadzonych w punkcie (1) przypuśćmy, że H jest dowolną (może być różną od F) formą zdaniową taką, że $L \overset{*}{\underset{G_n}{\Rightarrow}} H$.

Wtedy przez $P(T,M,\alpha,H)$ oznaczać będziemy napis $\alpha_1 H \beta_1$.

Jest to więc forma zdaniowa otrzymana z formy T przez zastąpienie napisu F napisem H.

Oczywiście $M \overset{*}{\underset{G_n}{\Rightarrow}} P(T,M,\alpha,H)$. Zauważmy, że aby znaleźć napis $P(T,M,\alpha,H)$ dla zadanych T,M,α,H trzeba wcześniej obliczyć $\phi(T,M,\alpha)$.

Znalezienie napisów $\phi(T,M,\alpha)$ i $P(T,M,\alpha,H)$ może się w tej chwili wydawać zadaniem trudnym do rozwiązania. Jednak w praktycznej realizacji okaże się ono bardzo proste. Wynika to z zastosowania innego, niż wypisanie explicite, zapisu form zdaniowych. Omawiamy dokładnie to zagadnienie w dodatku C niniejszej pracy.

I. Opis procedury AKTUAL $[(X,adr_1,i)\ ,\ (Y,adr_2,j)]$

Jak pamiętamy procedura AKTUAL będzie wykonywana wtedy, gdy na wierzchołku stosu STOS(x)pojawi się element (węzeł)(X,adr_1,i) taki, że prowadzi od niego strzałka do węzła (Y,adr,j). Załóżmy,że blokowanie określające tę strzałkę nastąpiło w hiperprodukcji h. Przyjmijmy, że w hiperprodukcji h na i-tej afikspozycji x występuje napis F_i oraz na j-tej afikspozycji Y występuje napis G_j. Aby nastąpiło blokowanie muszą istnieć $\alpha, \beta, \gamma, \delta$ i zmienna afiksowa L takie, że $F_i = \alpha L \beta$, $G_j = \gamma L \delta$. Oczywiście zmienna L może mieć kilka wystąpień zarówno w F_i jak i G_j. Niech:

$$F_i = \alpha_1 L \beta_1 = \alpha_2 L \beta_2 = \ldots = \alpha_n L \beta_n \quad \text{i}$$
$$G_j = \gamma_1 L \delta_1 = \gamma_2 L \delta_2 = \ldots = \gamma_m L \delta_m$$

będą wszystkimi takimi wystąpieniami.

Procedura AKTUAL $[(X,adr_1,i),\ (Y,adr_2,j)]$ jest wtedy ciągiem następujących, kolejno wykonywanych czynności:

$L_1 := \phi(\text{WARTOSC}\ (X,adr_1,i),\ D(F_i)\ ,\ D(\alpha_1)) + \ldots + \phi(\text{WARTOSC}\ (X,adr_1,i),\ D(F_i),\ D(\alpha_n))$;

$L_2 := \phi(\text{WARTOSC}\ (Y,adr_2,j),\ D(G_j)\ ,\ D(\gamma_1) + \ldots + \phi(\text{WARTOSC}\ (Y,adr_2,j),\ D(G_j),\ D(\gamma_m))$;

$L_3 := L_1 + L_2$;

$\text{WARTOSC}\ (Y,adr_2,j) := P(\text{WARTOSC}(Y,adr_2 j),\ D(G_j),\ D(\gamma_1),\ L_3)$;

$\text{WARTOSC}\ (Y,adr_2,j) := P(\text{WARTOSC}\ (Y,adr_2,j),D(G_j),\ D(\gamma_2),\ L_3)$;

.
.
.

$\text{WARTOSC}\ (Y,adr_2,j) := P(\text{WARTOSC}\ (Y,adr_2,j),\ D(G_j),\ D(\gamma_m),\ L_3)$.

(dodawanie, o którym tutaj mowa to oczywiście dodawanie form zdaniowych).

Analogiczne czynności wykonuje procedura AKTUAL dla wszystkich zmiennych afiksowych o własnościach takich jak zmienna L.

Zastanówmy się teraz jaki jest sens opisanych wyżej czynności:

Ponieważ jeszcze przed wykonaniem procedury AKTUAL węzeł (X,adr_1,i) ma wartość terminalną, więc każdy napis $\phi(\text{WARTOSC}(X,adr_1,i),D(F_i)\ D(\alpha_k)(k=1,2,\ldots,n)$ też będzie mieć wartość terminalną. Napis $\phi(\text{WARTOSC}(X,adr_1,i),\ D(F_i),\ D(\alpha_k)$ będziemy często nazywać wartością zmiennej L dla jej wystąpienia $F_i = \alpha_k L \beta_k$ na i-tej afikspozycji X).

Jeżeli wszystkie te napisy będą takie same (a zgodnie z umową o zastępowaniu tych samych zmiennych afiksowych tymi samymi napisami - powinny być) to działanie prowadzące do obliczenia L_1 wykona się (dodawanie w tym przypadku sprowadza się do sprawdzenia równości napisów). W przeciwnym przypadku sygnalizujemy błąd semantyczny. Podobny cel ma obliczenie L_2. L_3 obliczamy aby sprawdzić czy wartość zmiennej L występującej w i-tej afikspozycji X (zmienna L ma tutaj wartość terminalną) można podstawić na zmienną L występującą w j-tej afikspozycji Y. Jeżeli dodawanie L_1+L_2 jest niewykonalne, to znaczy, że ww. podstawienie jest niemożliwe - co sygnalizujemy jako błąd semantyczny.

W praktycznej realizacji działania prowadzące do obliczenia L_1, L_2 i L_3 są bardzo proste.

Ostatnie czynności są właśnie omawianymi podstawieniami odpowiedniej wartości terminalnej w miejsce każdego wystąpienia zmiennej L w G_j.

Twierdzenie 3.2.5.1 (Dodatek B, część I, rozdz.3)

Jeżeli analizowane słowo należy do języka L(G), to wszystkie opisane wyżej czynności wykonają się. Wykona się więc procedura AKTUAL.

II. Opis procedury PREDYKAT

Procedura PREDYKAT (X,adr) sprowadza się do obliczenia wartości funkcji $\tilde{F}_x$ dla zadanych argumentów $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$; gdzie $i_1, i_2, \ldots, i_k$ są numerami wszystkich dziedziczonych afikspozycji predykatu X (patrz punkt 2.4). Jeżeli $\tilde{F}_x$ przyjmuje wartość "fałsz" dla danych $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$, to fakt ten sygnalizujemy jako błąd semantyczny i przerywamy pracę parsera.

Rozdział 4. ZMIENNE GLOBALNE

Zmienne globalne, które wprowadzimy teraz nieformalnie (formalna definicja zmiennych globalnych dla gramatyk afiksowych znajduje się w dodatku B, cz.I, rozdz.4) nie wnoszą "ideowo" nic nowego. Są natomiast dużym udogodnieniem - ułatwiającym zarówno opis języka, jak i skracającym czas pracy parsera.

Intuicyjnie mówiąc - istnieją takie wielkości (nazywane przez Knutha atrybutami globalnymi), które mogą być obliczane tylko raz w dowolnym wywodzie gramatyki, natomiast wykorzystywane są wielokrotnie. Przykładami takich wielkości są: tablica deklaracji zmiennych występujących w programie, bądź tablica deklaracji etykiet (tzn. ich wystąpień przed zdaniami programu). Wielokrotne korzystanie z owych wielkości powoduje na ogół konieczność umieszczania ich w postaci afikspozycji każdego prawie symbolu nieterminalnego oraz konieczność nieustannego "przekazywania" wartości tych afikspozycji. Następstwem tego jest oczywiście duża strata czasu pracy parsera i znacznie zwiększona zajętość pamięci maszyny. Jest to również niewygodne dla definiującego język.

Poniższa nieformalna definicja zmiennych globalnych korzysta z założenia, że akajomat może występować jedynie po lewej stronie dowolnej hiperprodukcji (nie zmniejsza to w niczym ogólności rozważań).

Definicja 4.1

Przyjmijmy, że G jest gramatyką afiksową postaci:

$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$

Wybierzmy ze zbioru zmiennych afiksowych B podzbiór GLOBAL mający tyle elementów ile wynosi liczba afikspozycji aksjomatu σ.

Będziemy zakładać, że zbiór GLOBAL jest zbiorem uporządkowanym. Elementy zbioru GLOBAL oznaczymy g_1 (pierwszy element), g_2 (drugi element), itd. i nazwiemy zmiennymi globalnymi.

Każda ze zmiennych globalnych może wystąpić w dowolnej hiperprodukcji na dowolnej afikspozycji dowolnego symbolu nieterminalnego z wyjątkiem aksjomatu bądź predykatu.

Zmiennej globalnej g_1 będzie zawsze przypisywana wartość pierwszej afikspozycji aksjomatu σ gramatyki G, zmiennej globalnej g_2 - wartość drugiej afikspozycji σ, itd.

W związku z tym przyjmiemy następującą definicję:

Definicja 4.2

Załóżmy, że hiperprodukcja h jest postaci:

$h = X_0(F_1^0, \ldots, F_{N_{X_0}}^0) \to \alpha_0 X_1(F_1^1, \ldots, F_{N_{X_1}}^1)\alpha_1 \ldots X_n(F_1^n, \ldots, F_{N_{X_n}}^n)\alpha_n$ gdzie $\alpha_0, \alpha_1, \ldots, \alpha_n \in V_T^*$.

Powiemy, że i-ta afikspozycja aksjomatu σ blokuje w hiperprodukcji h j-tą afikspozycję symbolu X_k $(k=0,1,\ldots,n)$ jeżeli:

$$F_j^k = \beta g_1 \delta$$

Zgodnie z definicją 4.2 ulegnie odpowiedniej zmianie definicja grafu wywodu dla gramatyki ze zmiennymi globalnymi.

Definicja 4.3

Niech $\sigma \overset{*}{\underset{G_p}{\Rightarrow}} \alpha X \beta$ (gdzie $\alpha \in (V_N \cup V_T)^*$, $X \in V_N$, $\beta \in V_T^*$) będzie dowolnym wywodem prawostronnym w gramatyce podstawowej G_p.

Załóżmy dalej, że $p = X \to y$ (gdzie $y \in (V_N \cup V_T)^*$) jest produkcją podstawową. Oznaczamy przez GRAF dowolny, ale ustalony graf wywodu $\sigma \underset{G_p R}{\Rightarrow} \alpha Y \beta$. Przyjmijmy jeszcze, że produkcji p odpowiada hiperprodukcja h.

Grafem wywodu $\sigma \overset{*}{\underset{G_p}{\Rightarrow}} \alpha Y \beta$ dla gramatyki ze zmiennymi globalnymi nazwiemy graf GZG o własnościach: (definicja rekurencyjna):

(1) Trójka (A, adr_1, i) jest węzłem grafu GZG wtedy i tylko wtedy gdy jest ona węzłem grafu GRAF.

(2) Graf GZG ma strzałkę od węzła (A, adr_1, i) do węzła (B, adr_2, j) wtedy i tylko wtedy gdy ma miejsce jeden z przypadków:

(a) strzałka taka istnieje w grafie GRAF, bądź

(b) strzałka taka istnieje w grafie wywodu $\sigma \overset{*}{\underset{G_p R}{\Rightarrow}} \alpha X \beta$ dla gramatyki ze zmiennymi globalnymi, bądź

(c) $A = \sigma$ oraz i-ta afikspozycja σ blokuje w hiperprodukcji h j-tą afikspozycję symbolu B.

Jak już powiedzieliśmy, pojęcie zmiennych globalnych nie wprowadza "ideowo" nic nowego.

Jedyna właściwie różnica między gramatykami ze zmiennymi globalnymi, a "zwyczajnymi" gramatykami polega na innej definicji grafu wywodu. Każdą gramatykę ze zmiennymi globalnymi łatwo zastąpić "zwyczajną" gramatyką wg schematu:

(1) Jeżeli $g_1, g_2, \ldots, g_n$ są wszystkimi elementami zbioru GLOBAL, to dla każdego symbolu $X \in V_N \cup Q$ $(X \neq \delta)$ zwiększamy o n liczbę jego afikspozycji. Odpowiednia piątka S_X będzie więc mieć teraz postać:

$$S_X = (X, N_X + n, [g_1, \ldots, g_n, A_1, \ldots, A_{N_X}], [2, \ldots, 2, t_1, t_2, \ldots, t_{N_X}], F_X)$$

gdzie A_i i t_i odpowiednio dziedzina i typ i-tej afikspozycji X w gramatyce ze zmiennymi globalnymi.

(2) Każdą hiperprodukcję (gramatyki ze zmiennymi globalnymi) postaci

$$\delta(G_1, \ldots, G_{N_\delta}) \to \alpha_0 X_1(F_1^1, \ldots, F_{N_{X_1}}^1)\alpha_1 \ldots X_m(F_1^m, \ldots, F_{N_{X_m}}^m)\alpha_m$$

Zastępujemy hiperprodukcją (gramatyki "zwyczajnej")

$$\delta(G_1, \ldots, G_{N_\delta}) \to \alpha_0 X_1(G_1, \ldots, G_{N_\delta}, F_1^1, \ldots, F_{N_{X_1}}^1)\alpha_1 \ldots X_m(G_1, \ldots, G_{N_\delta}, F_1^m, \ldots, F_{N_{X_m}}^m)\alpha_m$$

(3) Każdą hiperprodukcję (gramatyki ze zmiennymi globalnymi) postaci:

$X_0(F_1^0, \ldots, F_{N_{X_0}}^0) \to \alpha_0 X_1(F_1^1, \ldots, F_{N_{X_1}}^1)\alpha_1 \ldots X_m(F_1^m, \ldots, F_{N_{X_m}}^m)\alpha_m$ gdzie $X_0 \neq \delta$ i $\alpha_0 \alpha_1 \ldots \alpha_m \in V_T^*$

Zastępujemy hiperprodukcją (gramatyki "zwyczajnej")

$$X_0(g_1, \ldots, g_n, F_1^0, \ldots, F_{N_{X_0}}^0) \to \alpha_0 X_1(g_1, \ldots, g_n, F_1^1, \ldots, F_{N_{X_1}}^1)\alpha_1 \ldots X_m(g_1, \ldots, g_n, F_1^m, \ldots, F_{N_{X_m}}^m)\alpha_m$$

(przypominamy, że $n = N_\delta$)

Jak łatwo zauważyć zmiana ta nie ma wpływu na generowany język. Badając gramatykę "zwyczajną", otrzymaną z gramatyki ze zmiennymi globalnymi możemy sprawdzić czy ta ostatnia ma pętlę. Pojęcie dobrze zdefiniowanej gramatyki afiksowej ze zmiennymi globalnymi definiujemy analogicznie jak dla gramatyki "zwyczajnej".

Nie będziemy tutaj omawiać metody konstrukcji parsera dla gramatyk ze zmiennymi globalnymi. Metoda ta jest identyczna z podaną wcześniej dla "zwyczajnych" gramatyk afiksowych. Jedyna różnica polega na konieczności zapamiętywania adresów węzłów blokowanych przez zmienne globalne będzie to robione w trakcie konstruowania grafu wywodu przebieg pierwszy . Adresy te będą potem adresami węzłów blokowanych przez odpowiednią afikspozycję aksjomatu gramatyki.

Wprowadzone tutaj pojęcia ilustruje przykład 8.

Rozdział 5. PODSUMOWANIE

W rozdziale pierwszym podaliśmy twierdzenie mówiące, że klasa gramatyk afiksowych generuje wszystkie języki rekurencyjnie przeliczalne.

Natomiast, w związku z wprowadzonymi przez nas ograniczeniami, klasa języków generowanych przez dobrze zdefiniowane gramatyki afiksowe będzie odpowiednio mniejsza:

Twierdzenie 5.1 (Dodatek B, część I, rozdz.4)

Dla każdego języka rekurencyjnego istnieje dobrze zdefiniowana gramatyka afiksowa, która go generuje.

Z dowodu tego twierdzenia wynika, że jeżeli usunąć warunek dobrej definiowalności zbioru predykatów (pozostawiając pozostałe warunki) to tak otrzymana klasa gramatyk generowałaby wszystkie języki rekurencyjnie przeliczalne.

Osobnym zagadnieniem są wyniki osiągane przez zaostrzenie bądź osłabienie pozostałych warunków.

Okazuje się, że jeżeli pozostawić warunek jednoznaczności każdej gramatyki $G_a=(A_N,A_T,R,a)$ (gdzie a jest dowolnym elementem zbioru A_N) oraz jednoznaczności gramatyki podstawowej, to zmiany wprowadzone dla pozostałych warunków nie wpływają na ogół na klasę generowanych języków. Z dowodu twierdzenia 5.1 wynika na przykład, że klasa dobrze zdefiniowanych gramatyk afiksowych takich, że każdy symbol nieterminalny danej gramatyki z tej klasy, ma wszystkie afikspozycje syntetyzowane - również generuje wszystkie języki rekurencyjne. Warunki dotyczące typów afikspozycji mają jednak duże znaczenie z punktu widzenia konstrukcji parsera.

Omówimy to trochę dokładniej na przykładzie warunku (4) z definicji 2.5.1 mówiącego, że gramatyka dobrze zdefiniowana nie może posiadać pętli. Warunkowi temu odpowiada w pracy D.Watta (D.Watt [16]) warunek: Jeżeli zmienna afiksowa L ma w hiperprodukcji p wystąpienie stosowane (czyli nie definiujące) w hiperpojęciu Z po prawej stronie p to zmienna L musi mieć wystąpienie definiujące albo po lewej stronie p, albo po prawej stronie p w hiperpojęciu występującym na lewo od Z.

Nieformalnie mówiąc: dla każdego grafu wywodu w danej gramatyce afiksowej nie może istnieć strzałka "w lewo" w tym grafie.

Wymienimy tu kilka następstw wynikających z warunku D.Watta:

(1) Często duża niewygoda dla definiującego język

(2) Konieczność "przekazywania" wielu istotnych wielkości (jak np. tablica wystąpień etykiet) w "górę" grafu wywodu. Powoduje to duże straty czasu pracy parsera i znaczny wzrost zajętości pamięci maszyny. Wielkości te przy dopuszczeniu "przekazań w lewo" w ogóle nie występują.

(3) Wadliwa sygnalizacja błędu - często nie tam gdzie błąd wystąpił (brak lokalizacji wystąpienia błędu)

(4) Nieoptymalny kod wynikowy (parser jest uruchomiany na ogół jako część translatora)

(5) Zwiększenie (w porównaniu z metodą proponowaną w niniejszej pracy) liczby afikspozycji, a często i liczby hiperprodukcji.

(6) Niemożność zastosowania zmiennych globalnych tam gdzie przynoszą one istotne korzyści (np. dla tablicy deklaracji zmiennych czy etykiet)

(7) Warunek Watta wpływa pośrednio na postać produkcji gramatyki podstawowej. Np. w związku z zakazem używania "przekazań w lewo" nie jest obojętne czy stosuje się rekursję lewo- czy prawostronną. Powoduje to konieczność używania odpowiedniego parsera gramatyki podstawowej.

Wymienione wyżej wady nie występują w metodzie proponowanej w niniejszej pracy.

Metoda Watta przystosowana jest do języków, które dają tłumaczyć się bezpośrednio od lewej do prawej, natomiast metoda przedstawiona przez nas dostosowana jest do języków, które nie muszą spełniać tego warunku.

Część druga - NIEZORIENTOWANE GRAMATYKI AFIKSOWE

W części pierwszej wprowadziliśmy gramatyki afiksowe jako pewne narzędzie służące do opisu języków programowania. Jedną z wad owych gramatyk jest konieczność określania typów wszystkich afikspozycji symboli nieterminalnych. Zmusza to definiującego język do przybliżonego przynajmniej wyobrażenia sobie jak będzie działał parser definiowanego języka.

Proponowane tutaj niezorientowane gramatyki afiksowe nie posiadają tej wady.

Rozdział 1. DEFINICJA

Definicja 1.1

Niezorientowaną gramatyką afiksową nazwiemy układ:

$$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma), \quad \text{gdzie:}$$

znaczenie wszystkich symboli z wyjątkiem S jest takie samo jak dla gramatyk afiksowych (definicja 1.1 w części pierwszej),

S jest zbiorem czwórek takim, że dla każdego $X \in V_N \cup Q$ istnieje dokładnie jedna czwórka S_X należąca do zbioru S.

S_X jest postaci: $S_X = (X, N_X, A_N^{N_X}, F_X)$

Znaczenie N_X, $A_N^{N_X}$, F_X jest takie samo jak dla gramatyk afiksowych.

Pojęcia produkcji, wyprowadzenia, języka oraz gramatyki podstawowej definiujemy analogicznie jak dla gramatyk afiksowych.

Oczywiście klasa języków generowanych przez niezorientowane gramatyki afiksowe jest taka sama jak klasa języków generowanych przez gramatyki afiksowe. Są to więc wszystkie języki rekurencyjnie przeliczalne. Klasę niezorientowanych gramatyk afiksowych będziemy dalej oznaczać NGA.

Rozdział 2. DOBRZE ZDEFINIOWANE GRAMATYKI NGA

Podobnie jak dla gramatyk afiksowych wyodrębnimy wśród gramatyk NGA podklasę, dla której będziemy umieli podać metodę automatycznej konstrukcji parsera.

2.1. JEDNOZNACZNE GRAMATYKI NGA

Definicja 2.1.1

Niech $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ będzie gramatyką NGA. Załóżmy, że $Y \to Y_1 Y_2 \ldots Y_m$ jest jedną z produkcji tej gramatyki.

Wtedy: Napiszemy $\alpha Y \beta \underset{G}{\Rightarrow}_R \alpha Y_1 Y_2 \ldots Y_m \beta$ jeżeli $\beta \in V_T^*$,

Napiszemy $X \overset{*}{\underset{G}{\Rightarrow}}_R Y$ i powiemy, że X wyprowadza prawostronnie Y w gramatyce G jeżeli: istnieje ciąg $X_0, X_1, \ldots, X_n$ o własnościach:

(1) $X_0 = X$

(2) $X_n = Y$

(3) dla każdego $i=1,2,\ldots,n$ jest $X_{i-1} \overset{}{\underset{G}{\Rightarrow}}_R X_i$

Ów ciąg $X_0, X_1, \ldots, X_n$ będziemy nazywać wyprowadzeniem prawostronnym Y z X.

Załóżmy teraz, że $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ jest dowolną niezorientowaną gramatyką afiksową oraz, że

$$h = X_0 (F^0_1, \ldots, F^0_{N_{X_0}}) \alpha_0 X_1 (F^1_1, \ldots, F^1_{N_{X_1}}) \alpha_1 \ldots X_k (F^k_1, \ldots, F^k_{N_{X_k}}) \alpha_k X_{k+1} (F^{k+1}_1, \ldots, F^{k+1}_{N_{X_{k+1}}}) X_n (F^n_1, \ldots, F^n_{N_{X_n}})$$

gdzie $\alpha_0, \alpha_1, \ldots, \alpha_k \in V_T^*$; $X_0, X_1, \ldots, X_k \in V_N$; $X_{k+1}, \ldots, X_n \in Q$

jest dowolną hiperprodukcją tej gramatyki.

Wtedy hiperprodukcję h_u postaci:

$$h_u = X_0 (F^0_1, \ldots, F^0_{N_{X_0}}) \to \alpha_0 X_1 (F^1_1, \ldots, F^1_{N_{X_1}}) \alpha_1 \ldots X_k (F^k_1, \ldots, F^k_{N_{X_k}}) \alpha_k$$

nazwiemy uproszczoną postacią hiperprodukcji h.

Definicja 2.1.2

Niech $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ będzie gramatyką NGA.

Wówczas niezorientowaną gramatykę afiksową:

$$G_u = (V_N, V_T, A_N, A_T, Q_u, R, B, D, S_u, P_u \sigma)$$

nazwiemy uproszczoną postacią gramatyki G jeżeli:

(1) $Q_u = \phi$

(2) $S_u = \{ S_x \in S \mid x \in V_N \}$

(3) $P_u = \{ h_u \mid$ istnieje $h \in P$ taka, że h_u jest uproszczoną postacią $h \}$

Definicja 2.1.3

Przyjmijmy, że $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ jest gramatyką NGA oraz, że gramatyka G_u jest uproszczoną postacią gramatyki G.

Gramatykę G nazwiemy jednoznaczną gramatyką NGA (JNGA) jeżeli spełnia ona warunki:

(1) Dla każdego $a \in A_N$ gramatyka $G_a = (A_N, A_T, R, a)$ jest gramatyką jednoznaczną

(2) Gramatyka podstawowa G_P gramatyki G jest jednoznaczna

(3) Dla każdego $x \in L(G)$ istnieje dokładnie jedno wyprowadzenie prawostronne $\sigma \overset{*}{\underset{G_u}{\Rightarrow}}_R X$ w gramatyce G_u.

Problem, czy dowolna gramatyka NGA jest jednoznaczną gramatyką NGA jest problemem nierozstrzygalnym.

Twierdzenie 2.1 (Dodatek B, część II, rozdz. 2)

Dla każdego języka rekurencyjnie przeliczalnego istnieje generująca go jednoznaczna gramatyka NGA.

2.2. DEFINICJA DOBRZE ZDEFINIOWANYCH GRAMATYK NGA

Z powodów analogicznych jak dla gramatyk afiksowych (wybór parsera dla gramatyk typu LR (k)) przyjmiemy definicje:

Definicja 2.2.2

Powiemy, że zbiór produkcji afiksowych R, jednoznacznej niezorientowanej gramatyki afiksowej $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, 6)$, jest dobrze zdefiniowany jeżeli dla każdego $a \in A_N$ istnieje k takie, że $G_a = (A_N, A_T, R, a)$ jest gramatyką typu LR (k).

Definicja 2.2.3

Powiemy, że zbiór predykatów Q, jednoznacznej niezorientowanej gramatyki afiksowej $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, 6)$ jest dobrze zdefiniowany, jeżeli dla każdego predykatu $q \in Q$ związana z nimi funkcja F_q jest ogólnie rekurencyjna.

Podsumowując:

Definicja 2.2.4

Powiemy, że jednoznaczna niezorientowana gramatyka afiksowa G jest dobrze zdefiniowana jeżeli:

(1) Gramatyka podstawowa gramatyki G jest dobrze zdefiniowana,

(2) Zbiór produkcji afiksowych gramatyki G jest dobrze zdefiniowany,

(3) Zbiór predykatów gramatyki G jest dobrze zdefiniowany.

Problem czy dana (dowolna) jednoznaczna gramatyka afiksowa jest dobrze zdefiniowana jest problemem nierozstrzygalnym.

Twierdzenie 2.2.1 (Dodatek B, część II, rozdz. 2)

Dla dowolnego języka ogólnie rekurencyjnego istnieje generująca go dobrze zdefiniowana jednoznaczna gramatyka NGA.

Wprowadzone tu pojęcia ilustruje przykład 9.

Rozdział 3. PARSER

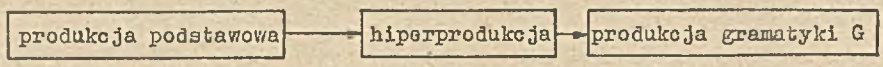
W rozdziale tym podamy metodę automatycznej konstrukcji parsera dla dowolnej dobrze zdefiniowanej jednoznacznej gramatyki NGA.

Parser ten podzielimy na dwa kolejno wykonywane przebiegi.

Przebieg pierwszy będzie - mówiąc intuicyjnie - odpowiadał analizie "z dołu do góry",

przebieg drugi analizie - "z góry na dół". Korzystając z metody typu LR(k) znajdziemy wywód analizowanego słowa $x \in V_T^*$ w gramatyce podstawowej G_p gramatyki G.

Następnie każdej produkcji tego wywodu zostaną przyporządkowane zawartości pamięci maszyny odpowiadające hiperpojęciom występującym w produkcji gramatyki G (powstałej z hiperprodukcji odpowiadającej danej produkcji podstawowej). Podobnie jak dla gramatyk afiksowych będzie to więc schemat:



We wszystkich dalszych rozważaniach ustalimy dowolną dobrze zdefiniowaną jednoznaczną gramatykę NGA:

$$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$$

3.1. PRZEBIEG PIERWSZY

W czasie realizacji czynności przebiegu pierwszego korzystać będziemy z parsera gramatyk typu LR(k) oraz ze stosów STACK i WYWOD. O parserze gramatyk typu LR(k) zakładamy, że jest to algorytm, który mając na wejściu analizowane słowo $x \in V_T^*$ poda na wyjściu ciąg produkcji podstawowych użytych w wywodzie prawostronnym tego słowa w gramatyce podstawowej G_p, bądź spowoduje sygnalizację błędu składniowego jeżeli $x \notin L(G_p)$.

Zakładamy, że parser ten poda najpierw ostatnią z produkcji zastosowanych w wywodzie $\sigma \underset{G_p}{\overset{n}{\Longrightarrow}}_R x$, potem przedostatnią, itd.

Przyjmijmy, że produkcja podstawowa p jest kolejną produkcją podaną przez parser gramatyki podstawowej.

Niech produkcji p odpowiada hiperprodukcja h postaci:

$$h = X_o(F_1^o,\ldots,F_{N_{X_o}}^o) \to \alpha_o X_1 (F_1^1,\ldots,F_{N_{X_1}}^1)\alpha_1 \ldots X_k (F_1^k,\ldots,F_{N_{X_k}}^k)\alpha_k X_{k+1}(F_1^{k+1},\ldots,F_{N_{X_{k+1}}}^{k+1}\ldots)$$
$$\ldots X_n (F_1^n,\ldots,F_{N_{X_n}}^n)$$

gdzie $\alpha_o, \alpha_1,\ldots,\alpha_k \in A_T^*$. Nie zmieniając ogólności możemy założyć, że $X_1, X_2,\ldots,X_k \in V_N$; $X_{k+1},\ldots,X_n \in Q$.

Dla hiperprodukcji h przebieg pierwszy wykona następujące czynności:

(A) Każdemu z elementów $X_o, X_{k+1},\ldots,X_n$ zostanie przydzielona pamięć.
Pamięć przydzielona X_j ($j=0,k+1,\ldots,n$) składać się będzie z N_{X_j} pozycji. (N_{X_j} jest liczbą afikspozycji symbolu x_j)
Zawartość i-tej ($i=1,2,\ldots,N_{X_j}$) pozycji oznaczymy przez WARTOŚĆ (X_j, adr_j, i) i będziemy nazywać wartością i-tej afikspozycji symbolu X_j o adresie adr_j.
Dla wszystkich $j=0,k+1,\ldots,n$; $i=1,2,\ldots N_{X_j}$ wykonane zostaną instrukcje:

$$\text{WARTOŚĆ}\ (X_j, adr_j, i) := D(F_i^j)$$

(B) Każdemu z symboli $X_1, X_2, \ldots, X_k$ pamięć została przydzielona (i odpowiednio wypełniona) już wcześniej (przypominamy o kolejności w jakiej są podawane produkcje przez parser gramatyki podstawowej). Teraz odbędzie się "aktualizacja" tej pamięci:
dla wszystkich j=1,2,...,k; i=1,2,...,N

$$\text{WARTOŚĆ}(X_j, adr_j, i) := \text{WARTOŚĆ}(X_j, adr_j, i) + D(F_i^j)$$

Przy czym adres adr_j odpowiadający X_j jest k-j+1-szym elementem (licząc od wierzchołka) stosu STACK.

(C) Załóżmy, że $F_{i_1}^{j_1} = \alpha_1 L \beta_1$, $F_{i_2}^{j_2} = \alpha_2 L \beta_2, \ldots$, $F_{i_m}^{j_m} = \alpha_m L \beta_m$ są wszystkimi wystąpieniami zmiennej L w hiperprodukcji h.

Wykonane zostaną wtedy działania:

$$L_1 := \Phi(\text{WARTOŚĆ}(X_{j_1}, adr_{j_1}, i_1), D(F_{i_1}^{j_1}), D(\alpha_1)) + \Phi(\text{WARTOŚĆ}(X_{j_2}, adr_{j_2}, i_2), D(F_{i_2}^{j_2}), D(\alpha_2)) +$$
$$+ \ldots + \Phi(\text{WARTOŚĆ}(X_{j_m}, adr_{j_m}, i_m), D(F_{i_m}^{j_m}), D(\alpha_m));$$

$$\text{WARTOŚĆ}(X_{j_1}, adr_{j_1}, i_1) := P(\text{WARTOŚĆ}(X_{j_1}, adr_{j_1}, i_1), D(F_{i_1}^{j_1}), D(\alpha_1), L_1)$$

$$\ldots; \text{WARTOŚĆ}(X_{j_m}, adr_{j_m}, i_m) := P(\text{WARTOŚĆ}(X_{j_m}, adr_{j_m}, i_m), D(F_{i_m}^{j_m}), D(\alpha_m), L_1)$$

gdzie dodawanie, funkcje Φ i P mają znaczenie takie jak zdefiniowaliśmy dla gramatyk afiksowych. adr_j, $adr_j, \ldots, adr_j$ oznaczają adresy pamięci przydzielonej X_j, $X_j, \ldots, X_j$ Analogiczne działania zostaną wykonane dla wszystkich zmiennych występujących w hiperprodukcji h.

(D) Po wykonaniu czynności opisanych w (A), (B) i (C) ze stosu STACK zostanie zdjętych k pierwszych elementów (będą to adresy $adr_1, adr_2, \ldots, adr_k$) a na ich miejsce zostanie umieszczony adres adr_0 (adres pamięci przydzielonej x_0).

(E) Hiperprodukcji h zostanie przydzielona pamięć, w której umieszczony będzie numer tej hiperprodukcji oraz adresy $adr_0, adr_1, \ldots, adr_k, adr_{k+1}, \ldots, adr_n$.

(F) Na stosie WYWOD umieszczony zostanie adres pamięci przydzielonej hiperprodukcji h.

Jeżeli którekolwiek z opisanych wyżej działań (dodawanie) jest niewykonalne to parser powoduje sygnalizację błędu.

Twierdzenie 3.1.1 (Dodatek B, część II, rozdz. 3)

Jeżeli analizowane słowo x należy do języka L(G) to wszystkie czynności przebiegu pierwszego są wykonalne.

3.2. PRZEBIEG DRUGI

Podczas przebiegu drugiego powtórnie będą analizowane hiperprodukcje odpowiadające produkcjom znalezionym przez parser gramatyki podstawowej. Teraz jednak hiperprodukcje te będą analizowane w odwrotnej kolejności niż w czasie przebiegu pierwszego. W trakcie realizacji przebiegu drugiego skonstruowany zostanie stos WYNIK.

Rozdział 2. DOBRZE ZDEFINIOWANE ROZSZERZONE GRAMATYKI AFIKSOWE

Podobnie jak dla gramatyk afiksowych, na rozszerzone gramatyki afiksowe nałożymy pewne ograniczenia. Umożliwią nam one automatyczną konstrukcję parsera.

Wszystkie ograniczenia nałożone na gramatyki afiksowe będą obowiązywać również dla rozszerzonych gramatyk afiksowych z tym, że warunek dotyczący zbioru predykatów zastąpimy obecnie innym warunkiem.

Przyjmiemy najpierw pewną definicję pomocniczą:

Definicja 2.1

Napiszemy $p\alpha \underset{G}{\Rightarrow_L} p_1p_2..p_n\alpha$ jeżeli $p \longrightarrow p_1p_2...p_n$ jest produkcją gramatyki G otrzymaną z hiperprodukcji predykatowej.

Napiszemy $p \overset{*}{\underset{G}{\Rightarrow_L}} X$ i powiemy, że hiperpojęcie predykatowe p wyprowadza lewostronnie napis X w gramatyce G jeżeli istnieje ciąg $X_0,X_1,...,X_k$ o własnościach:

(1) $X_0 = X$, $X_k = X$

(2) dla każdego $i=1,2,...,k$ $X_{i-1} \underset{G}{\Rightarrow_L} X_i$

Załóżmy teraz, że $p \in Q$ jest dowolnym predykatem rozszerzonej gramatyki afiksowej

$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \delta)$ oraz, że $i_1, i_2, ..., i_k$ są numerami wszystkich dziedziczonych afikspozycji p.

Weźmy pod uwagę dowolny, ale ustalony ciąg napisów $A_{i_1}, A_{i_2}, ..., A_{i_k} \in A_T^*$ oraz napis X taki, że $p(W_1,...,W_{N_p}) \overset{*}{\underset{G}{\Rightarrow_L}} X$; gdzie $W_1,...,W_{N_p} \in A_T^*$ jest dowolnym ciągiem napisów o własności:

$W_{i_1} = A_{i_1}$, $W_{i_2} = A_{i_2}, ..., W_{i_k} = A_{i_k}$

Przyporządkujmy następnie konkretnemu wyprowadzeniu $p(W_1,...,W_{N_p}) \overset{*}{\underset{G}{\Rightarrow_L}} X$ liczbę zastosowanych w nim produkcji predykatowych gramatyk G.

Zwróćmy uwagę, że przy ustalonym predykacie p i ustalonym ciągu napisów $A_{i_1}, A_{i_2}, ..., A_{i_k}$ owa liczba produkcji zależy zarówno od ciągu napisów $W_1,...,W_{N_p}$ i napisu X jak i od wyboru wyprowadzenia $p(W_1,...,W_{N_p}) \overset{*}{\underset{G}{\Rightarrow_L}} X$.

Niech teraz dla ustalonych p, $A_{i_1}, A_{i_2}, ..., A_{i_k}$ zbiór $PROD_L(p, A_{i_1}, A_{i_2}, ..., A_{i_k})$ będzie zbiorem wszystkich takich liczb. Powiemy wtedy, że predykat p jest predykatem skończonym jeżeli dla każdego ciągu $A_{i_1}, A_{i_2}, ..., A_{i_k}$ wśród liczb należących do zbioru $PROD_L(p, A_{i_1}, A_{i_2}, ..., A_{i_k})$ istnieje liczba największa.

Przyjmiemy następującą definicję:

Definicja 2.2

Powiemy, że zbiór hiperprodukcji predykatowych rozszerzonej gramatyki afiksowej $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \delta)$ jest dobrze zdefiniowany jeżeli spełnione są warunki:

(1) dla każdej hiperprodukcji predykatowej:

(*) $P_0(F_1^0,...,F_{N_{p_0}}^0) \longrightarrow P_1(F_1^1,...,F_{N_{p_1}}^1)...P_n(F_1^n,...,F_{N_{p_n}}^n)$

istnieje takie ustawienie:

(**) $P_{i_1}(F_1^{i_1},...,F_{N_{p_{i_1}}}^{i_1})P_{i_2}(F_1^{i_2},...,F_{N_{p_{i_2}}}^{i_2})...P_{i_n}(F_1^{i_n},...,F_{N_{p_{i_n}}}^{i_n})$ hiperpojęć występujących

po prawej stronie hiperprodukcji, że żadne z wystąpień dowolnej zmiennej afiksowej L w (* *) na pozycji syntetyzowanej nie może być poprzedzone wystąpieniem L w (* *) na pozycji dziedziczonej.

(2) każdy predykat $p \in Q$ jest predykatem skończonym.

(3) dla każdego predykatu $p \in Q$ oraz dla dowolnego ciągu $F_{i_1}, F_{i_2}, \ldots, F_{i_k}$ napisów ze zbioru A_T^* takiego, że $i_1, i_2, \ldots, i_k$ są numerami wszystkich dziedziczonych afikspozycji p, istnieje co najwyżej jeden ciąg napisów $G_1, G_2, \ldots G_{N_p} \in A_T^*$ taki, że $G_{i_1} = F_{i_1}$, $G_{i_2} = F_{i_2}, \ldots, G_{i_k} F_{i_k}$ oraz $p(G_1, G_2, \ldots, G_{N_p}) \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$

Powyższa definicja pozwala nam interpretować intuicyjnie każdy predykat jako funkcję ogólnie rekurencyjną, której argumentami są napisy stojące na afikspozycjach dziedziczonych, wartościami - napisy stojące na afikspozycjach syntetyzowanych.

Z podanych wyżej trzech warunków jedynie warunek(1) jest rozstrzygalny. W praktyce jednak hiperprodukcje predykatowe używa się na ogół do opisu operacji przeglądania list - np. listy deklaracji zmiennych, czy listy deklaracji etykiet (por. opis języka PASCAL w pracy WATT [16], produkcje (p.61.1) -(p.62.3)). Hiperprodukcje predykatowe pisze się wtedy zazwyczaj w postaci takiej, że również sprawdzenie warunków (2) i (3) jest czynnością prostą.

Podsumowując:

Definicja 2.3

Rozszerzoną gramatykę afiksową $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \delta)$ nazwiemy dobrze zdefiniowaną, jeżeli:

(1) Jej produkcje afiksowe są dobrze zdefiniowane,

(2) Jej gramatyka podstawowa jest dobrze zdefiniowana,

(3) Zbiór hiperprodukcji P jest dobrze zdefiniowany (dotyczy to również hiperprodukcji predykatowych),

(4) Gramatyka G nie posiada pętli,

(5) Liczba afikspozycji aksjomatu δ wynosi zero, bądź wszystkie jego afikspozycje są syntetyzowane,

(6) Zbiór hiperprodukcji predykatowych P_Q jest dobrze zdefiniowany.

Gramatyka z przykładu 10 jest dobrze zdefiniowana.

Twierdzenie 2.2 (Dodatek B, część III, rozdz. 2)

Dla każdego języka ogólnie rekurencyjnego istnieje dobrze zdefiniowana rozszerzona gramatyka afiksowa, która go generuje.

Rozdział 7. PARSER

Ograniczymy się tutaj do nieformalnego omówienia działania procedury PREDYKAT (formalny opis tej procedury znajduje się w dodatku B). Poza tym parser dla rozszerzonych gramatyk afiksowych jest taki sam jak dla gramatyk afiksowych.

Przyjmijmy najpierw, że konstruktor, oprócz działań opisanych dla gramatyk afiksowych, zapisze prawą stronę każdej hiperprodukcji predykatowej:

$h = p_0(F_1^0,\ldots,F_{N_{p_0}}^0) \to p_1(F_1^1,\ldots,F_{N_{p_1}}^1)\ldots p_n(F_1^n,\ldots,F_{N_{p_n}}^n)$ w postaci:

$(**)\ p_{i_1}(F_1^{i_1},\ldots,F_{N_{p_{i_1}}}^{i_1})\ p_{i_2}(F_1^{i_2},\ldots,F_{N_{p_{i_2}}}^{i_2})\ldots p_{i_n}(F_1^{i_n},\ldots,F_{N_{p_{i_n}}}^{i_n})$

(gdzie $i_1,i_2,\ldots,i_n \in \{1,2,\ldots,n\}$)

takiej, że żadne z wystąpień dowolnej zmiennej afiksowej L w (**) na pozycji syntetyzowanej nie może być poprzedzone wystąpieniem L w (**) na pozycji dziedziczonej. Czynność ta jest zawsze wykonalna (patrz pkt (1) definicji 2.2).

Przypuśćmy teraz, że na wejściu procedury PREDYKAT pojawił się predykat X taki, że wszystkie jego afikspozycje dziedziczone mają wartości terminalne.

Poszukujemy pierwszej takiej hiperprodukcji h_1:

$h_1 = X(F_1,\ldots,F_{N_X}) \to f_1(F_1^1,\ldots,F_{N_{f_1}}^1)\ldots f_n(F_1^n,\ldots,F_{N_{f_n}}^n)$ że hiperprodukcja ta da się "zastosować" dla danych wartości wszystkich afikspozycji predykatu X - tzn. że jeżeli wartości tych afikspozycji wyniosą odpowiednio $W_1,W_2,\ldots,W_{N_X}$, to wykonalne są dodawania:

$D(F_1) + W_1,\ D(F_2) + W_2,\ldots,\ D(F_{N_X}) + W_{N_X}$

Jeżeli hiperprodukcja taka nie istnieje to sygnalizujemy błąd semantyczny i przerywamy pracę parsera.

W przeciwnym przypadku działania te $(D(F_1) + W_1,\ldots,D(F_{N_X}) + W_{N_X})$ wykonujemy. Opierając się na ich wynikach wyznaczamy wartości wszystkich zmiennych występujących po lewej stronie hiperprodukcji h (a więc dla każdej zmiennej L występującej po lewej stronie h_1 - np. $F_i = \alpha_i L \beta_i$ - obliczamy wartość funkcji $\Phi(W_i, D(F_i), D(\alpha_i))$.

Następnie wstawiamy te wartości w miejsce każdego wystąpienia odpowiednich zmiennych po prawej stronie hiperprodukcji h_1.

Ponieważ wszystkie afikspozycje dziedziczone X miały wartości terminalne, więc również wszystkie zmienne afiksowe występujące w h_1 na tych afikspozycjach otrzymają wartości terminalne. Zgodnie z dobrą definiowalnością zbioru hiperprodukcji P (pkt (3) definicji 2.3) wszystkie zmienne afiksowe występujące na pozycjach dziedziczonych predykatu f_1 w hiperprodukcji h_1 muszą występować również na pozycjach dziedziczonych predykatu X (przypominamy o działalności konstruktora dla hiperprodukcji predykatowych). Stąd: wszystkie afikspozycje dziedziczone predykatu f_1 otrzymają wartości terminalne.

Poszukujemy teraz takiej hiperprodukcji h_2

$h_2 = f_1(G_1,\ldots,G_{N_{f_1}}) \to p_1(G_1^1,\ldots,G_{N_{p_1}}^1)\ldots p_m(G_1^m,\ldots,G_{N_{p_m}}^m)$, że hiperprodukcja ta da się "zastosować" dla wartości wszystkich afikspozycji predykatu f_1 wyznaczonych w czasie analizy hiperprodukcji h_1.

Jeżeli hiperprodukcja taka nie istnieje, to poszukujemy innej niż h_1 hiperprodukcji, która zawiera X po lewej stronie.

Jeżeli istnieje - to dla h_2 wykonujemy czynności analogiczne jak dla hiperprodukcji h_1.

Teraz więc obliczamy wartości terminalne wszystkich afikspozycji dziedziczonych predykatu p_1.

Postępowanie takie będziemy następnie kontynuowali dla predykatu p_1, itd. Zgodnie z punktem (2) definicji 2.2 postępowanie to musi mieć koniec. Musimy wybrać w którymś kroku hiperprodukcję postaci:

$$f(\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_{N_f}) \rightarrow \varepsilon$$

Załóżmy, że już hiperprodukcja h_2 jest tej postaci, a więc:

$$h_2 = f_1(G_1,\ldots,G_{N_{f1}}) \rightarrow \varepsilon$$

Zgodnie z warunkiem dobrej definiowalności zbioru hiperprodukcji P, każda zmienna afiksowa występująca na pozycji syntetyzowanej predykatu f_1 w hiperprodukcji h_2 musi występować również w h_2 na pozycji dziedziczonej f_1. Podstawiając w miejsce każdego wystąpienia dowolnej zmiennej jej wartość - wyznaczoną na podstawie wystąpienia tej zmiennej na pozycji dziedziczonej - otrzymujemy wartości terminalne wszystkich afikspozycji predykatu f_1.

Przystępujemy następnie do powtórnej analizy hiperprodukcji h_1:

$$h_1 = X(F_1,\ldots,F_{N_X}) \rightarrow f_1(F^1_1,\ldots,F^1_{N_{f1}})\ f_2(F^2_1,\ldots,F^2_{N_{f2}})\ldots f_n(F^n_1,\ldots,F^n_{N_{fn}})$$

Zgodnie z warunkiem dobrej definiowalności zbioru hiperprodukcji P oraz w wyniku działalności konstruktora, wszystkie zmienne afiksowe występujące na pozycji dziedziczonej predykatu f_2 w hiperprodukcji h_1 muszą występować również (w h_1) albo na pozycji dziedziczonej predykatu X, albo na pozycji syntetyzowanej predykatu f_1. Stąd łatwo obliczamy wartości terminalne wszystkich afikspozycji dziedziczonych predykatu f_2.

Następnie wybieramy dowolną hiperprodukcję z f_2 po lewej stronie, która da się "zastosować" dla danych wartości afikspozycji predykatu f_2, itd.

Analogicznie postępując w dalszym ciągu, otrzymujemy w końcu terminalne wartości wszystkich afikspozycji wszystkich predykatów $f_1,f_2,\ldots,f_n$ występujących w hiperprodukcji h_1. Zgodnie z warunkiem dobrej definiowalności zbioru hiperprodukcji P wszystkie zmienne afiksowe występujące w hiperprodukcji h_1 na afikspozycji syntetyzowanej X muszą występować na afikspozycji syntetyzowanej w którymś z predykatów $f_1,f_2,\ldots,f_n$. Stąd otrzymujemy wartości terminalne wszystkich afikspozycji predykatu X.

Jedyność takiego ciągu wartości afikspozycji wynika natychmiast z warunku (3) definicji 2.2

Zwróćmy uwagę, że kolejność w jakiej są ustawione predykaty wy... ...jące po prawej stronie dowolnej hiperprodukcji predykatowej, nie ma żadnego wpływu n... język generowany przez daną gramatykę.

W pracy D.Watta (Watt [16]) znajduje się opis języka PASCAL za pomocą proponowanych przez niego dobrze sformułowanych rozszerzonych gramatyk afiksowych (well formed extended

affix grammars). Ponieważ każda dobrze sformułowana rozszerzona gramatyka afiksowa (Watt) jest dobrze zdefiniowaną rozszerzoną gramatyką afiksową (jak jest to rozumiane w niniejszej pracy), więc opis języka PASCAL podany przez Watta jest również "prawidłowy" w sensie zdefiniowanych przez nas gramatyk.

Metoda proponowana przez D.Watta nie pozwoliła mu jednak uniknąć, w omawianym opisie języka PASCAL, użycia dwóch predykatów opisanych za pomocą funkcji.

Metoda podana w niniejszej pracy umożliwia opis języka PASCAL za pomocą dobrze zdefiniowanej rozszerzonej gramatyki afiksowej, w której rolę wyżej wymienionych funkcji spełniają hiperprodukcje predykatowe. Podane one zostały w dodatku A (przykład 11).

DODATEK A

Część 1. GRAMATYKI AFIKSOWE

Przykład 1.

Podana niżej gramatyka afiksowa opisuje (generuje) prosty język programów o strukturze blokowej. Wszystkie bloki są ciągami zdań. Każde zdanie składa się z etykiety poprzedzającej instrukcję skoku do dowolnego zdania danego programu. Wystąpienie tej samej etykiety w więcej niż jednym zdaniu oraz skok do etykiety nie poprzedzającej żadnej instrukcji, są traktowane jako błąd.

$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ gdzie:

$V_N = \{$program, blok, ciąg-zdań, zdanie, etykieta$\}$, σ = program

$V_T = \{$begin, end, goto, 0, 1, ;, :$\}$

$A_N = \{$LISTA, ETYKIETA, WYNIK, CYFRA$\}$

$A_T = \{$0, 1, jest, nie-ma, ,$\}$

$Q = \{$sprawdź$\}$

R = {LISTA → LISTA, ETYKIETA, ETYKIETA → CYFRA, WYNIK → jest
LISTA → ε, CYFRA → 0, WYNIK → nie-ma}
ETYKIETA → CYFRA ETYKIETA, CYFRA → 1

$B = \{L, L_1, L_2, L_3, E_1, E_2, C_1, C_2, W\}$

$D = \{(L,$LISTA$), (L_1,$LISTA$), (L_2,$LISTA$), (L_3,$LISTA$), (E_1,$ETYKIETA$), (E_2,$ETYKIETA$), (C_1,$CYFRA$),$
$(C_2,$CYFRA$), (W,$WYNIK$)\}$

S = {(program, 0, ε, ε, ε), (blok, 3, [LISTA, LISTA, LISTA], [ι, ι, σ], ε),
(ciąg-zdań, 3, [LISTA, LISTA, LISTA], [ι, ι, σ], ε),
(zdanie, 3, [LISTA, LISTA, LISTA], [ι, ι, σ] ε) (etykieta, 1, ETYKIETA, σ, ε)
(sprawdź, 3, [LISTA, ETYKIETA, WYNIK], [ι, ι, σ], $F_{sprawdź}$)}

gdzie:

$$F_{sprawdź}(l,e,jest) = \begin{cases} \text{prawda} & \text{jeżeli } (\exists l_1, l_2)(l = l_1 e l_2) \\ \text{fałsz} & \text{jeżeli } \neg(\exists l_1, l_2)(l = l_1 e l_2) \end{cases}$$

$$[F_{sprawdź}(l,e,\text{nie-ma}) = \text{prawda}] \Leftrightarrow [F_{sprawdź}(l,e,\text{jest}) = \text{fałsz}]$$

P = {(1) program → blok (ε, L, L)
(2) blok (L_1, L, L_2) → begin ciąg-zdań (L_1, L, L_2) end
(3) ciąg-zdań (L_1, L, L_3) → zdanie (L_1, L, L_2) ; ciąg-zdań (L_2, L, L_3)
(4) ciąg-zdań (L_1, L, L_2) → zdanie (L_1, L, L_2)
(5) zdanie (L_1, L, L_1, E_1) → etykieta (E_1): goto etykieta $(E_2$
sprawdź$(L, E_2,$ jest) sprawdź $(L_1, E_1,$ nie-ma$)$
(6) zdanie (L_1, L, L_2) → blok (L_1, L, L_2)
(7) etykieta $(0E_1)$ → 0 etykieta (E_1)
(8) etykieta $(1E_1)$ → 1 etykieta (E_1)

(9) etykieta (0) → 0

(10) etykieta (1) → 1]

Pokażemy jak z hiperprodukcji (5) możemy otrzymać jedną z produkcji gramatyki G.

Wstawmy w miejsce zmiennej L_1 napis, 0, w miejsce L napis ,0,01,1 w miejsce E_1 napis 01, w miejsce E_2 napis 1. Otrzymamy wtedy:

zdanie (,0,,0,01,1,0,01) → etykieta (01): goto etykieta (1)

sprawdź (,0,01,1, 1,jest) sprawdź (,0,01,nie-ma)

Z hiperprodukcji (7) możemy otrzymać (wstawiając w miejsce E_1 napis 1) produkcję:

etykieta (01) → 0 etykieta (1).

Ponieważ $F_{\text{sprawdź}}$ (,0,01,1,1,jest) = prawda, więc sprawdź (,0,01,1,1,jest) → ε jest produkcją gramatyki G. Podobnie wykazujemy, że sprawdź (,0,01,nie-ma) → ε też jest produkcją gramatyki G. Ostatecznie więc:

zdanie (,0,,0,01,1,,0,01) $\overset{*}{\underset{G}{\Rightarrow}}$ 01 : goto 1

Przykład 2

Załóżmy, że gramatyka afiksowa G jest taka jak w przykładzie 1. Wtedy gramatyka podstawowa tej gramatyki ma postać:

$G_p = (V_N, V_T, Z, 6)$, gdzie:

V_N = { program, blok, ciąg-zdań, zdanie, etykieta }, 6 = program

V_T = { begin, end, goto, 0, 1, ;, : }

Z = { (1) program → blok

(2) blok → begin ciąg-zdań end

(3) ciąg-zdań → zdanie; ciąg-zdań

(4) ciąg-zdań → zdanie

(5) zdanie → etykieta : goto etykieta

(6) zdanie → blok

(7) etykieta → 0 etykieta

(8) etykieta → 1 etykieta

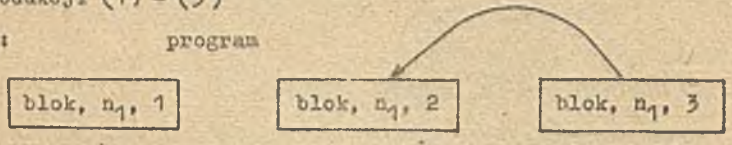
(9) etykieta → 0

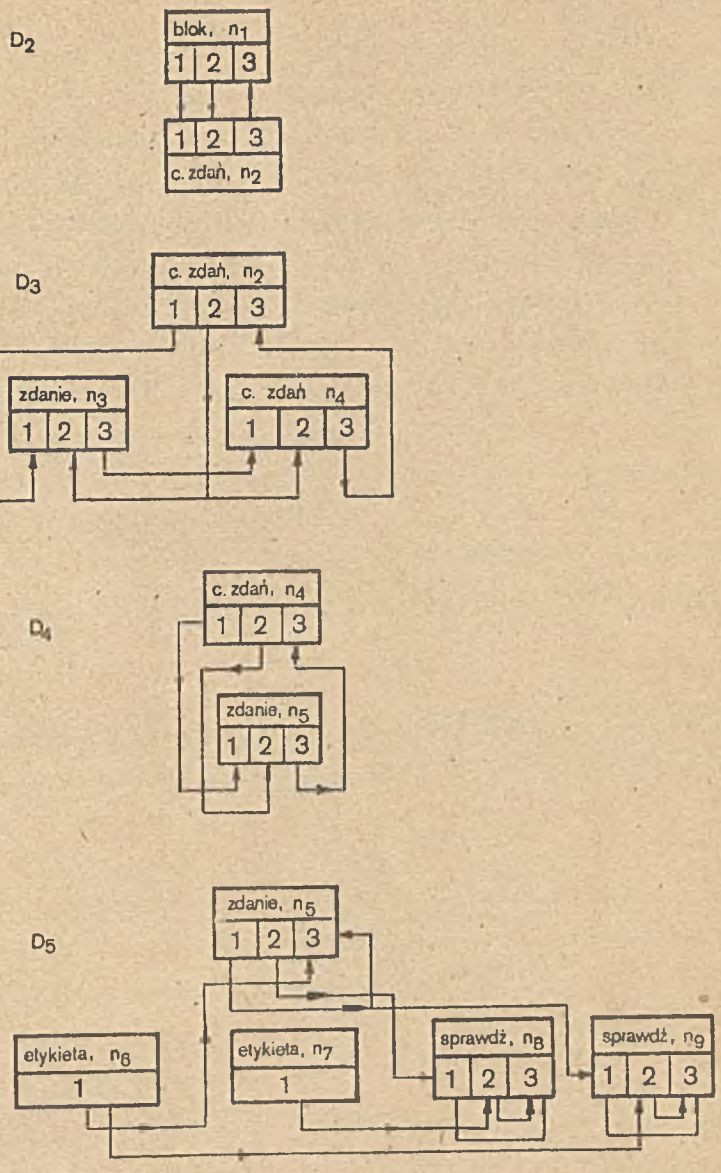
(10) etykieta → 1 }

Przykład 3

Załóżmy, że gramatyka G jest taka jak w przykładzie 1. Pokażemy jak wyglądają grafy hiperprodukcji (1) - (5)

D_1: program





Przykład 4

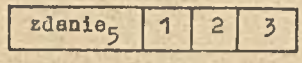
Przyjmijmy, że gramatyka afiksowa G jest postaci podanej w przykładzie 1. Poniższy graf zorientowany jest grafem wywodu słowa:

```
begin
   01:goto 0;
   begin
      0 : goto 01;
   end
end
```

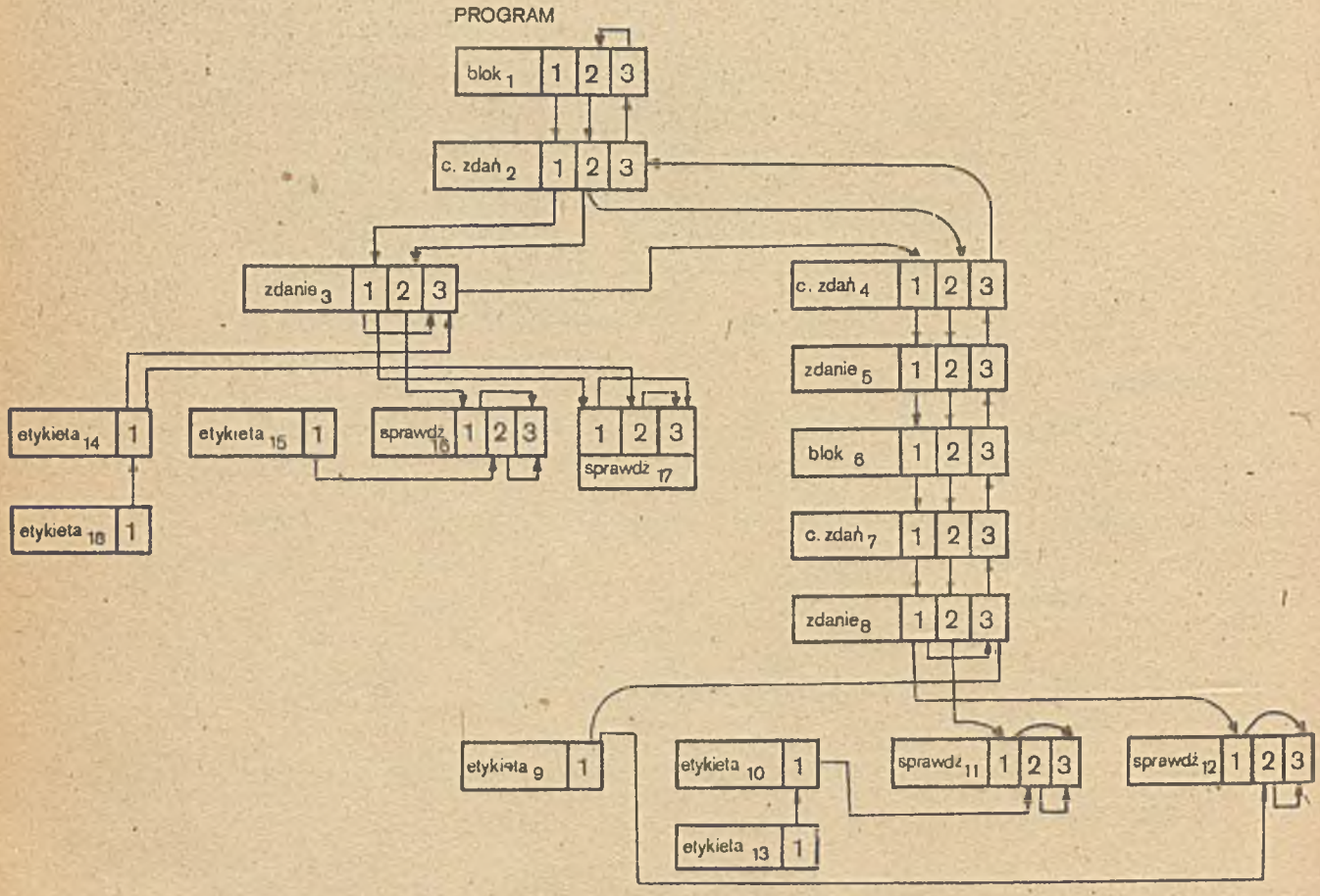
Słowo to ma następujący wywód w gramatyce podstawowej: (podajemy tutaj ciąg zastosowanych produkcji)

1. program → blok (1)
2. blok → begin ciąg-zdań end (2)
3. ciąg-zdań → zdanie; ciąg-zdań (3)
4. ciąg-zdań → zdanie (4)
5. zdanie → blok (6)
6. blok → begin ciąg-zdań end (2)
7. ciąg-zdań → zdanie (4)
8. zdanie → etykieta : goto etykieta (5)
9. etykieta → 0 etykieta (7)
10. etykieta → 1 (10)
11. etykieta → 0 (9)
12. zdanie → etykieta : goto etykieta (5)
13. etykieta → 0 (9)
14. etykieta → 0 etykieta (7)
15. etykieta → 1 (10)

Uwaga:

Zapis 

będzie oznaczał trzy węzły: (zdanie,5,1), (zdanie,5,2), (zdanie,5,3).



Przykład 5

Przebieg pierwszy parsera uruchomionego dla gramatyki i słowa z przykładu 1 poda na wyjściu graf podany w przykładzie 4. Węzły tego grafu będą miały następujące wartości:

WARTOŚĆ (blok,1,1)= ε, WARTOŚĆ(blok,1,2)= LISTA, WARTOŚĆ(blok,1,3)= LISTA

WARTOŚĆ(c-zdań,2,1) = LISTA, WARTOŚĆ(c-zdań,2,2)= LISTA, WARTOŚĆ(c-zdań,2,3)= LISTA

WARTOŚĆ(zdanie,3,1)= LISTA, WARTOŚĆ(zdanie,3,2)= LISTA, WARTOŚĆ(zdanie,3,3)= LISTA, ETYKIETA

WARTOŚĆ(c-zdań,4,1)= LISTA, WARTOŚĆ(c-zdań,4,2)= LISTA, WARTOŚĆ(c-zdań,4,3)= LISTA

WARTOŚĆ(zdanie,5,1)= LISTA, WARTOŚĆ(zdanie,5,2)= LISTA, WARTOŚĆ(zdanie,5,3)= LISTA

WARTOŚĆ(blok,6,1)= LISTA, WARTOŚĆ(blok,6,2)= LISTA, WARTOŚĆ(blok,6,3)= LISTA

WARTOŚĆ(c-zdań,7,1)= LISTA, WARTOŚĆ(c-zdań,7,2)= LISTA, WARTOŚĆ(c-zdań,7,3)= LISTA

WARTOŚĆ(zdanie,8,1)= LISTA, WARTOŚĆ(zdanie,8,2)= LISTA, WARTOŚĆ(zdanie,8,3)= LISTA, ETYKIETA

WARTOŚĆ(etykieta,9,1)= 0

WARTOŚĆ(etykieta,10,1)= OETYKIETA

WARTOŚĆ(sprawdź,11,1)= LISTA, WARTOŚĆ(sprawdź11,2)= ETYKIETA, WARTOŚĆ(sprawdź,11,3)= jest

WARTOŚĆ(sprawdź,12,1)= LISTA, WARTOŚĆ(sprawdź,12,2)= ETYKIETA, WARTOŚĆ(sprawdź,12,3)= nie-ma

WARTOŚĆ(etykieta,13,1)= 1

WARTOŚĆ(etykieta,14,1)= 0

WARTOŚĆ(sprawdź,16,1)= LISTA, WARTOŚĆ(sprawdź,16,2)= ETYKIETA, WARTOŚĆ(sprawdź,16,3)= jest

WARTOŚĆ(sprawdź,17,1)= LISTA, WARTOŚĆ(sprawdź,17,2)= ETYKIETA, WARTOŚĆ(sprawdź,17,3)= nie-ma

WARTOŚĆ(etykieta,18,1)= 1

Stos STOS będzie miał postać:

STOS = $\{$ (etykieta,13,1),(etykieta,9,1),(etykieta,15,1),(etykieta,18,1)$]$

Pomijamy tutaj węzły (sprawdź,11,3), (sprawdź,12,3),(sprawdź,16,3) i (sprawdź,17,3). Węzły te nie blokują bowiem żadnych innych węzłów.

Przykład 6

Prześledzimy tylko część czynności przebiegu drugiego:

Krok 1

STOS $\neq \phi$ wykonamy więc krok 2

Krok 2

etykieta $\in V_N$

Krok 3

od węzła (etykieta,13,1) prowadzi strzałka do węzła (etykieta,10,1)

Krok 4

AKTUAL $[$(etykieta,13,1), (etykieta,10,1)$]$;

Teraz WARTOŚĆ (etykieta,10,1) = 01

WYMAŻ $[$(etykieta,13,1), (etykieta,10,1)$]$;

Ponieważ wcześniej było: ILE(etykieta,10,1) = 1, więc teraz ILE(etykieta,10,1)= 0

Krok 5

STOS = {(etykieta,10,1),(etykieta,13,1),(etykieta,9,1),(etykieta,15,1) (etykieta,18,1)}

Krok 2

etykieta $\in V_N$

Krok 3

(Y, adr_1, j) = (sprawdź,11,2)

Krok 4

AKTUAL[(etykieta,10,1),(sprawdź,11,2)] skąd:

WARTOŚĆ(sprawdź,11,2) = 01

ILE(sprawdź,11,2) := 0; WYMAŻ[(etykieta,10,1),(sprawdź,11,2)]

Krok 5

STOS = {(sprawdź,11,2),(etykieta,10,1),(etykieta,13,1),.....}

Krok 2

sprawdź $\in Q$

Krok 7

ILE(sprawdź,11,3) = 2

Krok 9

WYMAŻ[(sprawdź,11,1), (sprawdź,11,3)];

ILE(sprawdź,11,3) := 1

STOS = {(etykieta,10,1),(etykieta,13,1),(etykieta,9,1),...}

Krok 1

STOS $\neq \phi$

Krok 2

etykieta $\in V_N$

Krok 3

od węzła (etykieta,10,1) nie prowadzi żadna strzałka,

Krok 6

STOS = {(etykieta,13,1),(etykieta,9,1),(etykieta,15,1),.....}

Krok 1

STOS $\neq \phi$

Krok 2

etykieta $\in V_N$

Krok 3

od węzła (etykieta,13,1) nie prowadzi żadna strzałka

Krok 6

STOS = {(etykieta,9,1), (etykieta,15,1),(etykieta,18,1)}

itd.,...

Przykład 7

Opiszemy działanie procedury AKTUAL[(etykieta,13,1),(etykieta,10,1)] uruchamianej podczas przebiegu drugiego opisanego w przykładzie 6. Odpowiednie blokowanie nastąpiło w hiperprodukcji (7):

(7) etykieta $(OE_1) \rightarrow$ 0 etykieta(E_1)

Przed wykonaniem procedury AKTUAL:

WARTOŚĆ(etykieta,13,1) = 1

WARTOŚĆ(etykieta,10,1) = 0 ETYKIETA

a więc:

L_1 := Φ(1,ETYKIETA,ε) = 1;

L_2 := Φ(OETYKIETA,OETYKIETA,0) = ETYKIETA;

L_3 := $L_1 + L_2$ = 1 + ETYKIETA = 1;

WARTOŚĆ(etykieta,10,1) := P(OETYKIETA,OETYKIETA,0,1) = 01

Przykład 8

Podana niżej gramatyka ze zmiennymi globalnymi jest równoważna gramatyce z przykładu 1

$G = (V_N', V_T, A_N, A_T, Q', R, B', D', GLOBAL, S', P', \sigma)$ gdzie

$V_N, V_T, A_N, A_T, R, \sigma$ są zbiorami takimi jak w przykładzie 1

Q' = Q$\cup${global} gdzie Q takie jak w przykładzie 1

B' = B$\cup\{g\}$ gdzie B takie jak w przykładzie 1

D' = D$\cup\{g$,LISTA$\}$ gdzie D takie jak w przykładzie 1

GLOBAL = $\{g\}$

S' = {(program,1,LISTA,σ, nieokr.),

(blok,2,[LISTA,LISTA],[τ,σ],nieokr.),

(ciąg-zdań,2,[LISTA,LISTA],[τ,σ], nieokr.)

(zdanie,2,[LISTA,LISTA],[τ,σ], nieokr.),(etykieta,1,ETYKIETA,σ, nieokr.)

(sprawdź,3,[LISTA,ETYKIETA,WYNIK],[τ,τ,σ], $F_{sprawdź}$)

(global,2,[ETYKIETA,ETYKIETA],[τ,σ],Fglobal)

P' = {(1) program(L) $\rightarrow$ blok(ε,L) global(L,g)

(2) blok(L_1,L_2) $\rightarrow$ begin ciąg-zdań(L_1,L_2) end

(3) ciąg-zdań (L_1,L_3) $\rightarrow$ zdanie(L_1,L_2); ciąg-zdań(L_2,L_3)

(4) ciąg-zdań(L_1,L_2) → zdanie (L_1,L_2)

(5) zdanie(L_1,L_1,E_1) → etykieta(E_1): goto etykieta(E_2) sprawdź$(\varepsilon,E_2,$jest$)$
sprawdź $(L_1,E_1,$nie-ma$)$

(6) zdanie (L_1,L_2) → blok(L_1,L_2)

Hiperprodukcje (7) - (10) takie jak w przykładzie 1]

gdzie $F_{\text{sprawdź}}$ określono tak jak w przykładzie 1,

$F_{\text{global}}\ x,y$ = prawda $\langle\Rightarrow\rangle\ x = y$

Część druga. NIEZORIENTOWANE GRAMATYKI AFIKSOWE

Przykład 9

Niech $G = (V_N,V_T,A_N,A_T,Q,R,B,D,S,P,\sigma)$ będzie gramatyką afiksową taką jak w przykładzie 1. Wtedy niezorientowana gramatyka afiksowa $G' = (V_N,V_T,A_N,A_T,Q,R,B,D,S',P,\sigma)$ gdzie:
$S'=\{(X,N_X,A_N^{N_X},F_X) \mid$ istnieje T^{N_X} takie, że $(X,N_X,A_N^{N_X},T^{N_X},F_X) \in S\}$
jest dobrze zdefiniowaną jednoznaczną gramatyką NGA.

Część trzecia. ROZSZERZONE GRAMATYKI AFIKSOWE

Przykład 10

Niech $G = (V_N,V_T,A_N,A_T,Q,R,B,D,S,P,\sigma)$ będzie gramatyką afiksową z przykładu 1. Wtedy układ $G'' = (V_N,V_T,A_N,A_T,Q'',R,B,S'',P'',\sigma)$
gdzie:
Q'' = {sprawdź, inna}
S'' = {(program,0,nieokr,nieokr),(blok,3,[LISTA,LISTA,LISTA],[τ,τ,σ]),(ciąg-zdań,3,
[LISTA,LISTA,LISTA],[τ,τ,σ]),
(zdanie,3,[LISTA,LISTA,LISTA],[τ,τ,σ]),(etykieta,1,ETYKIETA,σ),
(sprawdź,3,[LISTA,ETYKIETA,WYNIK],[τ,τ,σ]),
(inna,2,[ETYKIETA,ETYKIETA],[τ,τ])}

P'' = $P \cup$ {(11) sprawdź(L,E_1,E_2,W) → inna (E_1,E_2) sprawdź(L,E_2,W)
(12) sprawdź$(L,E_1,E_1,$jest$)$ → ε
(13) sprawdź$(\varepsilon,E_1,$nie-ma$)$ → ε
(14) inna(C_1,E_1,C_2,E_2) → inna (C_1,C_2)
(15) inna (C_1E_1,C_1E_2) → inna (E_1,E_2)
(16) inna $(0,1)$ → ε
(17) inna $(1,0)$ → ε
(18) inna (C_1E_1,C_2) → ε
(19) inna (C_1,C_2E_1) → ε }

jest rozszerzoną gramatyką afiksową, generującą ten sam język co gramatyka z przykładu 1.

Przykład 11

W proponowanym przez D.Watta (Watt [16])opisie języka PASCAL występuje predykat: unequal-tag. Opowiednia piątka $S_{unequal-tag}$ ma postać:(unequal-tag,2,[TAG,TAG], [ι,ι], λt λu (t≠u)).

Jest to więc predykat sprawdzający dla dwóch napisów TAG_1,TAG_2 czy są one różne.

Predykat ten występuje m.in. w hiperprodukcji (p.62.2) podanej w cytowanej pracy Watta.

Odpowiednia gramatyka $G_{TAG} = (A_N, A_T, R, TAG)$ ma postać:

(podajemy tutaj tylko "istotne" produkcje afiksowe)

TAG → ALPHA

TAG → TAG ALPHA

ALPHA → a ALPHA → b ..... ALPHA → z ALPHA → 0 ..... ALPHA → 9

Poniższe hiperprodukcje opisują działanie predykatu unequal-tag zgodnie z postacią piątki $S_{unequal-tag}$.

(1) unequal-tag (T_1A_1, T_2A_2) → unequal-tag (A_1, A_2)

(2) unequal-tag (T_1A_1, T_2A_1) → unequal-tag (T_1, T_2)

(3) unequal-tag (T_1A_1, A_2) → ε

(4) unequal-tag (A_1, T_1A_2) → ε

(5) unequal-tag (A_1, A_2) → search $(A_1, A_2, abcd...yz01...9)$

(6) search (A_1, A_2, T_1A_1) → further (A_2, T_1)

(7) search (A_1, A_2, T_1A_2) → further (A_1, T_1)

(8) search (A_1, A_2, T_1A_3) → search (A_1, A_2, T_1)

(9) further (A_1, T_1A_1) → ε

(10) further (A_1, A_1) → ε

(11) further (A_1, T_1A_2) → further (A_1, T_1)

gdzie $D(A_1)$ = ALPHA, $D(A_2)$ = ALPHA, $D(T_1)$ = TAG, $D(T_2)$ = TAG

oraz S_{search} =(search,3, [ALPHA,ALPHA,TAG],[ι,ι,ι])

$S_{further}$=(further,2, [ALPHA,TAG],[ι,ι])

Drugim predykatem opisanym za pomocą funkcji jest unequal-value (występuje on m.in. w hiperprodukcji (p.69.2)).

Odpowiednia piątka $S_{unequal-value}$ ma postać:

(unequal-value,2, [VALUE,VALUE],[ι,ι], λv λw(v≠w))

Gramatyka G_{VALUE} = $(A_N, A_T, R, VALUE)$ ma postać:

VALUE → N N → ε

VALUE → minus1N N → 1N

Działanie tego predykatu opisujemy za pomocą hiperprodukcji predykatowych podobnie jak predykatu unequal-tag.

DODATEK B

Część pierwsza. GRAMATYKI AFIKSOWE

Twierdzenie 1

Dla każdego języka rekurencyjnie przeliczalnego istnieje gramatyka afiksowa, która go generuje.

Dowód:

Załóżmy, że L jest dowolnym językiem rekurencyjnie przeliczalnym nad alfabetem $V = \{a_1, a_2, \ldots, a_n\}$. Przyjmijmy, że funkcja rekurencyjnie przeliczalna f :

$$f(x) = \begin{cases} \text{true} & \text{gdy } x \in L \\ \text{false} & \text{gdy } x \notin L \end{cases}$$

określona na zbiorze V^* jest funkcją charakterystyczną zbioru L.

Niech teraz gramatyka afiksowa $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$ będzie postaci:

$V_N = \{\sigma, X\}$, $V_T = V$, $A_N = \{\text{NAPIS}, \text{SYMBOL}\}$, $A_T = V$, $Q = \{\text{sprawdź}\}$

$R = \{$ NAPIS $\rightarrow$ SYMBOL NAPIS

NAPIS $\rightarrow \varepsilon$

SYMBOL $\rightarrow a_1$ SYMBOL $\rightarrow a_2$ ..... SYMBOL $\rightarrow a_n\}$

$B = \{N, S\}$, $D(N) =$ NAPIS, $D(S) =$ SYMBOL

$S = \{(\sigma, 0, \varepsilon, \varepsilon, \varepsilon), (X, 1, \text{NAPIS}, \sigma, \text{nieokr.}).\ (\text{sprawdź}, 1, \text{NAPIS}, \tau, f)\}$

$P = \{\ \sigma \rightarrow X(N)$ sprawdź (N)

$X(a_1 N) \rightarrow a_1 X(N)$ $\quad X(a_1) \rightarrow a_1$

$X(a_2 N) \rightarrow a_2 X(N)$ $\quad X(a_2) \rightarrow a_2$

. .

. .

. .

$X(a_n N) \rightarrow a_n X(N)$ $\quad X(a_n) \rightarrow a_n\}$

Jak łatwo zauważyć $L(G) = L$

Rozdział 2

DOBRZE ZDEFINIOWANE GRAMATYKI AFIKSOWE

Podamy teraz formalną definicję grafu wywodu $X \overset{*}{\underset{G_p}{\Rightarrow}}_R Y$ dla gramatyki podstawowej gramatyki afiksowej $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$, takiej, że G_p jest dobrze zdefiniowana (przypominamy, że wtedy każdej produkcji podstawowej odpowiada dokładnie jedna hiperprodukcja). Korzystać tutaj będziemy z pojęć zdefiniowanych w części pierwszej (definicja 2.3.1 i 2.3.2).

Poniższa definicja określa pojęcie grafu w sposób rekurencyjny.

Definicja 2.3.3

Niech $X \overset{*}{\underset{G_p}{\Rightarrow}}_R Y$; gdzie $X \in V_N$, $Y \in (V_N \cup V_T)^*$.

(1) Wywód $X \overset{k}{\underset{G_p}{\Rightarrow}}_R Y$ ma długość 0 (tzn. nie zastosowano w nim żadnej produkcji). Grafem wywodu $X \overset{*}{\underset{G_p}{\Rightarrow}}_R Y$ nazwiemy w tym przypadku każdy z grafów GRAF(X,Y), którego węzłami są trójki $(X,n,1),(X,n,2),\ldots,(X,n,N_x)$ i który nie posiada strzałek. Każdy z węzłów (X,n,i) będziemy nazywać węzłem wejściowym (n jest dowolną liczbą naturalną, N_x jest liczbą afikspozycji symbolu X).

(2) $X \overset{+}{\underset{G_p}{\Rightarrow}}_R Y$

Załóżmy, że $X \to \alpha_0 X_1 \alpha_1 X_2 \alpha_2 \ldots X_k \alpha_k$ $(\alpha_i \in V_T^*,\ x_i \in V_N)$ jest produkcją podstawową gramatyki G.

Bez straty na ogólności możemy przyjąć, że produkcji tej odpowiada hiperprodukcja h postaci:

$$X(F_1,\ldots,F_{N_X}) \to \alpha_0 X_1(F_1^1,\ldots,F_{N_{X_1}}^1)\alpha_1 \ldots X_k(F_1^k,\ldots,F_{N_{X_k}}^k)\,\alpha_k X_{k+1}(F_1^{k+1},\ldots,F_{N_{X_{k+1}}}^{k+1})\ldots X_n(F_1^n,\ldots,F_{N_{X_n}}^n)$$

gdzie $X_{k+1},\ldots,X_m$ są predykatami występującymi w hiperprodukcji h.

Przyjmijmy, że węzły: $(X,n,1), (X,n,2),\ldots, (X,n,N_x)$,

$$(X_1,n_1,1), (X_1,n_1,2),\ldots,(X_1,n_1,N_{X_1}),\ldots$$
$$\ldots(X_k,n_k,1), (X_k,n_k,2),\ldots,(X_k,n_k,N_{X_k}),$$
$$(X_{k+1},n_{k+1},1),(X_{k+1},n_{k+1},2),\ldots, (X_{k+1},n_{k+1},N_{X_{k+1}})$$
$$\ldots(X_m,n_m,1), (X_m,n_m,2),\ldots, (X_m,n_m,N_{X_m})$$

są wszystkimi węzłami dowolnego (ale ustalonego) grafu D_h hiperprodukcji h.

Załóżmy dalej, że $Y = \alpha_0 Y_1 \alpha_1 Y_2 \alpha_2 \ldots Y_k \alpha_k$ gdzie $Y_1, Y_2, \ldots, Y_k \in (V_N \cup V_T)^*$ i $X \overset{*}{\underset{G_p}{\Rightarrow}}_R Y_1$, $X_2 \overset{*}{\underset{G_p}{\Rightarrow}}_R Y_2, \ldots, X_k \overset{*}{\underset{G_p}{\Rightarrow}}_R Y_k$

Niech dla każdego $i=1,2,\ldots,k$ G_i będzie dowolnym (ale ustalonym) grafem wywodu $X_i \overset{*}{\underset{G_p}{\Rightarrow}}_R Y_i$ takim, że węzły:

$(X_i,n_i,1),(X_i,n_i,2),\ldots,(X_i,n_i,N_{X_i})$ są węzłami wejściowymi tego grafu (a więc są odpowiednimi węzłami grafu D_h).

Jeżeli $i \neq j$ to niech grafy G_i i G_j będą takie, że nie mają wspólnych węzłów.

Korzystając z powyższych oznaczeń grafem wywodu $X \overset{*}{\underset{G_p}{\Rightarrow}}_R Y$ nazwiemy każdy z grafów zorientowanych GRAF(X,Y) spełniający warunki:

(a) Węzłami (wierzchołkami) grafu GRAF(X,Y) są wszystkie węzły grafów $D_h, G_1, G_2, \ldots, G_k$

(b) Od węzła (A,a,i) do węzła (B,b,j) grafu GRAF(X,Y) prowadzi strzałka wtedy i tylko wtedy gdy taka strzałka występuje w jednym z grafów $D_h, G_1, G_2, \ldots G_k$

Odpowiednie węzły $(X,n,1), (X,n,2),\ldots,(X,n,N_x)$ grafu D_h nazwiemy węzłami wejściowymi grafu GRAF(X,Y)

Jednoznaczność powyższej definicji wynika natychmiast z faktu, że gramatyka G_p jest dobrze zdefiniowana.

Algorytm znajdowania pętli w gramatyce afiksowej

Będziemy tutaj rozważać dowolną gramatykę afiksową G

$$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \sigma)$$

taką, że jej gramatyka podstawowa jest dobrze zdefiniowana.

W wyżej podanej definicji przyporządkowaliśmy każdemu wywodowi $\sigma \overset{*}{\underset{G_P}{\Rightarrow}}_R x$ graf zorientowany. Węzłami tego grafu były trójki uporządkowane (x,n,i). Teraz wygodniej będzie przyjąć, że węzły rozważanych przez nas grafów są dwójkami postaci: (X,i).

Definicja

Załóżmy, że $p = x_o \to \alpha_o X_1 \alpha_1 X_2 \alpha_2 \ldots X_n \alpha_n$ (gdzie $\alpha_i \in V_T^*$, $X_i \in V_N$) jest produkcją podstawową gramatyki G.

Nie zmniejszając ogólności rozważań, możemy przyjąć, że tej produkcji odpowiada hiperprodukcja h postaci:

$$h = X_o(F_1^o, \ldots, F_{N_{X_o}}^o) \to \alpha_o X_1(F_1^1, \ldots, F_{N_{X_1}}^1)\alpha_1 \ldots X_n(F_1^n, \ldots, F_{N_{X_n}}^n)\alpha_n \ldots X_m(F_1^m, \ldots, F_{N_{X_m}}^m)$$

gdzie $X_{n+1}, X_{n+2}, \ldots, X_m \in Q$

Przyporządkujmy hiperprodukcji h graf zorientowany GRAF(h), którego wierzchołkami są pary:

$$(X_o,1),(X_o,2),\ldots,(X_o,N_{X_o}),$$
$$(X_1,1),(X_1,2),\ldots,(X_1,N_{X_1}) \quad \ldots$$
$$\ldots,(X_n,1),(X_n,2),\ldots,(X_n,N_{X_n}) \quad \ldots$$
$$\ldots,(X_m,1),(X_m,2),\ldots,(X_m,N_{X_m}).$$

Od węzła (X_i,j) do węzła (X_k,l) poprowadzimy strzałkę w grafie GRAF (h) wtedy i tylko wtedy gdy j-ta afikspozycja X_i blokuje w hiperprodukcji h l-tą afikspozycję X_k.

Załóżmy, że dowolnej (ale ustalonej) produkcji podstawowej p (postaci takiej jak wyżej) odpowiada hiperprodukcja h.

Jeżeli $G_1, G_2, \ldots, G_n$ są grafami zorientowanymi rozpiętymi na wierzchołkach $(X_j,1),(X_j,2),\ldots,(X_j,N_{X_j})$ (j=1,2,...,n) to przez $GRAF(h)[G_1,G_2,\ldots,G_n]$ będziemy oznaczać graf zorientowany powstały z grafu GRAF(h) w następujący sposób:

Wierzchołkami grafu $GRAF(h)[G_1,G_2,..,G_n]$ będą wszystkie wierzchołki grafu GRAF(h).

Strzałkę od węzła (X_i,j) do węzła (X_k,l) poprowadzimy w grafie $GRAF(h)[G_1,G_2,\ldots,G_n]$ jeżeli strzałka ta występuje w jednym z grafów GRAF(h), $G_1, G_2, \ldots, G_n$.

Korzystając z wprowadzonych wyżej pojęć, podamy algorytm sprawdzający czy dana gramatyka G ma pętlę.

Algorytm ten, dla każdego X ze zbioru V_N, będzie tworzyć zbiór grafów zorientowanych $\mathcal{G}(X)$, z których każdy jest rozpięty na węzłach $(X,1),(X,2),\ldots,(X,N_X)$; gdzie N_X jest liczbą afikspozycji X.

ALGORYTM

Krok 1

Dla każdego $X \in V_N$ wykonaj instrukcję:

$S(X) := \phi$; wykonaj krok 2;

Krok 2

Wybierz ze zbioru V_N dowolny element X_0, oraz ze zbioru P dowolną hiperprodukcję h taką, że odpowiada jej produkcja podstawowa p postaci:

$$p = X_0 \rightarrow \alpha_0 X_1 \alpha_1 X_2 \alpha_2 \ldots X_n \alpha_n \quad (\text{gdzie } \alpha_i \in V_T^*,\ X_i \in V_N)$$

Jeżeli istnieje j (j=1,2,...,n) takie, że $S(X_j) = \phi$ to wykonaj krok 3, wpp wykonaj krok 4.

Krok 3

Umieść w zbiorze $S(X_0)$ graf rozpięty na wierzchołkach $(X_0,1), (X_0,2),\ldots,(X_0,N_{X_0})$ Graf ten zawierać będzie strzałkę od (X_0,i) do (X_0,j) jeżeli w grafie GRAF(h) istnieje zorientowana droga od(X_0,i) do(X_0,j). Wykonaj krok 2.

Krok 4

Dla każdego j=1,2,...,n wybierz dowolny graf D_j' taki, że $D_j' \in S(X_j)$

Do zbioru $S(X_0)$ dołącz graf o wierzchołkach $(X_0,1),(X_0,2),\ldots,(X_0,N_x)$. Graf ten zawierać będzie strzałkę od(X_0,i) do(X_0,j) jeżeli w grafie GRAF(h)$[D_1', D_2', \ldots, D_n']$ istnieje zorientowana droga od (X_0,i) do (X_0,j).

Algorytm kończy pracę gdy nie można już dodać nowego grafu do żadnego ze zbiorów $S(X)$. Algorytm ten jest skończony ponieważ skończona jest liczba wszystkich możliwych grafów.

Twierdzenie 1

Jeżeli żaden z grafów GRAF(h)$[D_1', D_2', \ldots, D_n']$, opisanych w powyższym algorytmie, nie ma pętli to żaden z grafów wywodu żadnego słowa $x \in V_T^*$ nie ma pętli.

Dowód:

Zanim wykażemy prawdziwość twierdzenia udowodnimy lemat:

Lemat:

Załóżmy, że $\alpha X \beta \overset{*}{\underset{G_P^R}{\Rightarrow}} \alpha\gamma\beta$ gdzie $X \in V_N$; $\alpha, \beta, \gamma \in (V_N \cup V_T)^*$.

Załóżmy dalej, że $(X,adr_1,1)$, $(X,adr_1,2),\ldots,(X,adr_1,N_x)$ są węzłami wejściowymi grafu GRAF(X,Y) wywodu $X \overset{*}{\underset{G_P^R}{\Rightarrow}} Y$ (definicja 2.3.3)

Jeżeli istnieje droga zorientowana od węzła (X,adr_1,i) do węzła (X,adr_1,j) w grafie GRAF(X,Y) to S(X) zawiera strzałkę od (X,i) do (X,j).

[Mówiąc, że S(X) zawiera strzałkę od(X,i) do(X,j) mamy na myśli, że istnieje taka strzałka w którymś grafie należącym do S(X)]

Dowód:

Dowód przeprowadzimy przez instrukcję względem k-długość wywodu $X \overset{*}{\underset{G_p}{\Rightarrow}}_R Y$. (przypominamy, że G_p jest typu LR(k)).

Jeżeli k=1 to teza lematu jest oczywiście prawdziwa.

Przyjmijmy teraz, że $X \overset{*}{\underset{G_p}{\Rightarrow}}_R \alpha_1 X_1 \beta_1 \overset{*}{\underset{G_p}{\Rightarrow}}_R \alpha_1 X_1 \beta_2 \overset{*}{\underset{G_p}{\Rightarrow}}_R \alpha_1 Y_1 \beta_2 \overset{*}{\underset{G_p}{\Rightarrow}}_R \alpha_2 Y_1 \beta_2 = Y$;

gdzie: $\alpha_1 \overset{*}{\underset{G_p}{\Rightarrow}}_R \alpha_2$, $\beta_1 \overset{*}{\underset{G_p}{\Rightarrow}}_R \beta_2$, $X_1 \overset{*}{\underset{G_p}{\Rightarrow}}_R Y_1$ oraz $X_1 \in V_N$.

Niech produkcji $p = X \to \alpha_1 X_1 \beta_1$ odpowiada hiperprodukcja:

$h = X(F_1,\ldots,F_{N_X}) \to \alpha_1' X_1(G_1,\ldots,G_N) \beta_1'$

Jeżeli w grafie GRAF(X,Y) istnieje droga zorientowana od (X,adr_1,i) do (X,adr_1,j) to:

albo (A) powyższa droga istnieje w grafie D_h hiperprodukcji h.

Lemat jest wtedy oczywiście prawdziwy. (chodzi o graf D_h wg definicji 2.3.2) (rozdz.2.3)

albo (B) istnieją a $(1 \leq a \leq N_{X_1})$ i b $(1 \leq b \leq N_{X_1})$ o własnościach:

(1) istnieje droga zorientowana od węzła (X,adr_1,i) do węzła (X_1,adr_2,a) w grafie D_h

(2) istnieje droga zorientowana od węzła (X_1,adr_2a) do węzła (X_1,adr_2,b) w grafie GRAF(X_1,Y_1) wywodu $X_1 \overset{*}{\underset{G_p}{\Rightarrow}}_R Y_1$

(3) istnieje droga zorientowana od węzła (X_1,adr_2,b) do węzła (X,adr_1,j) w grafie D_h

(zwracamy uwagę, że symbol X_1 o własnościach takich jak wyżej musi istnieć dla każdego wywodu $X \overset{*}{\underset{G_p}{\Rightarrow}}_R Y$)

Na mocy założenia indukcyjnego $S(X_1)$ zawiera strzałkę od (X_1,a) do (X_1,b), a więc zgodnie z algorytmem tworzenia $S(X)$ teza lematu jest spełniona.

Przejdziemy teraz do dowodu twierdzenia

Na to aby graf GRAF wywodu $\sigma \overset{*}{\underset{G_p}{\Rightarrow}}_R x$ miał pętlę potrzeba i wystarcza aby istniały węzły (X,adr,i), (X,adr,j) tego grafu o własnościach:

(1) $\sigma \overset{*}{\underset{G_p}{\Rightarrow}}_R \alpha_1 X \beta \overset{*}{\underset{G_p}{\Rightarrow}}_R \alpha_1 Y \beta \overset{*}{\underset{G_p}{\Rightarrow}}_R \alpha Y \beta = x, \quad X \in V_N$

(2) W wywodzie $\sigma \overset{*}{\underset{G_p}{\Rightarrow}}_R x$ została zastosowana produkcja p taka, że:

jeżeli h jest hiperprodukcją odpowiadającą p, to istnieje droga zorientowana od węzła (X,adr,i) do węzła (X,adr,j) w grafie D_h hiperprodukcji h

(3) istnieje droga zorientowana od węzła (X,adr,j) do węzła (X,adr,i) w grafie GRAF(X,y) wywodu $X \overset{*}{\underset{G_p}{\Rightarrow}}_R y$.

Powyższej równoważności, ze względu na jej oczywistość nie będziemy dowodzić (pominęliśmy tutaj przypadek "pętli" w jakimś grafie hiperprodukcji. Teza twierdzenia jest wtedy oczywiście prawdziwa).

Twierdzenie wynika natychmiast z lematu oraz ze sposobu tworzenia grafów $GRAF(h)[D_1,D_2,\ldots,D_n]$.

Rozdział 3

PARSER

Twierdzenie 1

Jeżeli $G_a = (A_N, A_T, R, a)$ jest gramatyką typu LR(k) oraz M,N,F są formami zdaniowymi takimi, że istnieje $x \in (A_N \cup A_T)^*$ o własności: $a \overset{*}{\underset{G_a}{\Rightarrow}} F \overset{*}{\underset{G_a}{\Rightarrow}} x$, $a \overset{*}{\underset{G_a}{\Rightarrow}} M \overset{*}{\underset{G_a}{\Rightarrow}} x$, $a \overset{*}{\underset{G_a}{\Rightarrow}} N \overset{*}{\underset{G_a}{\Rightarrow}} x$, to istnieje dokładnie jedna forma zdaniowa Σ taka, że $\Sigma = M+N$ oraz:

(1) $\Sigma \overset{*}{\underset{G_a}{\Rightarrow}} x$

(2) Jeżeli $M \in A_T^*$ to $M+N = N+M = M$

(3) $M+N = N+M$

(4) $F+(M+N) = (F+M)+N$

Twierdzenie to pozostawimy bez dowodu. Wynika ono natychmiast z analogicznego twierdzenia udowodnionego w dodatku C (twierdzenie 3 z Dodatku C, rozdział 3)

Ustalmy we wszystkich dalszych rozważaniach dowolną dobrze zdefiniowaną gramatykę afiksową $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \delta)$

Twierdzenie 2

Po wykonaniu wszystkich czynności przebiegu pierwszego stos STOS(x) zawiera tylko takie węzły (X,adr,i), że WARTOŚĆ $(X,adr,i) \in A_T^*$.

Dowód:

Załóżmy, że (X,adr,i) jest dowolnym węzłem grafu wywodu słowa x w gramatyce podstawowej G_p takim, że ILE (X,adr,i) = 0..

Skoro (X,adr,i) jest węzłem tego grafu to w wywodzie $\delta \overset{*}{\underset{G_a}{\Rightarrow}} x$ musiały być zastosowane bezpośrednio po sobie produkcje p_1 i p_2 takie, że p_1 zawiera x po swojej prawej stronie, p_2 natomiast po lewej.

Weźmy pod uwagę hiperprodukcję h_2 odpowiadającą produkcji p_2 i załóżmy, że na i-tej afikspozycji X w hiperprodukcji h_1 występuje napis F_i. To z uwagi na to, że ILE (X,adr,i)= 0 możliwe są dwa przypadki:

(a) $F_i \in A_T^*$

Teza twierdzenia jest wtedy oczywiście prawdziwa (patrz punkt (2) dodatku B, cz.I, rozdz.3)

(b) $F_i \notin A_T^*$

Wtedy ponieważ ILE(X,adr,i) = 0 to w hiperprodukcji h_1 odpowiadającej produkcji p_1, musi na i-tej afikspozycji symbolu X występować napis G_i należący do zbioru A_T^*. Przebieg pierwszy wykona więc działanie:

WARTOŚĆ (X,adr,i) := $D(F_i)$ + $D(G_i)$ = $D(F_i)$ + G_i

(patrz podpunkt (a) punktu (B) z rozdz.3.2.4)

Zgodnie z twierdzeniem 1 punkt (2) wynik tego dodawania jest napisem terminalnym.

Twierdzenie 3

Jeżeli analizowane słowo x należy do języka L(G) to czynności opisane w podpunkcie (a) punktu (B) z rozdz.3.2.4 wykonają się dla każdego węzła (X,adr,i).

Dowód:

Załóżmy, że dla węzła (X,adr,i) zostały zastosowane dwie produkcje podstawowe p_1 i p_2. Niech odpowiadają im hiperprodukcje h_1 i h_2. Przyjmijmy dalej, że na i-tej afikspozycji x w hiperprodukcji h_1 występuje napis $F \in (A_N \cup A_T)^*$ oraz, że na i-tej afikspozycji X w hiperprodukcji h_2 występuje napis $G \in (A_N \cup A_T)^*$. Zgodnie z definicją tworzenia wyprowadzeń w gramatykach afiksowych musi istnieć napis $Y \in A_T^*$ taki, że:

$$D(G) \overset{*}{\underset{R}{\Rightarrow}} Y \quad \text{oraz} \quad D(F) \overset{*}{\underset{R}{\Rightarrow}} Y$$

Korzystając z definicji sumy form otrzymujemy tezę twierdzenia.

Twierdzenie 4

Jeżeli analizowane słowo x należy do języka L(G) oraz (X,adr,i), (Y,adr,j) są węzłami grafu GRAF wywodu $\sigma \overset{*}{\underset{GP}{\Rightarrow}} x$ takimi, że istnieje strzałka od (X,adr_1,i) do (Y,adr_2,j) w grafie GRAF (oznaczona numerem hiperprodukcji h) to procedura AKTUAL$[(X,adr_1,i), (Y,adr,j)]$ jest wykonalna.

Dowód:

Jeżeli wszystkie wystąpienia zmiennej L (w hiperprodukcji h) na i-tej afikspozycji X ij-tej afikspozycji Y są postaci:

$F_i = \alpha_1 L \beta_1 = \alpha_2 L \beta_2 = \ldots = \alpha_n L \beta_n$ -wystąpienia w X

$G_j = \gamma_1 L \delta_1 = \gamma_2 L \delta_2 = \ldots = \gamma_m L \delta_m$ -wystąpienia w Y

to musi istnieć takie $y \in A_T^*$, że $D(L) \overset{*}{\underset{R}{\Rightarrow}} y$ i y ma własności:

(1) Jeżeli węzłowi (X,adr_1,i) grafu GRAF odpowiada w wywodzie $\sigma \overset{*}{\underset{G}{\Rightarrow}} x$ napis $T_i \in A_T^*$ to istnieją $\alpha,\beta \in A_T^*$ takie, że $T_i = \alpha y \beta$

(2) Jeżeli węzłowi (Y,adr_2,j) grafu GRAF odpowiada w wywodzie $\sigma \overset{*}{\underset{G}{\Rightarrow}} x$ napis $R_j \in A_T^*$ to istnieją $\gamma,\delta \in A_T^*$ takie, że $R_j = \gamma y \delta$

(nie precyzujemy tutaj co formalnie oznacza zwrot "węzłowi (X,adr_1,i) grafu GRAF odpowiada w wywodzie $\sigma \overset{*}{\underset{G}{\Rightarrow}} x$ napis $T_i \in A_T^*$" zmusiłoby to bowiem do wprowadzenia dosyć skomplikowanego aparatu w celu opisu prostego intuicyjnie pojęcia).

Wynika stąd natychmiast, że istnieje takie $y \in A_T^*$, iż:

Φ WARTOŚĆ (X,adr_1,i), $D(F_1)$, $D(\alpha_k)) \overset{*}{\underset{R}{\Rightarrow}} y$ dla wszystkich k=1,2,...n

Φ WARTOŚĆ (Y,adr_2,j), $D(G_j)$, $D(\beta_1)) \overset{*}{\underset{R}{\Rightarrow}} y$ dla wszystkich l=1,2,...,m

Korzystając teraz z definicji dodawania form zdaniowych otrzymujemy tezę twierdzenia.

Zanim przejdziemy do dowodu twierdzenia 3.2.2.1 (patrz rozdz.3.2.2) będziemy musieli wprowadzić pewne pomocnicze pojęcie oraz udowodnić dwa lematy.

Definicja 1

Niech GRAF będzie dowolnym, ale ustalonym grafem wywodu $G \overset{*}{\underset{G_p R}{\Rightarrow}} x$, oraz niech (X,adr,i) będzie dowolnym węzłem tego grafu.

Węzłowi (X,adr,i) przyporządkujemy $\varsigma(X,adr,i)$ - element zbioru $N \cup \{\infty\}$ (gdzie N oznacza zbiór liczb naturalnych) określony w następujący sposób:

(1) $\varsigma(X,adr,i) = 0 \Leftrightarrow$ do węzła (X,adr,i) nie prowadzi żadna strzałka grafu GRAF

(2) $X,adr,i = n \Leftrightarrow n$ jest najmniejszą spośród liczb k o własności:

$\varsigma(Y,adr_1,j) < k$ dla wszystkich węzłów (Y,adr_1,j) takich, że od (Y,adr_1,j) prowadzi strzałka do węzła (X,adr,i) w grafie GRAF

(3) $\varsigma(X,adr,i) = \infty \Leftrightarrow \neg[(\exists n)(\varsigma(X,adr,i) = n)]$

Tak zdefiniowany element $\varsigma(X,adr,i)$ zbioru $N \cup \{\infty\}$ nazwiemy rzędem węzła (X,adr,i).

Lemat 1

Dla dobrze zdefiniowanej gramatyki G i dla dowolnego grafu wywodu GRAF w gramatyce podstawowej G_p, każdy węzeł tego grafu ma rząd skończony.

Dowód:

Na początku zauważmy, że zgodnie z założeniem, iż graf GRAF nie ma pętli (gramatyka G jest dobrze zdefiniowana) zbiór węzłów (X,adr,i) takich, że $\varsigma(X,adr,i) = 0$ jest niepusty. Niepusty jest więc również zbiór węzłów o rzędzie skończonym. Przypuśćmy teraz, że węzeł (Y,adr_1,j) grafu GRAF ma rząd nieskończony. Zgodnie z definicją 1 musi więc w GRAF istnieć węzeł (Z,adr_2,k) taki, że istnieje w GRAF strzałka od (Z,adr_2,k) do (Y,adr_1,j) oraz $\varsigma(Z,adr_2,k) = \infty$ (bo graf GRAF ma skończoną ilość węzłów). Tę cechę posiadają wszystkie węzły o rzędzie nieskończonym, a więc graf GRAF miałby pętlę.

Lemat 2

Niech graf GRAF będzie grafem wywodu $G \overset{*}{\underset{G_p \varrho}{\Rightarrow}} x$, zaś n dowolną (ale ustaloną) liczbą naturalną. Załóżmy dalej, że w czasie wykonywania przebiegu drugiego, wszystkie węzły (Y,adr_2,j) o własności: $\varsigma(Y,adr_2,j) < n$ otrzymają wartość terminalną.

Wtedy jeżeli $\varsigma(X,adr_1,i) = n$, to węzeł (X,adr,i) też otrzyma wartość terminalną (w wyniku realizacji przebiegu drugiego).

Dowód:

Zauważmy najpierw, że każdy węzeł (grafu GRAF) o wartości terminalnej znajdzie się, w trakcie wykonywania przebiegu drugiego, na wierzchołku stosu STOS(x). Z definicji rzędu węzła oraz lematu 1 wynika, że dla wszystkich węzłów (Y,adr_2,j) takich, że istnieje strzałka od węzła $(Y,adr_2 j)$ do węzła (X,adr_1,i), będzie: $\varsigma(Y,adr_2,j) < n$.

Zgodnie z założeniem lematu, każdy z owych węzłów (Y,adr_2,j) znajdzi się więc na wierzchołku stosu STOS(x) i wykonany zostanie dla niego ciąg procedur AKTUAL lub PREDYKAT.

Wykonaniu procedury PREDYKAT (Y,adr_2) (patrz rozdz.3.2.5) towarzyszy zawsze obliczenie wartości terminalnej węzła (X,adr_1,i) (tutaj X=Y, $adr_1 = adr_2$).

Jeżeli natomiast dla wszystkich węzłów blokujących węzeł (X,adr_1,i) wykonywana będzie procedura AKTUAL, to ponieważ wykonaniu tej procedury towarzyszy zawsze obliczenie wartości terminalnej jakiejś zmiennej występującej na i-tej afikspozycji X (chodzi o wykonanie procedur postaci AKTUAL $[(Y,adr_2,j),(X,adr_1,i)]$), a więc w końcu obliczona zostanie terminalna wartość węzła (X,adr_1,i). (obliczone zostaną wartości terminalne wszystkich zmiennych występujących na i-tej afikspozycji symbolu X o adresie adr_1).

Twierdzenie 5

Przebieg drugi jest algorytmem skończonym.

Po wykonaniu przebiegu drugiego wartością każdego węzła (X,adr,i), grafu otrzymanego w przebiegu pierwszym, jest napis M taki, że $M \in A_T^*$

Dowód:

Część pierwsza tego twierdzenia jest następstwem faktu, że rozważana gramatyka afiksowa nie posiada pętli.

Część druga wynika z zasady indukcji oraz z lematów 1 i 2 i twierdzenia 2.

Twierdzenie 6

Dla każdej produkcji podstawowej $p = X_0 \rightarrow \alpha_0 X_1 \alpha_1 \ldots X_k \alpha_k$ napis:

$X_0 (w_1^0,\ldots,w_{N_{X_0}}^0) \rightarrow \alpha_0 X_1 (w_1^1,\ldots,w_{N_{X_1}}^1)\, \alpha_1 \ldots X_k (w_1^k,\ldots,w_{N_{X_k}}^k)\alpha_k \ldots X_n (w_1^n,\ldots,w_{N_{X_n}}^n)$ przyporządkowany jej przez przebieg drugi, jest produkcją gramatyki G.

Dowód:

W wyniku działalności konstruktora (dopisanie predykatów equal $_{L_p}$), jeżeli w hiperprodukcji (odpowiadającej produkcji p) jest kilka wystąpień dowolnej zmiennej L to co najmniej jedno z nich jest stosowane (tzn. nie jest definiujące). Określone więc zostanie odpowiednie blokowanie (strzałka od wystąpienia definiującego do stosowanego), a co za tym idzie, zostaną wykonane odpowiednie procedury AKTUAL lub PREDYKAT. Korzystając z twierdzenia 5 oraz przeglądając czynności wykonywane przez procedury AKTUAL i PREDYKAT otrzymujemy na mocy twierdzenia 1 tezę dowodzonego twierdzenia.

Twierdzenia 3.2.2.3 (rozdz.3.2.2) i 3.2.2.4 (rozdz.3.2.2) pozostawimy (jako oczywiste) bez dowodu (twierdzenie 3.2.2.4 jest bezpośrednim następstwem udowodnionego wyżej twierdzenia 6 oraz twierdzenia 3.2.2.3).

Rozdział 4

ZMIENNE GLOBALNE

Definicja 4.1

Gramatyką afiksową ze zmiennymi globalnymi nazwiemy układ:

$G = (V_N, V_T, A_N, A_T, Q, R, B, D, GLOBAL, \Sigma, P, 6)$, gdzie

V_N, V_T, A_N, A_T mają znaczenie takie samo jak poczyniliśmy w definicji 1.1 w części pierwszej.

Q - jest skończonym zbiorem symboli nazywanych predykatami. Spośród elementów zbioru Q wyróżniamy n (n - liczba naturalna) elementów; będziemy je oznaczać $global_1, global_2, \ldots, global_n$.

R,B,D mają takie samo znaczenie jak podaliśmy w definicji 1.1 w części pierwszej.

GLOBAL jest n-elementowym podzbiorem zbioru B. Elementy zbioru GLOBAL będziemy oznaczać $g_1, g_2, \ldots, g_n$ i nazwiemy zmiennymi globalnymi.

S jest zbiorem piątek postaci takiej jak podaliśmy w definicji 1.1 w części pierwszej, przy czym:

(1) $S_\delta = (\delta n, [D(g_1), D(g_2), \ldots, D(g_n)], [\sigma, \sigma, \ldots, \sigma]$, nieokreślona)

(2) $S_{global_1} = (global_1, 2, [A_1, A_1], [\tau, \sigma], F_{global_1})$

$S_{global_2} = (global_2, 2)\ [A_2, A_2], [\tau, \sigma], F_{global_2})$

$\vdots$

$S_{global_n} = (global_n, 2, [A_n, A_n], [\tau, \sigma], F_{global_n})$

gdzie A_i $(i=1,2,\ldots,n)$ jest dziedziną i-tej afikspozycji δ,

oraz $F_{global_1}(x,y) =$ prawda $\Leftrightarrow y=x$ dla $x,y \in L_{a_1}$

$F_{global_2}(x,y) =$ prawda $\Leftrightarrow y=x$ dla $x,y \in L_{a_2}$

$\vdots$

$F_{global_n}(x,y) =$ prawda $\Leftrightarrow y=x$ dla $x,y \in L_{a_n}$

P jest zbiorem hiperprodukcji zdefiniowanym tak jak dla gramatyk afiksowych, przy czym każda hiperprodukcja taka, że δ występuje po jej lewej stronie musi być postaci:

$\delta(F_1, F_2, \ldots, F_n) \to \alpha\, global_1(F_1, g_1)\, global_2(F_2, g_2) \ldots global_n(F_n, g_n)$ gdzie

$\delta(F_1, F_2, \ldots, F_n) \to \alpha$ jest hiperprodukcją według definicji 1.1 z części pierwszej taką, że żadna ze zmiennych globalnych nie może występować w żadnym z napisów $F_1, F_2, \ldots, F_n$. (przypominamy, że każda zmienna globalna jest również zmienną afiksową, a więc może występować w dowolnej hiperprodukcji [z wyjątkiem, o którym napisaliśmy wyżej]).

Definicja 4.2

Powiemy, że napis $Y \xrightarrow{h_1, h_2, \ldots, h_n} Y_1 Y_2 \ldots Y_m$ jest produkcją gramatyki ze zmiennymi globalnymi $G = (V_N, V_T, A_N, A_T, Q, R, B, D, GLOBAL, S, P, \delta)$ (gdzie card(GLOBAL) = n) jeżeli:

albo (1) Istnieje hiperprodukcja $h = Z \to Z_1 Z_2 \ldots Z_m \in P$ taka, że para $Y \to Y_1 Y_2 \ldots Y_m$ powstała z tej hiperprodukcji w wyniku następującego postępowania: każde wystąpienie dowolnej zmiennej afiksowej b występującej w hiperprodukcji h zastępujemy dowolnym napisem $c \in A_T^*$ takim, że $D(b) \overset{*}{\Rightarrow} c$ (wszystkie wystąpienia jednej zmiennej zastępujemy tym samym napisem) przy czym każde (ewentualne) wystąpienie zmiennej globalnej g_1 zastępujemy napisem h_1, każde (ewentualne) wystąpienie zmiennej globalnej g_2 napisem h_2, itd.

albo (2) Para $Y \longrightarrow Y_1Y_2...Y_m$ jest postaci: $y(f_1,f_2,...,f_{N_y}) \longrightarrow \varepsilon$

gdzie: $y \in Q$: $f_1,f_2,...,f_{N_y} \in A_T^*$ oraz $F_y(f_1,f_2,...,f_{N_y}) =$ prawda.

Definicja 4.3

Napiszemy $\alpha Y \beta \xrightarrow[h_1,h_2,...,h_n]{G} \alpha Y_1Y_2...Y_m\beta$ jeżeli

$Y \xrightarrow[h_1,h_2,...,h_n]{} Y_1Y_2...Y_m$ jest produkcją gramatyki G.

Napiszemy $X \overset{*}{\underset{G}{\Longrightarrow}} Y$ i powiemy, że X wyprowadza Y w gramatyce G jeżeli istnieją ciągi $X_0,X_1,...,X_k$ i $h_1,h_2,...,h_n$ takie, że:

(1) $X_0 = X$

(2) $X_k = Y$

(3) dla każdego i=1,2,...,k jest $X_{i-1} \xrightarrow[h_1,h_2,...,h_n]{G} X_i$

Definicja 4.4

Zbiór $L(G) = \{x \in V_T^* \mid$ istnieje takie hiperpojęcie początkowe $\sigma(F_1,...,F_{N_\sigma})$, że $\sigma(F_1,...,F_{N_\sigma}) \overset{*}{\underset{G}{\Longrightarrow}} x\}$

Hiperpojęciu początkowemu przypisujemy znaczenie takie jak podaliśmy w definicji 1.4 z części pierwszej (Rozdz.1. Definicja)

Rozdział 5

PODSUMOWANIE

Twierdzenie 1

Dla każdego języka rekurencyjnego istnieje dobrze zdefiniowana gramatyka afiksowa, która go generuje.

Dowód:

Załóżmy, że L jest dowolnym językiem rekurencyjnym nad alfabetem $V = \{a_1,a_2,....,a_n\}$.

Przyjmijmy, że funkcja rekurencyjna f postaci:

$$f(x) = \begin{cases} \text{prawda} & \text{gdy } x \in L \\ \text{fałsz} & \text{gdy } x \notin L \end{cases}$$

określona na zbiorze V^* jest funkcją charakterystyczną zbioru L.

Wtedy gramatyka afiksowa określona analogicznie jak w dowodzie twierdzenia 1 (Dodatek B, część I) jest dobrze zdefiniowaną gramatyką afiksową i oczywiście generuje ona język L. (jedyna różnica polega na tym, że tam rozważana funkcja była funkcją rekurencyjnie przeliczalną).

Część druga. NIEZORIENTOWANE GRAMATYKI AFIKSOWE

Rozdział 2

DOBRZE ZDEFINIOWANE GRAMATYKI NGA

Twierdzenie 1

Dla każdego języka rekurencyjnie przeliczalnego istnieje generująca go jednoznaczna gramatyka NGA.

Dowód:

Załóżmy, że L jest dowolnym językiem rekurencyjnie przeliczalnym i że $G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \delta)$ jest gramatyką afiksową zdefiniowaną tak jak w dowodzie twierdzenia 1 (Dodatek B, cz.1). Wówczas gramatyka $G' = (V_N, V_T, A_N, A_T, Q, R, B, D, S', P, \delta)$ gdzie:

$S' = \{(X, N_X, A_N^{N_X}, F_X) \mid$ istnieje T^{N_X} takie, że $(X, N_X, A_N^{N_X}, T^{N_X}, F_X) \in S\}$

jest oczywiście jednoznaczną gramatyką afiksową oraz $L(G') = L$

Twierdzenie 2

Dla każdego języka ogólnie rekurencyjnego istnieje generująca go dobrze zdefiniowana jednoznaczna gramatyka NGA.

Dowód:

Twierdzenie to dowodzimy analogicznie jak powyższe twierdzenie opierając się tym razem na twierdzeniu 1 z Dodatku B, cz.I, rozdz.5

Rozdział 3

PARSER

W całym tym rozdziale obowiązywać będzie taka sama umowa jaką przyjęliśmy dla gramatyk afiksowych w Dodatku B, cz.I, rozdz.2.

Ustalmy dowolną dobrze zdefiniowaną niezorientowaną gramatykę afiksową

$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, \delta)$

Przebieg pierwszy

Twierdzenie 1

Jeżeli analizowane słowo x należy do języka L(G) to wszystkie czynności przebiegu pierwszego są wykonalne.

Dowód przeprowadzamy analogicznie jak dowody odpowiednich twierdzeń gramatyk afiksowych (patrz Dodatek B, część I, rozdz.3).

Lemat 1

Załóżmy, że hiperprodukcja h jest dowolną z hiperprodukcji analizowanych przez przebieg pierwszy. Przyjmijmy, że jest ona postaci:

$h = X_0(F^0_1,\ldots,F^0_{N_{X_0}}) \Rightarrow \alpha_0 X_1 (F^1_1,\ldots,F^1_{N_{X_1}}) \alpha_1 \ldots X_k (F^k_1,\ldots,F^k_{N_{X_k}}) \alpha_k \ldots X_n (F^n_1,\ldots,F^n_{N_{X_n}})$

gdzie $\alpha_0, \alpha_1, \ldots, \alpha_k \in V_T^*$; $X_0, X_1, \ldots, X_k \in V_N$; $X_{k+1}, \ldots, X_n \in Q$

Załóżmy dalej, że $F^t_p = \alpha_1 L \beta_1$, $F^u_r = \alpha_2 L \beta_2$ są dowolnymi wystąpieniami zmiennej L - po prawej stronie hiperprodukcji h, natomiast $F^w_s = \alpha_3 L \beta_3$ jest dowolnym wystąpieniem zmiennej L po lewej stronie hiperprodukcji h.

Wówczas, po wykonaniu wszystkich czynności przebiegu pierwszego:

(1) $\Phi(\text{WARTOŚĆ}(X_t, adr_t, p), D(F^t_p), D(\alpha_1)) = \Phi(\text{WARTOŚĆ}(X_u, adr_u, r), D(F^u_r), D(\alpha_2))$

(2) $\Phi(\text{WARTOŚĆ}(X_t, adr_t, p), D(F^t_p), D(\alpha_1)) \overset{*}{\underset{R}{\Rightarrow}} \Phi(\text{WARTOŚĆ}(X_w, adr_w, s), D(F^w_s), D(\alpha_3))$

gdzie $\text{WARTOŚĆ}(X_j, adr_j, i)$ dla j=t,u,w; i=p,r,s oznacza zawartość pamięci (o adresie adr_j) odpowiadającej symbolowi X_j występującemu w hiperprodukcji h.

Znaczenie Φ oraz znaku + jest takie samo jak podaliśmy w części pierwszej.

Dowód:

Po wykonaniu wszystkich czynności przebiegu pierwszego dla hiperprodukcji h, że specyfiki działań opisanych w punkcie (C) rozdziału 3.1. wynika równość:

$\Phi(\text{WARTOŚĆ}(X_t, adr_t, p)\ D(F^t_p), D(\alpha_1)) = \Phi(\text{WARTOŚĆ}(X_u, adr_u, r), D(F^u_r), D(\alpha_2)) = \Phi(\text{WARTOŚĆ}$
$(X_w, adr_w, s), D(F^w_s), D(\alpha_3))$

Ponieważ jedną z następnie analizowanych przez parser hiperprodukcji będzie taka hiperprodukcja h_1, że symbol X_w występuje po jej prawej stronie, więc dla formy zdaniowej $\text{WARTOŚĆ}(X_w, adr_w, s)$ będą jeszcze raz wykonywane działania opisane w rozdziale 3.1 (pierwszy raz były wykonywane w trakcie analizy hiperprodukcji h). Natomiast formy $\text{WARTOŚĆ}(X_t, adr_t, p)$ i $\text{WARTOŚĆ}(X_n, adr_n, r)$ w trakcie przebiegu pierwszego nie zmienią już wartości.

Teza lematu wynika z twierdzenia 1 (Dodatek B, cz.I, rozdz.3) oraz definicji sumy form.

Przebieg drugi

Twierdzenie 2

Jeżeli analizowane słowo x należy do języka L(G) to wszystkie czynności przebiegu drugiego są wykonywalne.

Dowód twierdzenia przeprowadzamy analogicznie jak dowód twierdzenia 4 dla gramatyk afiksowych (patrz Dodatek B, cz.I rozdz.3)

Lemat 2

Załóżmy, że hiperprodukcja h jest dowolną z hiperprodukcji analizowanych przez przebieg drugi. Przyjmijmy, że jest ona postaci podanej w Dodatku B, cz.II, rozdz.3, Lemat 1. Załóżmy, że $F^t_p = \alpha_1 L \beta_1$, $F^u_r = \alpha_2 L \beta_2$ są dowolnymi wystąpieniami zmiennej L w hiperprodukcji h. Wówczas po wykonaniu wszystkich czynności przebiegu drugiego:

Φ(WARTOŚĆ(X_t, adr_t, p), D(F_p^t), D(α_1)) = Φ(WARTOŚĆ X_u, adr_u, r), D(F_r^u), D(α_2))
gdzie adr_t i adr_u są adresami pamięci przydzielonej symbolom X_t i X_u.

Dowód:

Na początku zauważmy, że jeżeli $F_p^t = \alpha_1 L \beta_1$, $F_r^u = \alpha_2 L \beta_2$ są dowolnymi wystąpieniami zmiennej L po lewej stronie hiperprodukcji h, to teza lematu wynika natychmiast ze specyfiki działań wykonywanych podczas realizacji punktu (A) przebiegu drugiego (patrz rozdz.3.2).

Załóżmy teraz, że $F_p^t = \alpha_1 L \beta_1$ jest dowolnym wystąpieniem zmiennej L po lewej stronie hiperprodukcji h, oraz $F_r^u = \alpha_2 L \beta_2$ jest dowolnym wystąpieniem zmiennej L po prawej stronie hiperprodukcji h. Korzystając z wyżej udowodnionego lematu 1 otrzymujemy:
Φ(WARTOŚĆ(X_t, adr_t, p), D(F_p^t), D(α_1)) + Φ(WARTOŚĆ(X_u, adr_u, r), D(F_r^u), D(α_2)) = Φ(WARTOŚĆ (X_t, adr_t, p), D(F_p^t), D(α_1))
(wynika to z definicji sumy form zdaniowych).
Stąd, po wykonaniu wszystkich czynności związanych z realizacją punktu (A) dla hiperprodukcji h:
Φ(WARTOŚĆ(X_u adr_u, r), D(F_r^u), D(α_2)) := Φ(WARTOŚĆ(X_t, adr_t, p), D(F_p^t), D(α_1)) –
natomiast napis WARTOŚĆ(X_t, adr_t, p) nie zmienia swojej wartości (przypominamy, że X_t występuje po lewej stronie hiperprodukcji h). Teza lematu wynika natomiast z dowolności wyboru hiperprodukcji h. W przypadku gdy $F_p^t = \alpha_1 L \beta_1$, $F_r^u = \alpha_2 L \beta_2$ są wystąpieniami zmiennej L po prawej stronie hiperprodukcji h, lemat udowadniamy analogicznie jak wyżej (a więc wykazujemy, że w trakcie analizy dowolnej hiperprodukcji zmienie ulegają tylko wartości afiksowe pozycji symboli występujących po prawej stronie tej hiperprodukcji).

Twierdzenie 3

Po wykonaniu wszystkich czynności parsera każda z obliczonych wielkości WARTOŚĆ(X, adr, i) ma wartość terminalną.

Dowód:

W trakcie wykonywania czynności parsera każdej hiperprodukcji h (powstałej z produkcji p zastosowanej w wywodzie $S \underset{G_p}{\Rightarrow}_R x$) postaci:
$X_0(F_1^0, \ldots, F_{N_{X_0}}^0) \to \alpha_0 X_1(F_1^0, \ldots, F_{N_{X_1}}^1)\, \alpha_1 \ldots X_k(F_1^k, \ldots, F_{N_{X_k}}^k) \alpha_k \ldots X_n(F_1^n, \ldots, F_{N_{X_n}}^n)$ gdzie
$\alpha_0, \alpha_1, \ldots, \alpha_k \in V_T^*$; $X_0, X_1, \ldots, X_k \in V_N$; $X_{k+1}, \ldots, X_n \in Q$, został przyporządkowany napis:
$X_0(W_1^0, \ldots, W_{N_{X_0}}^0) \to \alpha_0 X_1(W_1^1, \ldots, W_{N_{X_1}}^1)\, \alpha_1 \ldots X_k(W_1^k, \ldots, W_{N_{X_k}}^k) \alpha_k \ldots$ taki, że dla wszystkich
i=0,1,...,k; j=1,2,...,N_X jest: W_j^i = WARTOŚĆ(X_i, adr_i, j).

Zgodnie z lematem 2 wszystkim wystąpieniom dowolnej zmiennej afiksowej L w hiperprojekcjach:
$X_0(F_1^0, \ldots, F_{N_{X_0}}^0), X_1(F_1^1, \ldots, F_{N_{X_1}}^1), \ldots, X_k(F_1^k, \ldots, F_{N_{X_k}}^k)$ odpowiadają w hiperp[illegible]lach:
$X_0(W_1, \ldots, W_{N_{X_0}}^0), X_1(W_1^1, \ldots, W_{N_{X_1}}^1), \ldots, X_k(W_1^k, \ldots, W_{N_{X_k}}^k)$ dokładnie te same wartości.
Gdyby więc którakolwiek z wielkości W_j^i nie otrzymała, po zakończeniu realizacji czynności przebiegu drugiego, wartości terminalnej to w gramatyce G_u (będącej uproszczoną postacią gra-

matyki G) istniałyby co najmniej dwa wywody prawostronne analizowanego słowa x. Otrzymalibyśmy więc sprzeczność z definicją jednoznaczności gramatyki G.

Przyjęliśmy tutaj, nie zmniejszając ogólności, założenie, że dla każdego $a \in A_N$ język $L(G_a)$ ma więcej niż jeden element, $(G_a = (A_N, A_T, R, a))$.

Część trzecia

ROZSZERZONE GRAMATYKI AFIKSOWE

Rozdział 2

Twierdzenie 1

Dla każdego języka rekurencyjnego istnieje generująca go dobrze zdefiniowana rozszerzona gramatyka afiksowa.

Dowód:

Załóżmy, że $L \in \Sigma^*$ jest dowolnym językiem rekurencyjnym. Zgodnie z definicją klasy języków rekurencyjnych (Hopcroft [8]), istnieje maszyna Turinga T $T = (K, \Sigma, \Gamma, \delta, q_0, F)$ o własnościach:

(1) $L(T) = L$

(2) dla każdego $w \in \Sigma^*$, dla każdego $q \in K$ maszyna T zatrzymuje się dla konfiguracji (q,w,i) (gdzie $0 \leq i \leq |w|$)

Rozważmy teraz rozszerzoną gramatykę afiksową

$G = (V_N, V_T, A_N, A_T, Q, R, B, D, S, P, 6)$ gdzie:

$V_N = \{6, X\}$, $V_T = \Sigma$, $A_N = \{\text{LEWA}, \text{PRAWA}, \text{SYMBOL}\}$, $A_T = \Sigma$

$R = \{$ LEWA $\rightarrow$ LEWA SYMBOL PRAWA $\rightarrow$ SYMBOL PRAWA

LEWA $\rightarrow \varepsilon$ PRAWA $\rightarrow \varepsilon$

SYMBOL $\rightarrow s$ dla każdego $s \in \Sigma$ $\}$

$B = \{L, B, S\}$, $D = \{(\text{LEWA}, L), (\text{PRAWA}, P), (\text{SYMBOL}, S)\}$

$Q = K \cup \{\text{sprawdź}\}$, $S = \{S_x\} \cup \{S_q \mid q \in K\} \cup \{S_{\text{sprawdź}}\} \cup \{S_6\}$

$S_x = (X, 1, \text{PRAWA}, \delta)$, $S_6 = (6, 0, \varepsilon, \varepsilon)$,

$S_q = (q, 2, [\text{LEWA}, \text{PRAWA}], [\tau, \tau])$ dla każdego $q \in K$

$S_{\text{sprawdź}} = (\text{sprawdź}, 1, \text{PRAWA}, \tau)$

$P = \{6 \rightarrow X(P)$ sprawdź $(P)\} \cup \{X(s\,P) \rightarrow s\,X(P) \mid s \in \Sigma\} \cup \{X(\sigma) \rightarrow \sigma \mid \sigma \in \Sigma\} \cup P_Q$

Zbiór P_Q (hiperprodukcji predykatowych) tworzymy następująco:

(1) dla każdej instrukcji postaci: $\delta(q_i, s_i) = (q_f, s_f, R)$ tworzymy hiperprodukcję:

$q_i(L, s_i P) \rightarrow q_f(L s_f, P)$

(2) dla każdej instrukcji postaci: $\delta(q_1,B) = (q_f,s_f,R)$ tworzymy hiperprodukcję:
$q_1(L,\varepsilon) \rightarrow q_f(Ls_f,\varepsilon)$ (B oznacza spację (blanck))

(3) dla każdej instrukcji postaci: $\delta(q_1,s_1) = (q_f,s_f,L)$ tworzymy hiperprodukcję:
$q_1(LS,s_1P) \rightarrow q_f(L,Ss_fP)$

(4) dla każdej instrukcji postaci: $\delta(q_1,B) = (q_f,s_f,L)$ tworzymy hiperprodukcję:
$q_1(LS,\varepsilon) \rightarrow q_f(L,Ss_f)$

(5) do tak utworzonego zbioru P_Q dodajemy jeszcze hiperprodukcje:
sprawdź$(P) \rightarrow q_o(\varepsilon,P)$, $q_f(L,\varepsilon) \rightarrow \varepsilon$ dla każdego $q_f \in F$

Pokażemy, że tak zdefiniowana rozszerzona gramatyka afiksowa spełnia dwa warunki:

(A) $L(G) = L(T) = L$

(B) G jest dobrze zdefiniowana

(A) Zauważmy, że dla każdej konfiguracji maszyny T postaci: $(q_o,w,1)$ gdzie $w \in \Sigma^*$ istnieje w gramatyce G wyprowadzenie:

$\delta \overset{*}{\underset{G}{\Rightarrow}} x(w)$ sprawdź$(w) \overset{*}{\underset{G}{\Rightarrow}} X(w)\, q_o\ (\varepsilon, w) \overset{*}{\underset{G}{\Rightarrow}} w\, q_o(\varepsilon,w)$

Załóżmy teraz, że $k = (q_1,s_1s_2\ldots s_j\ldots s_n,j)$ (gdzie $s_1s_2,\ldots,s_n \in \Sigma$) jest dowolną konfiguracją maszyny T i niech $I \equiv \delta(q_1,s_j) = (q_f,s_f,R)$ będzie instrukcją tej maszyny.

Instrukcji I odpowiada w gramatyce G dokładnie jedna hiperprodukcja h:

$h = q_1\ (L,s_jP) \rightarrow q_f\ (Ls_f,P)$

Podstawiając: $L := s_1s_2\ldots s_{j-1}$, $P := s_{j+1}\ldots s_n$ otrzymujemy z hiperprodukcji h produkcję
$p = q_1(s_1s_2\ldots s_{j-1},s_js_{j+1}\ldots s_n) \rightarrow q_f(s_1s_2\ldots s_{j-1}s_f,s_{j+1}\ldots s_n)$.

Przeglądając zbiór hiperprodukcji gramatyki G widzimy, że produkcję p możemy otrzymać jedynie z hiperprodukcji h. Stąd:

$(q_1,s_2s_2,\ldots s_n,j) \overset{*}{\underset{T}{\vdash}} (q_f,\ s_1s_2\ldots s_{j-1}s_fs_{j+1}\ldots s_n,j) \Longleftrightarrow$

$q_1(s_1s_2\ldots s_{j-1},s_js_{j+1}\ldots s_n) \overset{*}{\underset{G}{\Rightarrow}} q_f(s_1\ldots s_{j-1}s_f,s_{j+1}\ldots s_n)$

Korzystając z zasady indukcji otrzymujemy punkt (A).

(B) Ze względu na to, że pozostałe warunki dobrej definiowalności są spełnione przez gramatykę G w sposób oczywisty, wykażemy tylko, że zbiór hiperprodukcji predykatowych gramatyki G jest dobrze zdefiniowany.

Załóżmy więc, że istnieje nieskończona ilość napisów $q_f(G_1,G_2)$ takich, że dla ustalonego napisu $q_1(F_1,F_2)$ ma miejsce fakt:

$q_1(F_1,F_2) \overset{*}{\underset{G}{\Rightarrow}} q_f(G_1,G_2)$, gdzie $F_1,F_2,G_1,G_2 \in \Sigma^*$

Zgodnie z tym co udowodniliśmy wyżej, istnieje nieskończona ilość konfiguracji $(q_f,G_1\ G_2,j)$ takich, że dla ustalonej konfiguracji $(q_1,F_1\ F_2,k)$ ma miejsce fakt:
$(q_1,F_1\ F_2,k) \overset{*}{\underset{T}{\vdash}} (q_f,G_1\ G_2,j)$.

Wynika stąd, że maszyna T nie zatrzymuje się dla konfiguracji $(q_1F_1F_2\ldots)$ co jest sprzeczne z przyjętym założeniem

Przypuśćmy teraz, że wprawdzie dla każdego ustalonego napisu $q_1(F_1,F_2)$ istnieje skończona

liczba napisów $q_j(G_1,G_2)$ takich, że $q_1(F_1,F_2) \overset{+}{\Rightarrow}_G q_j(G_1,G_2)$, ale dla pewnego napisu $q_f(H_1,H_2)$ istnieje nieskończona ilość wyprowadzeń $q_i(F_1,F_2) \overset{*}{\Rightarrow}_G q_f(H_1,H_2)$.
Wobec przyjętego wyżej założenia, musi więc istnieć napis $q_1(J_1,J_2)$ taki, że

$$q_i(J_1,J_2) \overset{+}{\Rightarrow}_G q_1(J_1,J_2)$$

Rozumując tak jak poprzednio otrzymujemy:

$$(q_1,J_1J_2,k) \overset{+}{\models}_T (q_1,J_1J_2,k)$$

(relacja $\overset{+}{\models}$ oznacza, że konfigurację (q_1,J_1J_2,k) otrzymaliśmy z konfiguracji (q_1,J_1J_2,k) stosując co najmniej jedną instrukcję)

Wynika stąd natychmiast, że maszyna T nie zatrzymuje się dla konfiguracji (q_1,J_1J_2,k).

Rozdział 3

PARSER

W opisie parsera dla dowolnej dobrze zdefiniowanej gramatyki afiksowej ograniczymy się jedynie do omówienia procedury PREDYKAT. Poza tą procedurą parser rozszerzonej gramatyki afiksowej jest taki sam jak parser odpowiadającej jej gramatyki afiksowej. W mocy pozostają więc uwagi uczynione w Dodatku B, cz.I, rozdz.3.

Opis procedury PREDYKAT

Procedura ta analizować będzie hiperprodukcje predykatowe korzystając z różnych wystąpień tej samej zmiennej w danej hiperprodukcji.
Przyjmijmy, że na wejściu procedury PREDYKAT pojawił się predykat X taki, że wszystkie jego afikspozycje dziedziczone mają wartości terminalne.
W trakcie wykonywania procedury PREDYKAT (X,adr) będą konstruowane dwa stosy: STAN i RESULT. Stos STAN będzie stosem roboczym, natomiast stos RESULT określać będzie produkcje (powstałe z hiperprodukcji predykatowych) zastosowane dla obliczenia afikspozycji syntetyzowanych predykatu X.

(A) Każdemu elementowi stosu STAN - predykatowi f - zostanie przydzielona pamięć składająca się z czterech części:
Pierwsza część będzie zawierać N_f pozycji (N_f jest liczbą afikspozycji f) oznaczanych przez $W_1(f),W_2(f),\ldots,W_{N_f}(f)$. Będą tam zapisywane "wejściowe" wartości odpowiednich afikspozycji predykatu f.

Druga część pamięci przydzielonej predykatowi f też będzie zawierać N_f pozycji oznaczanych przez $A_1(f),\ldots,A_{N_f}(f)$. Będą to "aktualne" wartości afikspozycji f.

Trzecia część będzie stosem LEWA(f), na którym umieszczane będą numery tych hiperprodukcji, w których f występuje po lewej stronie.

Czwarta część, oznaczana WIEK(f), będzie liczbą 0, 1 lub 2.
0 informuje o tym, że nie była jeszcze stosowana żadna hiperprodukcja, w której f występuje po lewej stronie, 1 mówi nam, iż wybraliśmy hiperprodukcję taką, że możliwe jest zastosowa-

nie jej dla danych wartości afikspozycji f, 2 - że wybraliśmy hiperprodukcję, której nie można zastosować dla danych wartości afikspozycji predykatu f.

Każdy z elementów stosu STAN będziemy często utożsamiać z adresem pamięci, o której była wyżej mowa.

(B) Stos RESULT będzie zawierać adresy pamięci. Pamięć o danym adresie zawierać będzie numer pewnej hiperprodukcji predykatowej oraz adresy wszystkich predykatów występujących w tej hiperprodukcji.

Instrukcje podanego niżej algorytmu, określającego procedurę PREDYKAT, ujęliśmy w trzy grupy.

Postępowanie nasze będzie uzależnione od wartości zmiennej WIEK(f)dla wierzchołka f stosu STAN.

Pierwsza grupa instrukcji opisuje czynności wykonywane dla takiego wierzchołka f stosu STAN, że WIEK(f) = 0. Będzie to predykat, dla którego nie stosowaliśmy jeszcze żadnej hiperprodukcji (z f po lewej stronie). Zostanie wtedy zastosowana odpowiednia hiperprodukcja, a predykaty występujące po jej prawej stronie zostaną kolejno umieszczone na stosie STAN.

Druga grupa opisuje postępowanie dla predykatu f - wierzchołka stosu STAN - takiego, że wybraliśmy dla niego "złą" hiperprodukcję z f po lewej stronie. Nastąpi wtedy wybór innej hiperprodukcji (przypadek gdy WIEK(f) = 2). Jeżeli na wierzchołku stosu STAN znajdzie się predykat f, taki że była dla niego stosowana już jakaś hiperprodukcja i była to hiperprodukcja "dobra", (WIEK(f) = 1), to wtedy nastąpi "przekazanie" obliczonych przez predykat f wartości, a następnie zdjęcie predykatu f ze stosu STAN.

Szczególne znaczenie będą miały hiperprodukcje postaci $f(F_1,\ldots,F_{N_f}) \to \varepsilon$ analizowane dla wierzchołka f stosu STAN takiego, że WIEK(f) = 0.

W trakcie wykonywania procedury PREDYKAT używane będą następujące podprocedury:

(1) TA(r)

Procedura TA(r) mając na wejściu predykat f (występujący po lewej stronie hiperprodukcji r) oraz wartości terminalne odpowiednich $W_i(f)$ (tj. takich, że i-ta afikspozycja f jest dziedziczona), sprawdzi czy możliwe jest zastosowanie hiperprodukcji r dla wartości owych $W_i(f)$.

(2) OJCIEC(r)

Procedura OJCIEC może być uruchamiana tylko w przypadku gdy procedura TA(r) da odpowiedź pozytywną. Oblicza ona "aktualne" wartości $A_j(f)$ (f- lewa strona r). Jeżeli przy tym $r = f(F_1,\ldots,F_{N_f}) \to \varepsilon$ to wszystkie $A_j(f)$ otrzymają wartości terminalne.

(3) SYN(r,i)

Procedura SYN(r,i) może być uruchamiana tylko po wykonaniu odpowiedn procedury OJCIEC(r).

W trakcie wykonywania procedury SYN(r,i) dla hiperprodukcji $r = f(F_1,\ldots,F_{N_f}) \to f_1(F_1^1,\ldots,F_{N_{f_1}}^1)\ldots f_i(F_1^i,\ldots,F_{N_{f_i}}^i)\ldots f_n(F_1^k,\ldots,F_{N_{f_k}}^k)$ predykatowi f_i zostanie przydzielona pamięć oraz zostaną obliczone "wejściowe" wartości $W_j(f_i)$ $(j=1,2,\ldots,N)$ (na podstawie wartości odpowiednich $A_k(f)$).

Nastąpi umieszczenie na stosie LEWA(f_1) numerów odpowiednich hiperprodukcji.

(4) MOŻNA(r,f)

Dla hiperprodukcji r postaci:

$r = g(G_1,G_2,\ldots,G_{N_g}) \rightarrow f_1(F_1^1,\ldots,F_{N_{f_1}}^1)\ldots f(F_1,\ldots,F_{N_f})f_k(F_1^k,\ldots,F_{N_{f_k}}^k)\ldots f_n(F_1^n,\ldots,F_{N_{f_n}}^n)$

oraz predykatu f takiego, że $A_1(f),\ldots,A_{N_f}(f)$ mają wartości terminalne procedura ta sprawdza czy tym samym zmiennym afiksowym występującym w $F_1,\ldots,F_N$ odpowiadają te same wartości oraz czy możliwe jest "podstawienie" tych wartości w miejsce odpowiednich zmiennych występujących w $G_1,\ldots,G_N$, $F_1^k,\ldots,F_{N_{f_k}}^k,\ldots,F_1^n,\ldots,F_{N_{f_n}}^n$

(5) PRZEKAŻ(r,f)

Procedura ta może być uruchamiana tylko wtedy gdy odpowiednia procedura MOŻNA(r,f) da odpowiedź pozytywną. Używając oznaczeń takich jak w punkcie (4), PRZEKAŻ(r,f) "uaktualni" odpowiednie W_j dla predykatów $f_k,\ldots,f_n$ i A_i dla g ("przekaże" wartości obliczone dla f).

Wróćmy teraz do opisu procedury PREDYKAT(X,adr)

Wykonywać ona będzie operacje na stosach STAN i RESULT.

Początkowo na wierzchołku stosu STAN umieszczamy analizowany predykat X. Dla wszystkich $i=1,2,\ldots,N_X$; $W_i(X)$ i $A_i(X)$ mają początkowe wartości równe wartościom odpowiednich węzłów (X,adr,i). Stos LEWA(X)zawiera numery odpowiednich hiperprodukcji. Zmienna WIEK(X) ma wartość 0. Stos RESULT jest początkowo pusty.

W podanym niżej algorytmie przyjęto następujące oznaczenia:

Jeżeli S oznacza dowolny stos to przez TOP(S) oznaczymy element znajdujący się na wierzchołku stosu S, przez S+x. oznaczymy operację umieszczania na stosie S elementu x. Zapis ZDEJMIJ(S) będzie oznaczał zdjęcie ze stosu S elementu znajdującego się aktualnie na jego wierzchołku.

Zwrot "w przeciwnym przypadku" będziemy zastępowali zapisem "wpp".

ALGORYTM (procedura PREDYKAT (X,adr))

Krok 1

Przyjmijmy, że TOP (STAN) = f

Jeżeli WIEK(f) = 0 to wykonaj krok 2

Jeżeli WIEK(f) = 1 to wykonaj krok 11

Jeżeli WIEK(f) = 2 to wykonaj krok 7

Krok 2

Przyjmijmy, że TOP (LEWA(f)) = r

Wykonaj procedurę TA(r);

Jeżeli TA(r) daje odpowiedź pozytywną to wykonaj krok 3, wpp wykonaj krok 6

Wykonywane wtedy gdy WIEK (f) = 0

Krok 3

Wykonaj procedurę OJCIEC(r); WIEK(f) := 1; (przypominamy, że f = TOP(STAN))

Jeżeli r jest postaci $r = f(F_1,\ldots,F_{N_f}) \to \varepsilon$ to wykonaj krok 4, wpp wykonaj krok 5;

Krok 4

Przydziel hiperprodukcji r pamięć. W pamięci tej umieść adresy wszystkich wystąpujących w r predykatów (adresy n pierwszych (licząc od wierzchołka) elementów stosu STAN; gdzie n jest liczbą predykatów występujących w hiperprodukcji r). Umieść adres tej pamięci na stosie RESULT; wykonaj krok 1

Krok 5

Przyjmijmy, że r jest postaci: $r = f(F_1,\ldots,F_{N_f}) \to f_1(F_1^1, \ldots, F_{N_{f_1}}^1)\ldots f_n(F_1^n,\ldots,F_{N_{f_n}}^n)$

Wykonaj procedurę SYN(r,n); $WIEK(f_n) := 0$; $STAN := STAN + f_n$;

Wykonaj procedurę SYN(r,n-1); $WIEK(f_{n-1}) := 0$; $STAN := STAN + f_{n-1}$;

.
.
.

Wykonaj procedurę SYN(r,1); $WIEK(f_1) := 0$; $STAN := STAN + f_1$;

wykonaj krok 4

Krok 6

WIEK(f) := 2 ; wykonaj krok 7;

Wykonywane wtedy gdy WIEK (f) = 2

Krok 7

LEWA f := ZDEJMIJ (LEWA(f)) (przypominamy, że f = TOP (STAN))

Jeżeli LEWA(f) = ϕ to wykonaj krok 9, wpp wykonaj krok 8;

Krok 8

WIEK(f) := 0; wykonaj krok 1;

Krok 9

Jeżeli STAN $\neq \phi$ to wykonaj krok 10, wpp wykonaj krok 15;

Krok 10

STAN := ZDEJMIJ (STAN); RESULT := ZDEJMIJ(RESULT);

Jeżeli WIEK(TOP (STAN)) $\neq$ 0 to podstaw WIEK(TOP(STAN)) := 2 i wykonaj krok 7, wpp wykonaj jeszcze raz krok 10;

Wykonywane wtedy gdy WIEK (f) = 1

Krok 11

Jeżeli card(STAN) > 1 to wykonaj krok 12, wpp wykonaj krok 14;

(przyjmujemy tutaj, że card(STAN) oznacza liczbę elementów s STAN .

Krok 12

Przeszukaj stos STAN w poszukiwaniu elementu g spełniającego warunki:

(1) g $\neq$ TOP(STAN)

(2) WIEK(g) $\neq$ 0

Wykonane wtedy gdy WIEK (f) = 1

(3) g leży możliwie najbliżej wierzchołka stosu STAN.
Załóżmy, że TOP(LEWA(g)) = r
Wykonaj procedurę MOŻNA(r,f) (przypominamy, że f = TOP(STAN))
Jeżeli daje ona odpowiedź negatywną to podstaw WIEK(TOP(STAN)) := 2 i wykonaj krok 1; wpp wykonaj krok 13;

Krok 13
Wykonaj procedurę PRZEKAŻ(r,f) *); WIEK(g) := 1; STAN := ZDEJMIJ(STAN);
wykonaj krok 1; (f i g są tymi samymi predykatami, o których mówiliśmy wyżej)

Krok 14
dla i=1,2,...,N_X podstaw: WARTOSC(X,adr,i) := A_i (TOP(STAN)) zatrzymaj wykonywanie algorytmu.

Krok 15
Zasygnalizuj błąd semantyczny; zatrzymaj wykonywanie algorytmu

Procedury TA i MOŻNA użyte zostały jedynie w celu zwiększenia komunikatywności algorytmu. W rzeczywistości wykonanie procedur OJCIEC i PRZEKAŻ wiąże się zawsze ze sprawdzeniami opisanymi dla procedur TA i MOŻNA.

(1). Opis procedury OJCIEC(r)
Załóżmy, że po lewej stronie hiperprodukcji r występuje hiperpojęcie $f(F_1,\ldots,F_{N_f})$.
Procedura OJCIEC(r) wykona wtedy czynności:
$A_1(f) := W_1(f) + D(F_1)$; $A_2(f) := W_2(f) + D(F_2)$;...; $A_{N_f}(f) := W_{N_f}(f) + D(F_{N_f})$;
Załóżmy teraz, że $F_{i_1} = \alpha_1 L \beta_1$, $F_{i_2} = \alpha_2 L \beta_2, \ldots, F_{i_m} = \alpha_m L \beta_m$ są wszystkimi wystąpieniami zmiennej L po lewej stronie hiperprodukcji r.
Dla każdej takiej zmiennej są wykonywane działania:
$L_1 := \Phi(A_{i_1}(f), D(F_{i_1}), D(\alpha_1)) + \Phi(A_{i_2}(f), D(F_{i_2}), D(\alpha_2)) + \ldots \Phi(A_{i_m}(f), D(F_{i_m}), D(\alpha_m))$
$A_{i_1}(f) := P(A_{i_1}(f), D(F_{i_1}), D(\alpha_1), L_1);\ldots; A_{i_m}(f) := P(A_{i_m}(f), D(F_{i_m}), D(\alpha_m), L_1)$
(zwracamy uwagę, że liczby $i_1, i_2, \ldots, i_m$ nie muszą wszystkie być różne).
W przypadku niewykonalności któregokolwiek z powyższych działań postępujemy tak jak gdyby procedura TA(r) dała odpowiedź negatywną.

(2). Opis procedury SYN(r,i)
Załóżmy, że hiperprodukcja r jest postaci:
$r = f(F_1,\ldots,F_{N_f}) \to f_1(F_1^1,\ldots,F_{N_{f_1}}^1)\ldots f_i(F_1^i,\ldots,F_{N_{f_i}}^i)\ldots f_n(F_1^n,\ldots,F_{N_{f_n}}^n)$
Predykatowi f_i zostanie przydzielona pamięć o odpowiedniej strukturze oraz dla wszystkich $j=1,2,\ldots,N_{f_i}$ wykonają się podstawienia: $W_j(f_i) := D(F_i)$;

*) procedury MOŻNA (r,f) i PRZEKAŻ (r,f) są wykonywane dla predykatów f,f,f,...,g znajdujących się najbliżej wierzchołka stosu STAN.

Przyjmijmy teraz, że zmienna afiksowa L występuje po lewej stronie hiperprodukcji r w którymś z napisów $F_k = \alpha L \beta$ (wybór wystąpienia L po lewej stronie nie ma znaczenia) oraz niech $F^i_{j_1} = \alpha_1 L \beta_1, F^i_{j_2} = \alpha_2 L \beta_2, \ldots, F^i_{j_t} = \alpha_t L \beta_t$ będą wszystkimi wystąpieniami zmiennej L w hiperpojęciu $f_i(F_1, \ldots, F_{N_{f_i}})$.

Dla każdej zmiennej o własnościach analogicznych jak L wykonujemy działania:

$L_1 := \Phi(A_k(f), D(F_k), D(\alpha))$;

$W_{j_1}(f_i) := P(W_{j_1}(f_i), D(F^i_{j_1}), D(\alpha_1), L_1); \ldots; W_{j_t}(f_i) := P(W_{j_t}(f_i), D(F_{j_t}), D(\alpha_t), L_1)$;

(3). Opis procedury PRZEKAŻ(r,f)

Przyjmijmy, że hiperprodukcja r jest postaci:

$r = g(G_1, \ldots, G_{N_g}) \to f_1(F^1_1, \ldots, F^1_{N_{f_1}}) \ldots f(F_1, \ldots, F_{N_f}) f_k(F^k_1, \ldots, F^k_{N_{f_k}}) \ldots f_n(F^n_1, \ldots, F^n_{N_{f_n}})$

Załóżmy, że $F_j = \alpha_1 L \beta_1$, $F_{j_2} = \alpha_2 L \beta_2, \ldots, F_{j_m} = \alpha_m L \beta_m$ są wszystkimi wystąpieniami zmiennej L w hiperpojęciu $f(F_1, \ldots, F_{N_f})$. Wykonają się wtedy działania:

$L_1 := P(A_{j_1}(f), D(F_{j_1}), D(\alpha_1)) + \ldots + P(A_{j_m}(f), D(F_{j_m}), D(\alpha_m))$

Przypuśćmy dalej, że $F^{i_1}_{L_1} = \gamma_1 L \delta_1$, $F^{i_2}_{L_2} = \gamma_2 L \delta_2, \ldots, F^{i_\rho}_{L_\rho} = \gamma_p L \delta_p$ są wszystkimi wystąpieniami zmiennej L w hiperpojęciach $f_k(F^k_1, \ldots, F^k_{N_{f_k}}), \ldots, f_n(F^n_1, \ldots, F^n_{N_{f_n}})$

Dla wszystkich s=1,2,...,p wykonają się wtedy kolejno (tzn. wynik poprzedniego jest argumentem następnego) działania:

$W_{i_s} := P(W_{i_s}(f_{i_s}), D(F^{i_s}_{i_s}), D(\gamma_s), L_1) + \Phi(W_{i_s}(f_{i_s}), D(F^{i_s}_{i_s}), D(\gamma_s))$;

Jeżeli zmienna L występuje jeszcze w którymś z G_i i $G_{n_1} = \varrho_1 L \omega_1$, $G_{n_2} = \varrho_2 L \omega_2, \ldots, G_{n_x} = \varrho_x L \omega_x$ są wszystkimi takimi wystąpieniami, to wtedy dla wszystkich u=1,2,...,x wykonają się kolejno działania:

$A_{n_u}(g) := P(A_{n_u}(g), D(G_{n_u}), D(\varrho_u), L_1) + \Phi(A_{n_u}(g), D(G_{n_u}), D(\varrho_u))$

Analogiczne działania wykonujemy dla wszystkich zmiennych występujących w hiperprodukcji r o własnościach takich jak zmienna L.

Jeżeli którekolwiek z opisanych wyżej działań jest niewykonalne, postępujemy tak jak w przypadku gdy MOŻNA(r,f) daje odpowiedź negatywną.

"Dodawanie", o którym była mowa w procedurach OJCIEC, SYN i PRZEKAŻ to oczywiście dodawanie form.

Na stosie RESULT znajdują się, po zakończeniu wykonywania procedury PREDYKAT, adresy pamięci przydzielonej hiperprodukcjom. Przeglądając zawartości tych pamięci łatwo przyporządkowujemy hiperprodukcjom odpowiednie produkcje.

Twierdzenie 2

Procedura PREDYKAT(X,adr) jest procedurą skończoną.

Jeżeli analizowane słowo należy do języka L(G) to PREDYKAT (X,adr) obliczy wartości terminalne wszystkich węzłów $(X,adr,j_1),(X,adr,j_2),\ldots,(X,adr,j_L)$ gdzie $j_1,j_2,\ldots,j_L$ są numerami wszystkich syntetyzowanych afikspozycji X.

Dowód:

Pierwsza część twierdzenia wynika natychmiast z tego, że rozważamy rozszerzoną grama-

tykę afiksową o dobrze zdefiniowanym zbiorze hiperprodukcji predykatowych.
W celu udowodnienia drugiej części zauważmy najpierw, że przy przyjętych dla twierdzenia założeniach, procedury OJCIEC, SYN i PRZEKAŹ są wykonalne (udowadniamy to analogicznie jak twierdzenie 4 Dowód B, cz.I, rozdz. 3.)
Tezę twierdzenia otrzymujemy korzystając z powyższej uwagi oraz następujących faktów:

(1) Każda zmienna afiksowa L występująca w dowolnej hiperprodukcji p musi w niej występować po lewej stronie na pozycji dziedziczonej albo po prawej stronie na pozycji syntetyzowanej - na lewo od wszystkich wystąpień stosowanych zmiennej L.

(2) W chwili rozpoczęcia wykonywania procedury PREDYKAT(X,adr) wszystkie afikspozycje dziedziczone X mają wartości terminalne.

(3) Dla każdej kolejnej analizowanej przez procedurę PREDYKAT(X,adr) hiperprodukcji p, najbliżej wierzchołka stosu STAN będzie pierwszy od lewej predykat z prawej strony hiperprodukcji p, następnie drugi od lewej, itd.

(4) Jeżeli wystąpienie definiujące danej zmiennej L (w analizowanej hiperprodukcji p) znajduje się po lewej stronie p, to we wszystkich swoich występieniach L otrzyma wartość terminalną: w wyniku działania procedury OJCIEC jeżeli dane występienie stosowane L znajduje się po lewej stronie p, lub w wyniku działania procedury SYN jeżeli dane występienie stosowane L znajduje się po prawej stronie p.

(5) Jeżeli wystąpienie definiujące danej zmiennej L w analizowanej hiperprodukcji p znajduje się po prawej stronie p, to we wszystkich swoich wystąpieniach stosowanych L otrzyma wartość terminalną w wyniku działania procedury PRZEKAŹ.

(6) Jeżeli wszystkie zmienne afiksowe, występujące na danej afikspozycji predykatu f, otrzymają wartość terminalną, to automatycznie afikspozycja ta otrzyma wartość terminalną.
Punkty te a więc i całe twierdzenie będą udowadniane indukcyjnie.
Udowodnimy dla przykładu punkt (4),
Załóżmy, że hiperprodukcja h_1:
$h_1 = X(F_1,\ldots,F_{N_X}) \to f_1(F_1^1,\ldots,F_{N_{f_1}}^1)\ldots f_n\ F_1,\ldots,F_N$ jest pierwszą z hiperprodukcji analizowanych przez procedurę PREDYKAT (X,adr).
Przyjmijmy, że zmienna afiksowa L występuje po lewej stronie hiperprodukcji h_1 w F_i i niech $F_i = \alpha_1 L \beta_1$. Przypuśćmy dalej, że i-ta afikspozycja X jest dziedziczona. Rozważmy dwa przypadki:

(a) L występuje po lewej stronie hiperprodukcji h_1 w F_j (gdzie j-ta afikspozycja X jest syntetyzowana). Niech $F_j = \alpha_2 L \beta_2$.
Procedura OJCIEC wykona wtedy działanie:
$L_1 := \Phi(A_i(X),D(F_i),D(\alpha_1)) + \Phi(A_j(X),D(F_j),D(\alpha_2))$
Zgodnie z powyższym punktem (2) L_1 ma wartość terminalną, a więc po wykonaniu działania: $A_j(X) := P(A_j(X),D(F_j),D(\alpha_1),L_1)$; wystąpienie zmiennej L w F_j otrzyma wartość terminalną.

(b) L występuje po prawej stronie hiperprodukcji h_1 w F_i (i-ta afikspozycja predykatu f_k) - załóżmy, że $F_i^k = \alpha_3 L \beta_3$.

Procedura SYN(h_1,k) wykona wtedy działanie:

$L_1 := \Phi(A_1(X),\ D(F_i),\ D(\alpha_1))$

Korzystając z punktu (2) widzimy, że L_1 ma wartość terminalną a stąd, po wykonaniu: $W_i(f_k) := P(W_i(f_k), D(F_i^k), D(\alpha_3), L_1)$

wystąpienie zmiennej L w F_i^k otrzyma wartość terminalną.

Dla kolejnej m-tej analizowanej przez procedurę PREDYKAT hiperprodukcji punkt (4) udowadniamy analogicznie. Korzystamy przy tym z faktu, że wartość terminalna dowolnej zmiennej L występującej w hiperpojęciu $f(G_1,\ldots,G_{N_f})$(znajdującemu się po lewej stronie m-tej hiperprodukcji)na pozycji dziedziczonej, została obliczona, w poprzednio analizowanej. przez procedurę PREDYKAT, hiperprodukcji. Wynika to z założenia indukcyjnego (punkty (4) i (5)).

DODATEK C

WSTĘP

Jak pamiętamy w trakcie tworzenia (a następnie aktualizacji) grafu wywodu analizowanego słowa (patrz parser dla języka opisanego gramatyką afiksową), przyporządkowaliśmy węzłom tego grafu - ich wartości. Wartości owe były pewnymi formami zdaniowymi.

Jednak zapis form zdaniowych, polegający na ich wypisaniu explicite, jest dla naszych celów mało przydatny. Operacja tworzenia sumy dwóch zadanych form zdaniowych, byłaby bowiem wtedy bardzo trudna do zrealizowania.

Z tego powodu dla każdej formy zdaniowej (wartości węzła grafu wywodu) będziemy używać zapisu, który nazwiemy strukturą tej formy. Struktura formy jest pewnym sposobem liniowego zapisu drzewa wywodu danej formy.

Formalną definicję struktury formy zdaniowej, a następnie sumy struktur, podamy w dalszym ciągu pracy. Teraz ograniczymy się jedynie do przykładu.

Rozważmy gramatykę typu LR(1) postaci $G = (A_N, A_T, R, a)$; gdzie:

$A_N = \{LISTA, ETYKIETA, CYFRA\}$, $A_T = \{0, 1, ,\}$

$R = \{$ LISTA → LISTA , ETYKIETA ETYKIETA → CYFRA

LISTA → ε CYFRA → 0

ETYKIETA → CYFRA ETYKIETA CYFRA → 1 $\}$

a = LISTA

oraz formę zdaniową w tej gramatyce:

M = , 0 , 1 ETYKIETA , CYFRA

Drzewo wywodu formy M na postać

Strukturę formy M utworzymy trawersując powyższe drzewo w następujący sposób:

Odwiedź najwyższy i stojący najbardziej na lewo wierzchołek W, taki którego jeszcze nie przeglądałeś. Wypisz go.

Jeżeli W ma synów to napisz nawias otwierający, a następnie odwiedź lewego skrajnego syna wierzchołka W. Wypisz tego syna.

Jeżeli W nie ma synów to odwiedź brata W stojącego po prawej stronie W. Jeżeli nie ma takich wierzchołków, które byłyby braćmi stojącymi na prawo od W napisz nawias zamykający.

Kolejność przeglądania wierzchołków ilustrują narysowane wyżej strzałki. Otrzymaliśmy rezultat:

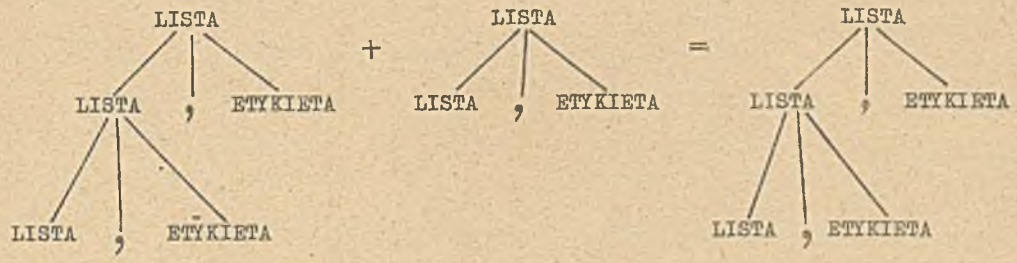
ST(M) = LISTA(LISTA (LISTA (LISTA (ε), ETYKIETA (CYFRA (0)),
ETYKIETA (CYFRA (1) ETYKIETA)), ETYKIETA (CYFRA))

Między dodawaniem form zdaniowych a dodawaniem struktur tych form zachodzi oczywista odpowiedniość.

Załóżmy np., że M = LISTA, ETYKIETA, ETYKIETA; N = LISTA, ETYKIETA.

Wtedy: M + N = LISTA, ETYKIETA, ETYKIETA

Działaniu temu odpowiada "dodawanie" drzew:



oraz "dodawanie" struktur:

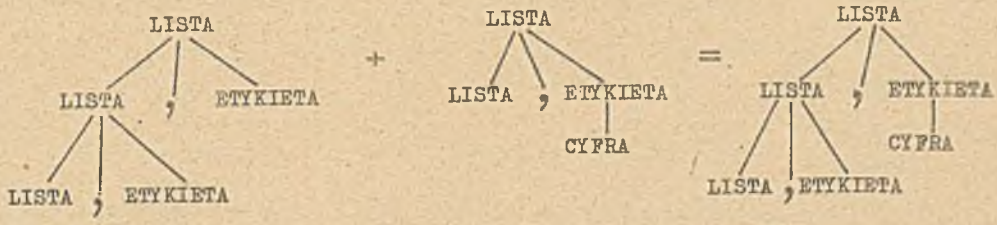
LISTA (LISTA (LISTA, ETYKIETA), ETYKIETA) + LISTA(LISTA, ETYKIETA) =

= LISTA (LISTA (LISTA, ETYKIETA), ETYKIETA

Rozważmy jeszcze formy zdaniowe:

M = LISTA, ETYKIETA, ETYKIETA, N = LISTA, CYFRA. Wtedy:

M + N = LISTA, ETYKIETA, CYFRA;



LISTA(LISTA(LISTA, ETYKIETA), ETYKIETA)+ LISTA(LISTA, ETYKIETA(CYFRA)) =

LISTA(LISTA(LISTA, ETYKIETA), ETYKIETA(CYFRA))

Formalną definicję struktury formy zdaniowej oraz sumy struktur podamy w rozdziałach drugim i trzecim. Rozdział 1 zawiera pewne pojęcia pomocnicze.

Rozdział 1. POJĘCIA POMOCNICZE

Aby móc przyporządkować formie zdaniowej F odpowiadającą jej strukturę ST(F), musimy znaleźć wywód tej formy. Pomocne nam będą następujące definicje i twierdzenia:

Definicja 1.1

Niech będzie dana gramatyka bezkontekstowa $G_a = (A_N, A_T, R, a)$,
Utwórzmy zbiór A_N^F taki, że:

(1) $A_N^F \cap (A_N \cup A_T)^* = \phi$

(2) card (A_N^F) = card (A_N)

Załóżmy dalej, że FOR : $(A_N \cup A_T)^* \xrightarrow{na} (A_N^F \cup A_T)^*$ jest dowolnym (ale ustalonym) homomorfizmem różnowartościowym spełniającym warunki:

(1) FOR $(A_N) = A_N^F$

(2) dla każdego $x \in A_T$ ma miejsce: FOR(x) = x

Zdefiniujmy jeszcze zbiory A_T^F i R_F :

$A_T^F = (A_T \cup A_N^F)$, $R_F = R \cup P_F$ gdzie:

P_F jest zbiorem wszystkich produkcji postaci $X \rightarrow FOR(X)$ takich, że $X \in A_N$.

Gramatyką form gramatyki G_a nazwiemy gramatykę bezkontekstową

$G_R^F = (A_N, A_T^F, R_F, a)$

Twierdzenie 1

Jeżeli G_R jest gramatyką typu LR(k) to gramatyka form G_R^F też jest gramatyką typu LR(k).

Dowód:

Przyjmijmy, że $x, w, y \in A_T^F$; $\alpha, \beta, \gamma \in (A_N \cup A_T^F)^*$; A, B $\in A_N$ są dowolnymi napisami spełniającymi założenia:

(1) $a \xrightarrow[G_R^F]{*}_R \alpha A w \xrightarrow[G_R^F]{}_R \alpha\beta w$

(2) $a \xrightarrow[G_R^F]{*}_R \gamma B x \xrightarrow[G_R^F]{}_R \alpha\beta y$

(3) $FIRST_k(w) = FIRST_k(y)$

Ponieważ wyżej wymienione wywody są prawostronne więc ani α, ani γ nie zawierają żadnego z symboli ze zbioru A_N^F. Rozważmy dwa przypadki:

(a) β nie zawiera żadnego symbolu ze zbioru A_N^F

Niech: $x = x_0^1 X_F x_1 X_F^2 x_2 \ldots X_F^l x_l$

$y = Y_0 Y_F^1 y_1 Y_F^2 y_2 \ldots Y_F^m Y_m$

$w = w_0 W_F^1 w_1 W_F^2 w_2 \ldots W_F^n y_n$

gdzie dla każdego i=0,1,...,l; j=0,1,...,m; k=0,1,...,n jest:

$x_i, Y_j, w_k \in V_T^*$; $X_F^i = FOR(X_i)$; $Y_F^j = FOR(Y_j)$, $W_F^k = FOR(W_k)$; $X_i, Y_j, W_K \in A_N$

Zastąpmy teraz każde X_F^i napisem $s_i \in A_T^*$ takim, że $X_i \xrightarrow[G_R]{*}_R s_i$ i $|s_i| > 0$, każde Y_F^j napisem $t_j \in A_T^*$ takim, że $Y_j \xrightarrow[G_R]{*}_R t_j$ i $|t_j| > 0$, każde W_F^k napisem $h_k \in A_T^*$ takim, że $W_k \xrightarrow[G_R]{*}_R h_k$ i $|h_k| > 0$, przy czym te same symbole

nieterminalne zastępujemy tymi samymi napisami, różne symbole nieterminalne zastępujemy różnymi napisami (tzn. jeżeli np. X_i i Y_j oznaczają ten sam symbol (a więc X_F i Y_F też) to X_F^i i Y_F^i zastąpimy tymi samymi napisami).

Taka konstrukcja jest zawsze możliwa (bowiem nie ograniczając ogólności możemy założyć, że gramatyka G nie zawiera symboli X takich, że produkcja $X \to \varepsilon$ byłaby jedyną produkcją z X po lewej stronie).

W wyniku tej konstrukcji otrzymujemy:

$x' = x_0 s_1 \; x_1 s_2 x_2 \ldots s_l x_l$

$y' = y_0 t_1 \; y_1 t_2 y_2 \ldots t_n y_m$

$w' = w_0 h_1 \; w_1 h_2 w_2 \ldots h_n w_n$

Ponieważ α, β i γ nie zawierają symboli ze zbioru A_N^F to:

a $\overset{*}{\underset{G_R}{\Longrightarrow}} \alpha A w'$ $\underset{G_R}{\Longrightarrow} \alpha\beta w'$

a $\overset{*}{\underset{G_R}{\Longrightarrow}} \gamma B x'$ $\underset{G_R}{\Longrightarrow} \alpha\beta y'$

Skoro $FIRST_k(w) = FIRST_k(y)$, więc tym bardziej $FIRST_k(w') = FIRST_k(y')$

Korzystając teraz z tego, że gramatyka G_R jest typu LR(k) mamy:

a $\overset{*}{\underset{G_R}{\Longrightarrow}} \alpha A \; w' \underset{G_R}{\Longrightarrow} \alpha\beta w'$

a $\overset{*}{\underset{G_R}{\Longrightarrow}} \alpha A \; x' \underset{G_R}{\Longrightarrow} \alpha\beta x'$ bo $\gamma = \alpha$ $B = A$, $x' = y'$

Wynika stąd, że ostatnią zastosowaną w wywodzie (2) produkcją (wywód ten ma teraz postać a $\overset{*}{\underset{G_R^F}{\Longrightarrow}} \alpha A x \underset{G_R^F}{\Longrightarrow} \alpha\beta y$) była $A \to \beta$,

a stąd mamy natychmiast, że $x = y$. Razem z poprzednio uzyskanymi wynikami ($\alpha = \gamma$, $B = A$) dowodzi to tezy.

(b) $\beta = FOR(A)$

Oczywiste jest, że forma αAy jest jedyną formą γBx taką, że:

$\gamma \in (A_N \cup A_T)^*$, $B \in A_N$, $x \in (A_N^F)^*$, $y \in (A_N^F)^*$ i $\gamma Bx \underset{G_R^F}{\Longrightarrow} \alpha FOR(A) y$

Definicja 1.2

Niech $G = (A_N, A_T, R, a)$ będzie gramatyką bezkontekstową.

Utwórzmy zbiór symboli A'_N taki, że

(1) $A'_N \cap (A_N \cup A_T) = \phi$

(2) $card(A'_N) = card(A_N)$

Oznaczmy przez NO dowolną (ale ustaloną) funkcję różnowartościową

NO : $A_N \xrightarrow{na} A'_N$ i niech $[_A$ oznacza wartość funkcji NO dla elementu A ze zbioru A_N (tzn. $[_A = NO(A)$).

Gramatyką pochodną gramatyki G nazwiemy gramatykę

$G' = (A_N, A'_T, R', a)$, gdzie

$A'_T = A_T \cup A'_N \cup \{], e\}$ (] i e są symbolami $\notin A_N \cup A_T \cup A'_N$)

R' jest zbiorem wszystkich produkcji postaci $A \to [_A x]$ takich, że $x \neq \varepsilon$ i produkcja

$A \to x$ należy do R. Do R' dołączamy wszystkie produkcje $A \to [_A e]$ takie, że $A \to \varepsilon$ należy do R.

Określimy jeszcze dwa homomorfizmy:

(1) $H_{AF} : (A_N \cup A_T')^* \to (A_N \cup A_T)^*$

$$H_{AF}(x) = \begin{cases} x & \text{dla } x \in A_N \cup A_T \\ \varepsilon & \text{dla } x \in A_T' - A_T \end{cases}$$

(2) $H_{AF} : (A_N \cup A_T')^* \to (A_N \cup A_T \cup \{e\})^*$

$$H_{AF}(x) = \begin{cases} x & \text{dla } x \in A_N \cup A_T \cup \{e\} \\ \varepsilon & \text{dla } x \in A_N' \cup \{]\} \end{cases}$$

Twierdzenie 2

Niech $G = (A_N, A_T, R, a)$ będzie gramatyką typu LR(k), $G' = (A_N, A_T', R', a)$ jej gramatyką pochodną.

Jeżeli $a \overset{*}{\underset{G}{\Rightarrow}} M$ i $a \overset{*}{\underset{G'}{\Rightarrow}} N$ oraz $H_{AF}(M) = H_{AF}(N)$ to $M = N$

Dowód:

W dowodzie korzystać będziemy z następującego lematu:

Gramatyka bezkontekstowa G jest jednoznaczna $\Leftrightarrow$ dla każdego słowa $x \in L(G)$ istnieje dokładnie jeden napis y ($y \in (A_N \cup A_T')^*$) taki, że $H_{AF}(y) = x$

Dowód tego lematu znajduje się w pracy A. Blikle [1] - str. 152

Załóżmy teraz, że homomorfizmy: $FOR : (A_N \cup A_T)^* \to (A_N \cup A_T)^*$

$FOR' : (A_N \cup A_T')^* \to (A_N \cup A_T')^*$

określone są tak jak w definicji 1.1 odpowiednio dla gramatyk:

$G_F = (A_N, A_N^F \cup A_T, R_F, a)$ - gramatyka form gramatyki G.

$G_F' = (A_N, (A_T^F)', R_F', a)$ - gramatyka pochodna gramatyki G_F.

Przyjmijmy dalej, że homomorfizmy: $H_{AF} : (A_N \cup A_T')^* \to (A_N \cup A_T)^*$

$H_{AF} : (A_N \cup (A_T^F)')^* \to (A_N \cup A_T^F)^*$

są odpowiednimi homomorfizmami określonymi analogicznie jak w definicji 1.2

Niech $M_F = FOR(M)$, $N_F = FOR'(N)$. Wtedy oczywiście $a \overset{*}{\underset{G_F}{\Rightarrow}} M_F$ i $a \overset{*}{\underset{G_F'}{\Rightarrow}} N_F$.

Skoro $H_{AF}(M) = H_{AF}(N)$ to $H_{AF}^F(M_F) = H_{AF}^F(N_F)$.

Korzystając teraz z tego, że każda gramatyka typu LR(k) jest gramatyką jednoznaczną oraz z tego, że $H_{AF}(M_F)$, $H_{AF}(N_F) \in L(G_F)$ i stosując wyżej podany lemat otrzymujemy: $M_F = N_F$ (wynika to również, z twierdzenia 1).

Stąd oczywiście $M = N$.

A więc dla każdej formy zdaniowej M gramatyki G istnieje dokładnie jedna forma N w gramatyce G' taka, że $H_{AF}(N) = M$ (przy założeniu, że G jest typu LR(k)).

Taką formę będziemy nazywać formą pochodną formy M i oznaczać przez FP(M).

Rozdział 2. STRUKTURA FORMY ZDANIOWEJ

Załóżmy, że $G_A = (A_N, A_T, R, a)$ jest gramatyką typu LR(k), oraz że $G'_a = (A_N, A'_T, R, a)$ jest gramatyką pochodną gramatyki G_a.

Zdefiniujemy homomorfizm: $f : (A_N \cup A'_T)^* \to (A_N \cup A_T \cup \{(,), e\})^*$

$$f(a) = \begin{cases} a & \text{gdy } a \in V_N \cup V_T \cup \{e\} \\ A\,(& \text{gdy } a = \lceil_A \\ ) & \text{gdy } a = \rfloor \end{cases}$$

Definicja 2.1

Strukturą formy zdaniowej M w gramatyce G_a nazwiemy napis ST(M) taki, że $ST(M) = f(M')$, gdzie M' jest formą pochodną formy M.

Ponieważ dla danej formy zdaniowej M istnieje dokładnie jedna forma pochodna, więc pojęcie struktury ST(M) jest określone jednoznacznie.

Zamiast "istnieje forma M taka, że napis ST jest strukturą formy M w gramatyce G" będziemy często pisać "napis ST jest strukturą w gramatyce G",

Ustalmy we wszystkich dalszych rozważaniach gramatykę $G_a = (A_N, A_T, R, a)$ typu LR(k).

Definicja 2.2

(0) Załóżmy, że ST_1 jest strukturą w gramatyce G_a. Napis ST_2 nazwiemy strukturą pochodną zerowego rzędu struktury ST_1 jeżeli $ST_2 = ST_1$

(1) Załóżmy, że $ST_1 = \alpha A \beta$ (gdzie $A \in A_N$) jest strukturą w gramatyce G_a. Napis $ST_2 = \alpha A(x) \beta$ nazwiemy strukturą pochodną struktury ST_1 jeżeli $A \to x \in R$

(n) Załóżmy, że ST_1 jest strukturą w gramatyce G_a.

Napis $ST_2 = \alpha A(x) \beta$ nazwiemy strukturą pochodną n-tego rzędu struktury ST_1 jeżeli: $A \to x \in R$ oraz napis $ST = \alpha A \beta$ jest strukturą pochodną rzędu n-1 struktury ST_1.

Powiemy, że ST_2 jest strukturą pochodną struktury ST_1 jeżeli istnieje n takie, że ST_2 jest strukturą pochodną n-tego rzędu struktury ST_1.

Jak łatwo zauważyć struktura pochodna struktury w gramatyce G_a też jest strukturą w gramatyce G_a.

Prawdziwy jest następujący lemat ilustrujący pojęcie struktury pochodnej.

Lemat 1

Przypuśćmy, że $a \overset{*}{\underset{G_a}{\Rightarrow}} M$ i $a \overset{*}{\underset{G_a}{\Rightarrow}} N$. Wtedy równoważne są zdania

(1) $M \overset{*}{\underset{G_a}{\Rightarrow}} N$

(2) Struktura ST(N) jest strukturą pochodną struktury ST(M).

Dowód:

(1) $\Rightarrow$ (2)

Ponieważ G_a jest typu LR(k), więc liczba n produkcji zastosowanych w wywodzie $M \overset{*}{\underset{G_a}{\Rightarrow}} N$ jest określona jednoznacznie.

Dowód przeprowadzimy przez indukcję względem n.

Dla n = 1 lemat jest oczywiście prawdziwy.

Załóżmy teraz, że dla wszystkich form zdaniowych F takich, że

(a) $M \overset{*}{\underset{G_a}{\Rightarrow}} F$

(b) w wywodzie $M \overset{*}{\underset{G_a}{\Rightarrow}} F$ zastosowano n produkcji

struktura ST(F) jest pochodną n-tego rzędu struktury ST(M).

Wybierzmy spośród owych form F taką formę N_1, że $N_1 \overset{}{\underset{G_a}{\Rightarrow}} N$. Niech $N_1 = \alpha A \beta$ i $N = \alpha x \beta$ gdzie $A \rightarrow x \in R$.

Wtedy $ST(N_1) = \alpha_1 A \beta_1$ oraz $ST(N) = \alpha_1 A(x) \beta_1$ a więc struktura ST(N) jest pochodną pierwszego rzędu struktury $ST(N_1)$,

Teza lematu wynika stąd natychmiast.

(2) $\Rightarrow$ (1)

Dowód przeprowadzamy analogicznie jak dowód implikacji (1) $\Rightarrow$ (2).

Indukcję zastosujemy względem liczby n takiej, że struktura ST(N) jest pochodną n-tego rzędu struktury ST(M)(nie ma tutaj znaczenia czy owa liczba n jest określona jednoznacznie - wystarczy wybrać jedną spośród nich).

Podamy teraz algorytm tworzenia struktury ST(M) formy M, dla zadanej formy $M = a_1 a_2 \ldots a_n$ (gdzie $a_1 \in A_N \cup A_T$) i gramatyki $G_a = (A_N, A_T, R, s)$

Zamiast "w przeciwnym przypadku" będziemy pisać wpp.

W algorytmie tym zakładamy, że napis pusty ε jest pewnym symbolem ze zbioru A_T.

Algorytm

Krok 1

Utwórz gramatykę form $G'_a = (A_N, A_T, R_p, s)$ gramatyki G_a, oraz napis M_F FOR (M); (gdzie FOR to homomorfizm opisany w definicji 1.1 w Dodatku C, rozdz.1). Wykonaj krok 2.

Krok 2

Korzystając z tego, że gramatyka G_a jest typu LR(k) (wynika to z twierdzenia 1) zbuduj parser tej gramatyki,

* Będziemy dalej zakładać, że parser ten podaje najpierw pierwszą z produkcji zastosowanych w wywodzie prawostronnym a $\overset{*}{\underset{G_a}{\Rightarrow}} M_F$, potem drugą, itd. (Kolejność taką możemy łatwo uzyskać "odwracając" ciąg produkcji rzeczywiście podawanych przez parser gramatyki LR(k)). Załóżmy jeszcze, że w wywodzie M_F zastosowano N produkcji *

wynik := a; n := 1; wykonaj krok 3

Krok 3

Załóżmy, że prawą stroną n-tej produkcji podanej przez parser jest napis x. Niech aktualnie:

wynik = $\alpha A \beta$; gdzie $\beta \in (A_T^F \cup \{(,)\})^*$, $A \in A_N$

Wtedy:

wynik := $\alpha A(x) \beta$ wykonaj krok 4

Krok 4

n := n+1;

Jeżeli $n \leq N$ wykonaj krok 3, wpp wykonaj krok 5

Krok 5

Zastąp każdy symbol $a \in A_T^F$ występujący w napisie wynik symbolem b takim, że FOR(b) = a.

Zatrzymaj algorytm.

W praktyce strukturę formy zdaniowej będziemy zapisywać w innej, wygodniejszej dla naszych celów, postaci:

Załóżmy, że napis ST jest strukturą w gramatyce $G_a = (A_N, A_T, R, a)$, oraz że napis $X(\beta)$ jest strukturą w gramatyce $G_X = (A_N, A_T, R, X)$ gdzie $\beta \in (A_N \cup A_T \cup \{(,)\})^*$.
Jeżeli istnieją $\alpha, \gamma \in (A_N \cup A_T \cup \{(,)\})^*$ takie, że ST = $a(\alpha X(\beta) \gamma)$ to strukturę ST będziemy zapisywać w postaci

ST = $a(\alpha \S \gamma)$

gdzie § jest adresem wskazującym na pamięć w której będzie umieszczona struktura $X(\beta)$.
Oczywiście obydwa zapisy struktury ST są równoważne i znając zapis ST = $a(\alpha \S \gamma)$ możemy łatwo uzyskać zapis ST = $a(\alpha X(\beta) \gamma)$ → $X(\beta)$

Niech, na przykład, ST będzie postaci:

ST = LISTA(adres, ETYKIETA) → LISTA(LISTA, ETYKIETA)

Strukturę tę można zapisać: ST = LISTA(LISTA(LISTA,ETYKIETA),ETYKIETA)

Owym różnym zapisom struktur odpowiadają różne zapisy drzewa:

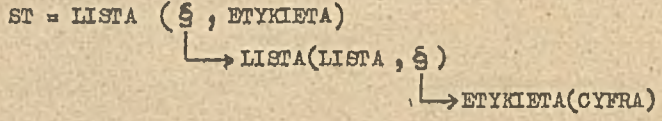
W jednej strukturze może występować kilka adresów, np:

ST = LISTA (§ , ETYKIETA (§)) → CYFRA (0)
§ → LISTA(LISTA , ETYKIETA)

zapisowi temu równoważny jest zapis:

ST = LISTA(LISTA(LISTA , ETYKIETA), ETYKIETA(CYFRA (0)))

W szczególności może się zdarzyć, że w pamięci wskazywanej przez dany adres występuje też struktura, która zawiera adres, a więc może mieć miejsce sytuacja:



W dalszym ciągu pracy będziemy milcząco zakładać, że dla każdej struktury takie "adresowanie" jest procesem skończonym.

Nie podamy formalnej definicji struktury pochodnej dla takiego zapisu struktur. Pojęcie to wydaje się intuicyjnie zrozumiałe (wystarczy znaleźć dla danej struktury równoważny zapis "konwencjonalny").

Rozdział 3. SUMA STRUKTUR

Pojęcie sumy struktur określiliśmy nieformalnie na początku tego dodatku. Teraz podamy formalną definicję (zakładamy, że obydwie struktury są zapisane w postaci "jawnej" - tzn. bez adresów)

Definicja 3.1

Załóżmy, że ST_1 i ST_2 są strukturami gramatyki $G_a = (A_N, A_T, R, a)$.
Jeżeli istnieje taka struktura ST_3 tej gramatyki, że jest ona strukturą pochodną zarówno ST_1 jak i ST_2 to powiemy, że określone jest dodawanie struktur ST_1 i ST_2. Sumą tych struktur nazwiemy wtedy taką strukturę SUMA, że:

(1) SUMA jest strukturą pochodną struktury ST_1

(2) SUMA jest strukturą pochodną struktury ST_2

(3) dla każdej struktury SUMA' spełniającej (1) i (2) SUMA' jest strukturą pochodną struktury SUMA.

Strukturę SUMA będziemy oznaczać przez $ST_1 + ST_2$.

Twierdzenie 3

Załóżmy, że struktury ST_1, ST_2, ST_3 są strukturami gramatyki $G_a = (A_N, A_T, R, a)$.
Przyjmijmy, że istnieje taka struktura ST_4 tej gramatyki, że jest ona strukturą pochodną struktur ST_1, ST_2 i ST_3.
Wtedy:

(1) istnieje dokładnie jedna struktura μ gramatyki G_a taka, że $\mu = ST_1 + ST_2$

(2) ST_4 jest strukturą pochodną struktury $ST_1 + ST_2$

(3) $ST_1 + ST_2 = ST_2 + ST_1$

(4) $ST_1 + (ST_2 + ST_3) = (ST_1 + ST_2) + ST_3$

(5) $ST_1 + ST_4 = ST_4$

Dowód:

(1) wynika natychmiast z definicji oraz z faktu, że G_a jest gramatyką typu LR(k), więc nieprawdą jest, że dowolna struktura może być swoją pochodną rzędu większego niż zero.

(2) Wynika z tego, że struktura ST_4 jest strukturą pochodną struktur ST_1 i ST_2, a więc spełnia punkty (1) i (2) definicji.

(3) Oczywiste

(4) Wykażemy, że struktura $\eta = (ST_1 + ST_2) + ST_3$ jest strukturą pochodną struktury $\xi = ST_1 + (ST_2 + ST_3)$. Jest tak ponieważ:

1° Struktura $ST_1 + ST_2$ jest strukturą pochodną struktury ST_2.
Z kolei struktura η jest strukturą pochodną struktury $ST_1 + ST_2$, a więc struktura η jest strukturą pochodną struktury ST_2 (przechodniość wynika natychmiast z lematu 1).

2° analogicznie dowodzimy, że η jest strukturą pochodną struktury ST_3

3° Korzystając z 1° i 2° oraz z punktu (3) definicji sumy struktur widzimy, że η jest strukturą pochodną struktury $ST_2 + ST_3$.

4° Podobnie jak w 1° dowodzimy, że η jest strukturą pochodną struktury ST_1

5° Z punktów 3° i 4° wynika, że ξ jest strukturą pochodną struktury η (definicja sumy struktur).

Analogicznie udowadniamy, że ξ jest strukturą pochodną struktury η a stąd natychmiast mamy, że $\eta = \xi$

(5) Wynika z definicji sumy struktur (punkt (3)) oraz z założeń twierdzenia.

Aby natychmiast uzyskać sumę struktur, w których niektóre symbole są adresami wystarczy zamienić obydwie struktury na równoważne im, ale zapisane w postaci "jawnej".

W praktyce jednak dodawanie struktur "z adresami" będziemy realizować nieco inaczej:
niech np. $\mu = \alpha \underset{\eta}{\S} \beta$, $\varphi = \alpha A(x) \beta$
Załóżmy, że dodawanie struktur $\mu' = \alpha \eta \beta$ i φ jest wykonalne.
Wtedy oczywiście struktura η musi być postaci $\eta = A(y)$, oraz wykonalne musi być dodawanie struktur $A(x)$ i $A(y)$. Sumę $\mu + \varphi$ napiszemy w postaci:

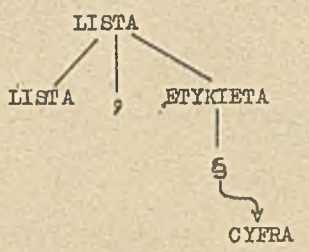
$$\mu + \varphi = \alpha \underset{A(y)+A(x)}{\S} \beta$$

Rozważmy przykład dwóch konkretnych struktur:

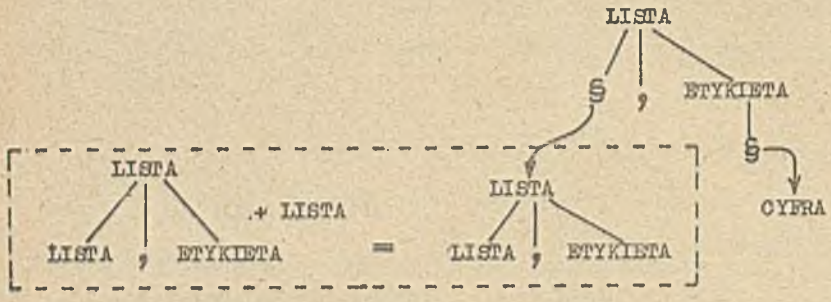
μ = LISTA(§ , ETYKIETA) → LISTA(LISTA, ETYKIETA)

φ = LISTA (LISTA , ETYKIETA(§) → CYFRA

Strukturom tym odpowiadają drzewa:



Sumą tych dwóch drzew będzie drzewo:



A więc:

$\mu + \varphi$ = LISTA (§ , ETYKIETA (§))

LISTA(LISTA , ETYKIETA) → CYFRA

Podana niżej procedura SUMA (μ, η) znajduje sumę dwóch struktur μ i η (zapisanych być może w postaci "adresowej"). Jest to procedura rekursywna.

Opis działania procedury SUMA (μ, η)

Analizuje ona jednocześnie kolejne (poczynając od lewej) symbole obydwóch struktur. Póki analizowane symbole obydwóch struktur są jednakowe i żaden z nich nie jest adresem, procedura dopisuje je do swojego wyniku (początkowo: wynik := ε).

Zastanówmy się teraz jakie są wszystkie możliwe przypadki, gdy albo analizowane symbole S_1 (ze struktury μ) i S_2 (ze struktury η) nie są jednakowe, albo co najmniej jeden z nich jest adresem.

(A) S_1 = (, $S_2 \in A_N \cup A_T \cup \{)\}$ (i przypadek symetryczny tzn.

S_2 = (, $S_1 \in A_N \cup A_T \cup \{)\}$)

Procedura SUMA wczyta wtedy (uruchamiając procedurę SUBTREE(μ)) wszystkie symbole struktury μ poczynając od S_1 a kończąc na takim nawiasie "zamykającym", który "paruje się" z S_1 [S_1 = (] .

Sytuację taką rozpoznajemy po tym, że wtedy liczba nawiasów "otwierających" jest równa liczbie nawiasów "zamykających".

Czynności tej odpowiada znalezienie poddrzewa, którego korzeniem jest symbol stojący w strukturze μ przed symbolem S_1.

Wczytana część struktury μ jest dopisywana do wyniku procedury SUMA (μ,η), a następnie przechodzimy do porównywania symbolu S_2 z tym symbolem struktury μ, który występuje po części wczytanej.

(B) $S_1 = \S_1$, $S_2 \in A_N \cup A_T$ (i przypadek symetryczny)

Znajdowana jest wtedy struktura φ zapisana w miejscu wskazywanym przez $\S_1$, oraz struktura ψ (będąca częścią struktury η) odpowiadająca symbolowi S_2 (uruchamiana jest procedura SUBTREE (η); jeżeli po S_2 nie występuje nawias "otwierający" to $\psi := S_2$).

Następnie obliczana jest suma $\varphi + \psi$ (wołana jest procedura SUMA (φ,ψ)). Wynik tego dodawania umieszczany jest w miejscu wskazywanym przez $\S_1$. Na zakończenie, do dotychczasowego wyniku procedury SUMA (μ,η) dopisywany jest symbol $\S_1$.

Następnymi analizowanymi symbolami będą: następny po S_1 symbol struktury μ oraz pierwszy po ψ symbol struktury η.

(C) $S_1 = \S_1$, $S_2 = \S_2$

Procedura SUMA znajdzie wtedy strukturę φ zapisaną pod adresem $\S_1$, strukturę zapisaną pod adresem $\S_2$ i obliczy sumę $\varphi + \psi$. (tutaj mamy rekurencję: wołana jest procedura SUMA (φ,ψ)). Wynik tego dodawania zostanie umieszczony w miejscu wskazywanym przez $\S_1$, natomiast w miejscu wskazywanym przez $\S_1$ umieszczony zostanie napis $\S_2$.

Do wyniku procedury SUMA (μ,η) dopisywany jest symbol $\S_1$, oraz procedura przechodzi do analizy następnych po S_1 i S_2 symboli struktur μ i η. (zwróćmy uwagę, że takie postępowanie nie ma wpływu na przemienność dodawania $\mu + \eta$).

Jeżeli nie zachodzi żaden z wyżej wymienionych przypadków to oznacza to, że dodawanie jest niewykonalne. Procedura sygnalizuje ten fakt.

W procedurze SUMA używać będziemy funkcji FIRST i SKRÓĆ.

Jeżeli $\mu = a_1a_2a_3\ldots a_n$ (gdzie $a_1 \in A_N \cup A_T \cup \{(,),\S\}$) to:

FIRST $(\mu) = a_1$, SKRÓĆ $(\mu) = a_2a_3\ldots a_n$. Zapis $u \mid v$ oznacza konkatenację napisów u i v.

Zawartość pamięci o adresie § będziemy oznaczać ZAW(§).

```
procedure SUMA (μ,η)
   wynik := ε ;
E0: if FIRST(μ) ≠ FIRST(η) then goto E1;
    wynik := wynik | FIRST(μ);
    μ := SKRÓĆ(μ); η := SKRÓĆ(η); goto E0;
E1: if [FIRST(μ)=(] ∧ [FIRST(η) ∈ A_N ∪ A_T ∪ {)}] then goto E2;
    if [FIRST(μ)=§] ∧ [FIRST(η) ∈ A_N ∪ A_T] then goto E3;
    if [FIRST(μ)=§] ∧ [FIRST(η) = §_2] then goto E4;
    if [FIRST(η)=(] then call SUMA(η,μ);
    if [FIRST(η)=§_2] then call SUMA(η,μ);
    OUTPUT (* DODAWANIE NIEWYKONALNE *); goto E5;
```

```
E2:  φ := SUBTREE(μ);   wynik := wynik|φ ;  goto EO;
E3:  φ := ZAW(§1)   ψ := SUBTREE(η);  ZAW(§1) := SUMA(φ,ψ);
     μ := SKRÓĆ(μ)   wynik := wynik | §1 ;  goto EO
F4:  φ := ZAW(§1) ;  ψ := ZAW(§2) := call SUMA(φ,ψ) ; ZAW(§1) := §2;
     μ := SKRÓĆ(μ)   η := SKRÓĆ(η); wynik := wynik | §1; goto EO;
E5: end  SUMA
```

Procedura ta przy założeniu, że adresowanie w strukturach jest skończone, nie zapętla się.

```
procedure SUBTREE (μ)
   φ := ε ;
   if FIRST(μ) ∈ A_N ∪ A_T then goto AO;
A5: l1 := 0;  l2 := 0;
A4: if FIRST(μ) ∈ A_N ∪ A_T ∪ {§} then goto A1;
   if FIRST(μ) = ) then goto A2;
   l1 := l1 + 1;
A1: φ := φ | FIRST(μ) ;  μ := SKRÓĆ(μ); goto A4
A2: l2 := l2 + 1;  if l1 ≠ l2 then goto A1 else goto A3;
AO:  φ := FIRST(μ);  μ := SKRÓĆ(μ);
    if FIRST(μ) = ( then goto A5 else goto A6;
A3:  φ := φ | FIRST(μ);   μ := SKRÓĆ(μ);
A6: end SUBTREE
```

Rozdział 3. STRUKTURY FORM ZDANIOWYCH W PARSERZE

Realizację dodawania form zdaniowych w parserze omówimy na przykładzie gramatyk afiksowych. Analogicznie wykonywalibyśmy podobne czynności dla niezorientowanych gramatyk afiksowych i dla rozszerzonych gramatyk afiksowych.

3.1. Konstruktor

Konstruktor (po wykonaniu czynności opisanych przy omawianiu parsera dla gramatyk afiksowych) dla każdej hiperprodukcji h postaci:

$$h = X_0(F_1^0,\ldots,F_{N_{X_0}}^0) \to \alpha_0 X_1(F_1^1,\ldots,F_{N_{X_1}}^1)\ \alpha_1 \ldots X_n(F_1^n,\ldots,F_{N_{X_n}}^n)\alpha_n$$

znajduje strukturę każdej formy $D(F_j^i)$ (dla i=0,1,...,n; j=1,2,...,N)

Posługiwać się będzie algorytmem postępowania podanym w Dod.C, rozdz.2.

Następnie wyszukuje się w każdej ze struktur $ST(D(F_j^i))$ te symbole ze zbioru A_N, które odpowiadają zmiennym afiksowym z napisu F_j^i. Będą to symbole, po których nie występuje nawias "otwierający". Każdy taki symbol jest zastępowany w strukturze symbolem § indeksowanym kolejną liczbą całkowitą. Zapamiętuje się odpowiedniość między każdą zmienną afiksową występującą w hiperprodukcji h, a przyporządkowanym jej symbolem $§_i$. (Przy czym tym samym zmiennym przyporządkowujemy te same symbole §).

Np. dla hiperprodukcji:,

$X(L,E) \rightarrow a\ Y\ (E_1)\ b\ Z\ (L,E,C)$

Konstruktor wykona:

(1) $D(L,E)$ = LISTA , ETYKIETA

$D(E_1)$ = ETYKIETA

$D(L,E,C)$ = LISTA,ETYKIETA,CYFRA

(2) $ST\ (D\ (L,E))$ = LISTA (LISTA , ETYKIETA)

$ST\ (D\ (E_1))$ = ETYKIETA

$ST\ (D\ (L,E,C))$ = LISTA(LISTA(LISTA', ETYKIETA) , ETYKIETA(CYFRA))

(3) $ST`(D(L,E))$ = LISTA $(\S_1, \S_2)$

$ST`(D\ (E_1))$ = $\S_3$

$ST`(D(L,E,C))$ = LISTA(LISTA $(\S_1, \S_2)$, ETYKIETA $(\S_4)$)

(4) L przyporządkowujemy $\S_1$

E przyporządkowujemy $\S_2$

E_1 przyporządkowujemy $\S_3$

C przyporządkowujemy $\S_4$

Czynności te są przygotowaniem do przydzielania pamięci zmiennym afiksowym występującym w danej hiperprodukcji.

W dalszym ciągu mówiąc o strukturze $ST(D(F_j^i))$ dla danego napisu $D(F_j^i)$ będziemy mieli na myśli struktury typu takiego jak w powyższym punkcie (3).

3.2. Parser właściwy

Zapis form zdaniowych w postaci ich struktur będzie miał oczywiście wpływ na czynności wykonywane przez parser właściwy.

Przebieg pierwszy

Podczas analizy kolejnej hiperprodukcji h, każdej zmiennej afiksowej L występującej w tej hiperprodukcji zostanie przydzielona pamięć maszyny. W pamięci tej umieścimy napis D L . Następnie adres tej pamięci zostanie umieszczony w miejsce odpowiedniego symbolu $\S_i$ - dla każdej struktury z tej hiperprodukcji (patrz 3.1 konstruktor).

Po wykonaniu tych czynności będziemy wykonywać wszystkie czynności przebiegu pierwszego, które opisaliśmy przy omawianiu gramatyk afiksowych, pamiętając o tym, że:

(1) Zamiast instrukcji $WARTOSC(X_j, adr_j, i) := D(F_i^j)$ (rozdz. 3.2.4) wykonamy instrukcję:
$WARTOSC(X_j, adr_j, i) := ST(D(F_i^j))$

(2) Zamiast instrukcji $(WARTOSC\ X_j, adr_j, i) := WARTOSC(X_j, adr_j, i) +$) wykonamy instrukcję:
$WARTOSC(X_j, adr_j, i) := WARTOSC(X_j, adr_j, i) + ST(D(F_i^j))$

(zwróćmy uwagę, że w czasie wykonywania tego dodawania, będą ulegać zmianom zawartości pamięci przydzielonych odpowiednim zmiennym)

(3) W pamięci przydzielonej hiperprodukcji h zapisane zostaną również adresy pamięci przydzielonej zmiennym występującym w tej hiperprodukcji.

Przebieg drugi

Jeżeli $F_i = \alpha_k L \beta_k$ to, jak łatwo zauważyć, napisowi

$$\Phi(\text{WARTOŚĆ}(X, adr_K, i),\ D(F_i),\ D(\alpha_k))$$

odpowiada zawartość pamięci przydzielonej zmiennej L.

Z kolei obliczenie $P\ (\text{WARTOŚĆ}(Y, adr_2, j), D(G_j), D(\delta_k), L_3)$ (patrz rozdz.3.2.5) realizowane jest przez wstawienie wartości L_3 (jest to suma pewnych struktur) w miejsce wskazywane przez adres zmiennej L.

Przebieg drugi (procedura AKTUAL) będziemy więc realizować pamiętając o wyżej podanym znaczeniu funkcji Φ i P.

Jedyną zmianą jaką wprowadzimy w porównaniu z algorytmem opisującym przebieg drugi (rozdz.3.2.2) jest zmiana dotycząca kroku 6;

Krok 6

Niech WIERZCHOŁEK = (X,adr,i) :

WARTOŚĆ(X,adr,i) := VALUE(X,adr,i) ;

STOS(x):= ZDEJMIJ(STOS(x)) ; wykonaj krok 1.

gdzie VALUE(X,adr,i) oznacza napis otrzymany ze struktury ST = WARTOŚĆ(X,adr,i) w następujący sposób:

(1) Strukturze ST, która jest zapisana w postaci "adresowej" przyporządkowujemy odpowiadającą jej strukturę ST zapisaną w postaci "jawnej"

(2) W strukturze ST "wycieramy" wszystkie symbole $\notin A_T$.

Zmiana wprowadzona w kroku 6 ma za zadanie taką modyfikację parsera, aby na jego wyjściu były nie struktury, ale słowa ze zbioru A_T.

Na zakończenie zauważmy jeszcze, że w związku ze zmianą sposobu zapisu wartości afiks-pozycji będziemy musieli zmienić działanie procedury PREDYKAT(X,adr) (por.rozdz.3.2.5). Działanie to objaśnimy na przykładzie hiperpojęcia predykatowego X(F,G) gdzie: $\delta_X = (X, 2, [A_1, A_2], [1, 0], F_X)$. Procedura PREDYKAT(X,adr) sprowadza się wtedy do wykonania następujących instrukcji:

(1) a := VALUE(X,adr,1);

(2) w := $\tilde{F}_X(a)$

(3) WARTOŚĆ(X,adr,2) := WARTOŚĆ (X,adr,2) + ST(w)

Przy czym jeżeli dodawanie występujące w (3) jest niewykonalne, to sygnalizujemy błąd semantyczny.

Jeżeli natomiast $G \in V_m^*$ to (3) możemy zastąpić prostszą instrukcją:

(3') if G ≠ w then SEMANTIC ERROR

LITERATURA

[1] BLIKLE, A.: Automaty i gramatyki, Warszawa 1971

[2] CROWE, D.: Generating Parsers for Affix Grammars, Comm. ACM 1972,15, s.728-731

[3] DeREMER, F.L.: Practical Translates for LR(k) Languages, Ph.D.Th. Cambridge 1969

[4] DEUSSEN, P.: A Decidability Criterion for Van Vijngaarden Grammars, Acta Informatica 1975, 5, s.353-375

[5] FRANZEN, H.: Ein Parser-Generator für Erweiterte Affix-Grammatiken, Technischen Universität Berlin, 1976

[6] FRANZEN, H.: The Eagle Parser Generator, publ. wewn. Uniwersytetu w Berlinie, 1977

[7] GRZYMKOWSKI, M.: O pewnej klasie gramatyk analitycznych, Praca doktorska, Uniwersytet Warszawski, 1972

[8] HOPCROFT, J.E., ULLMAN, J,D.: Formal Languages and their Relation to Automata, Addison-Wesley, 1969

[9] KOSTER, C.H.A.: Affix Grammars, Mathematische Centrum, Amsterdam, 1971

[10] KNUTH, D.: Semantics of Context-Free Languages, Math. Systems Theory 2, 1968, s.127-145

[11] KNUTH, D.: Examples of Formal Semantics, W: "Symposium on Semantics of Algorithmic Languages", Springer-Verlag, 1971

[12] MALCEV, A.I.: Algoritmy i rekursivnyje funkcji, Moskva, 1964

[13] MAZURKIEWICZ, A.: A Note on Recursively Ennumerable Grammars, Information and Control, 1969, t.14

[14] SINTZOFF, M.: Existence of a van Vijngaarden Syntax for every Recursively Enumerable Set, Annales de la Societe de Bruxelles, 1967, t.81.II

[15] van VIJNGAARDEN, A.: Report on the Algorithmic Language ALGOL 68, Mathematische Centrum, Amsterdam, 1969

[16] WATT, D.: Analysis Oriented Two-Level Grammars, Technical University Berlin, 1975

[17] WATT, D.: The Parsing Problem for Affix Grammars, Acta Informatica 1977, nr 8, s. 1-20

Prace naukowo-badawcze Instytutu Maszyn Matematycznych

LOGICAL VALUES EQUIVALENCES IN FAULT TEST GENERATION

JAN MICHAŁ KLIMOWICZ

Four different fault test generation logic systems, as mentioned below /1-4/ will be presented. It is immediately obvious that sets of values used in these systems have common intersections. Relations between all the values in different systems will be stated and set up in a tabular form. Equivalence proofs are supplied in cases which - in the opinion of the author are not obvious.
The possibility of Akers' calculus extension is noticed, though this does not seem to be necessary for ASC testing.

NANDOR-net: Logical network which can be modeled with and-, nand-, or-, nor-gates
- ASC - asynchronous sequential switching circuit
- SSC - synchronous sequential switching circuit
- FLP - feedback-loop path
- TGS - test generation system
- G - circuit /network/ - faulty-free circuit /network/ - GOOD circuit
- F-circuit /network/ - the same circuit /network/, which does contain a failure - FAULTY circuit
- DC - don't care

Considered calculus list:

1. Roth's D-calculus, 5-valued /1,2/
2. Muth's model, 9-valued /3/
3. Nowicki's /Novi/ calculus, 10-valued /4/
4. Akers'logic system, 16-valued /5/
5. Possible 17-valued calculus, 5-valued model

CONTENTS

INTRODUCTION

In paper [8] one can find an approach which treats a number of values in model used as an important factor in test pattern synthesis and hazard and races analysis theory. The most important and maximal valued model mentioned there is a 4-valued model.
In logic systems proposed for test generation the term model is defined in a slightly different way, and there fore the present author prefers to talk about the number of values in calculus rather than in model. It is possible however to treat both at the same time and this will be attempted here.
Only such logic systems will be considered in which each symbol describes the ordered pair of networks /the GOOD and the FAILED one/ simultaneously using each time only one symbol, which was for the first time attempted by Roth [1] in his D-calculus. In the four calculuses, listed above /1-4/ 26 different symbols were used. It is evident that they do not describe 26 different values, but it may be troublesome to answer, how many they do /see Appendix/. The answer is evidently 16, and these are Akers' values. We shall demonstrate this below and his symbols which are inconvenient will be replaced due to

1/ D-calculus popularity
2/ using string symbols, where one symbol is enough.

The considered calculus list is restricted to heuristic test generation methods as these are used in the majority of TGS-es working in industry [6]. Nevertheless, in the author's opinion it is possible to make some references to algebraic methods well.

CALCULUS PRESENTATION

A briefly description of the logic systems considered follows:

1. Roth's 5-valued calculus

Well known; it uses 5 values: 0,1,D,$\bar{D}$,X. It is based on a 4-valued model; when it is immaterial which one out of 0,1,D,$\bar{D}$ to use, the fifth value, X appears in the calculus /not in the model/.

1. is sufficient, practically necessary for combinational circuits analysis, not sufficient, as Muth has shown for ASC or SSC analysis.

For the sake of simplicity we assume here, that D /$\bar{D}$/ can represent only the ordered pair <1,0> /<0,1>/, i.e. 1/0/ in G circuit and 0/1/ in F circuit, respectively.

2. Muth's 9-valued calculus

Derived for ASC analysis, or may be rather for multiple fault analysis it uses 9 values: 0,1,D,$\bar{D}$,X;G0,F0,G1,F1 /for symbols D,$\bar{D}$,X Muth employs other letters/. One can say that Muth's model is only 3-valued; considering G/F circuits he employs only three values: 0,1,X, so he can obtain only 9-values in calculus and is unable to obtain a contradiction condition.

Muth's values:

G:	0	0	0	x	x	x	1	1	1
F:	0	x	1	0	x	1	0	x	1
symbol:	0	G0	S0	F0	U	F1	S1	G1	1
=	0		$\bar{D}$		X		D		1

x denotes don't care condition in model

X denotes don't care condition /or don't know?/ in calculus

2. is perhaps sufficient* for SSC analysis, perhaps sufficient* for ASC analysis but it has some disadvantage:
it can show the contradiction condition only as a result of checking consistency intersections.

*the author's experience with ASC with FLP cut test generation leads him to consider that it is acceptable in such cases

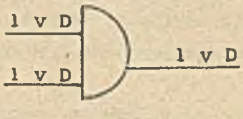
3. Nowicki's 10-valued calculus

Based on D-propagation conditions given in [2] for NANDOR nets it uses following ten values /in parentheses D propagation through NANDOR condition sets are given/:

X = /0,1,D,$\bar{D}$/	0^- = /0,$\bar{D}$/	unordered pairs
D = /D/	0^+ = /0,D/	
$\bar{D}$ = /$\bar{D}$/	1^- = /1,$\bar{D}$/	
0 = /0/	1^+ = /1,D/	
1 = /1/	E = / /	

3. is of course sufficient, very speed for combinational networks, speed, perhaps sufficient for SSC*, may be for ASC* also.

The speed of this calculus is a result of the fact that propagation conditions are treated concurrently for pairs of Roth's values; it means:



/1 v D/ = /1,D/ = $= 1^+$ NOVI

The calculus can also demonstrate the contradiction condition. It does not define the number of values in model /in the meaning used here/, because it doesn't consider a pair of good and faulty circuits.

4. Akers' 16-valued calculus

Well known; it is based on 4-valued model: 0,1,D,$\bar{D}$ - when all the possible 4^2 binary combinations for these values are taken into account the following 16 values in calculus are obtained:

0,1,D,$\bar{D}$,X,G0,F0,G1,F1,K,+,-,$\sim 0^+$,$\sim 0^-$,$\sim 1^+$,$\sim 1^-$

/the last four symbols have nothing in common with Novi 0^+, 0^-, 1^+, 1^-, respectively/

Akers' system designed for NANDOR nets is extremely sophisticated, it's very speedy, demonstrates contradiction immediately and has that wonderfully convenient feature, that the reversals of binary assignments used for coding Akers' alphabet letters /i.e. symbols/ are symbols negations.

4. gives the best procedure, known to the author for combinational circuits testing, but it may suffer a little bit for ASC test generation due to the fact that some symbols are getting very rare in computation. Perhaps it's sufficient for SSC*, perhaps for ASC*. It suffers from the fact, that X does not mean longer DON'T CARE, it means DON'T

KNOW. There is no don't care condition in the calculus.

The last six symbols denote sets of ordered pairs in G/F as follows:

$+ = /\langle 01\rangle, \langle 10\rangle/ = /\bar{D},D/$

$- = /\langle 00\rangle, \langle 11\rangle/ = /0,1/$

$\sim 0^{-} = /\langle 01\rangle, \langle 10\rangle, \langle 11\rangle/ = /\bar{D},D,1/$

$\sim 0^{+} = /\langle 00\rangle, \langle 10\rangle, \langle 11\rangle/ = /0,D,1/$

$\sim 1^{-} = /\langle 00\rangle, \langle 01\rangle, \langle 10\rangle/ = /0,\bar{D},D/$

$\sim 1^{+} = /\langle 00\rangle, \langle 01\rangle, \langle 11\rangle/ = /0,\bar{D},1/$ and

$K = /\langle 00\rangle, \langle 01\rangle, \langle 10\rangle, \langle 11\rangle/ = /0,\bar{D},D,1/$

EQUIVALENCE PROOFS

1. Roth. By our assumption 5 symbols represent the following ordered pairs in G/F:

$0 = \langle 0,0\rangle$, $1 = \langle 1,1\rangle$, $D = \langle 1,0\rangle^{K}$, $\bar{D} = \langle 0,1\rangle^{J}$, $X = \langle x,x\rangle$

2. Muth. $1 \subset 2$. Additional four pairs were already mentioned above in 2.

3. Novi. $2 \cong 3$ /excluding tenth value E/.
Proof /see the list given in 3/:

a/ $\langle 0,x\rangle$ = GO b/ $\langle x,0\rangle$ = FO
0^{-}_{NOVI} = GO 0^{+}_{NOVI} = FO

c/ $\langle x,1\rangle$ = F1 d/ $\langle 1,x\rangle$ = G1
1^{-}_{NOVI} = F1 1^{+}_{NOVI} = G1

Corollary:

We can state now, that Novi model is 3 valued, in fact - if value E is excluded.

4. Akers. $3 \subset 4$. Proof is clear from the pair lists in 4, 1.Roth and 3.Novi

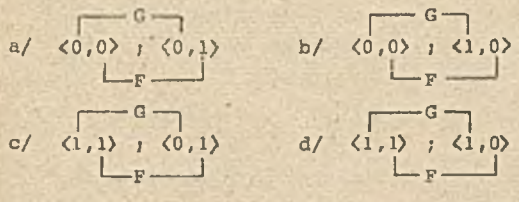
In the first case /for X/ the proof is trivial if one leaves out the don't care/don't know problem.

The next 4 cases have been proved above /Roth/ and presented in tabular form in 2.

The last 4 cases can be proved as follows[N/]:

In G/F $0^{-} = /0;\bar{D}/$, $0^{+} = /0;D/$, $1^{-} = /1;\bar{D}/$, $1^{+} = /1;D/$

represent respectively:



Observing results we can state that for instance in case a/ the two considered pairs represent 0 or 1 in F and 0 in G. Denoting /0 or 1/ in G by x as we have done before /Muth/ we have in G/F:

MULTIVALUED CALCULUS EQUIVALENCE TABLE

We can now set up a table gathering all the four mentioned calculuses. Thus we have:

ASSIGN.	SYMBOLS					JK NOVI COMMENTS	REVERSE JK ASSIGNMENTS NOT	REV. JK CODE
CCC1	AKERS	JK CODE	MUTH	NOVI				
CC11								
CCCC	K	K		E		NOC	0000	K
CCC1	1	1	1	1		NOC	1000	0
CC10	1	0	51	0		NOC	0100	$\bar{D}$
CC11	1	51	51	1		1vD	1100	00
C100	C	$\bar{D}$	50	$\bar{D}$		NOC	0010	0
C1C1	(1)	F1	F1	1		1vD	1010	F0
C110	+	+	+			NOC	0110	+
C111	-0	N0				NOC	1110	N1
1CCC	0	0	0	0		NOC	0001	1
1CC1	-	-	-			NOC	1001	-
1010	(0)	F0	F0	0		0vD	0101	F1
1C11	-0	N0				NOC	1101	N0
11CC	0	00	00	0		0vD	0011	01
11C1	-1	N0				NOC	1011	$\bar{N}0$
1110	-1	N1				NOC	0111	N0
1111	X	X	X	X		NOC	1111	X

/NOC - no comments, 1vD = 1 or D/.

The first two columns coincide with the Akers' table transfo ion /the author had no Akers' paper at hand when programming the computer - and it is a computer output/ but the order of assigniny is as good as the original one- the reverse assignment is its negation - - the reader can check this in the last two columns - several similiar assignments are pos-

J/ - one can use symbol U /UP/ for $\bar{D}$, or H which seems to be better in 01 string, mnemonic HIGH /not the best English, however/

K/ - DOWN convention, D means DOWN: 1 in G and 0 in F

N/ - Relations between 4 Novi and Muth values were discussed with Mr. Nowicki

sible. In the last column sixteen symbols can be clearly seen, which the author proposes to use - they seemed to be most popular, clear and easy to programme:

0,1,D,$\overline{D}$,?,GO,FO,G1,F1,+,-,NO,N1,ND,$N\overline{D}$,K

We'll give now all the sixteen names in full:

0	both in G and F
1	both in G and F
D	DOWN
$\overline{D}$	UP
?	DON'T KNOW
GO	0 IN GOOD circuit
FO	0 IN FAULTY circuit
G1	1 IN GOOD circuit
F1	1 IN FAULTY circuit
+	CHANGE between G and F
-	NO CHANGE between G and F
NO	NOT 0, all others
N1	NOT 1, all others
ND	NOT D, all others
$N\overline{D}$	NOT $\overline{D}$, all others
K	CONTRADICTION

At the left of the table we have 5 Roth's values.

POSSIBLE AKERS' CALCULUS EXTENTION

Assuming 5-rather than 4-valued model one can achieve 32 values, as follows:

0	$\overline{D}$	X	D	1	
0	0	0	0	0	K
0	0	0	0	1	VALUES IDENTICAL WITH AKERS ONES
.	.	.	.	.	
.	.	.	.	.	E.G.JK SYMBOLS
.	.	.	.	.	
				0	
1	1	0	1	1	DON'T KNOW, mnemonic proposed:?
0	0	1	0	0	DON'T CARE, X
0	0	1	0	1	
0	0	1	1	0	
0	0	1	1	1	
1	0	1	0	0	
1	0	1	0	1	
1	0	1	1	0	
1	0	1	1	1	
0	1	1	0	0	
0	1	1	0	1	
0	1	1	1	0	
0	1	1	1	1	
1	1	1	0	0	
1	1	1	0	1	
1	1	1	1	0	
1	1	1	1	1	

Fortunately, the underlined assignment is the only one, which can be useful. The remaining fifteen values are in practice identical with X, because they contain X, which covers four other values: 0,$\overline{D}$,D,1. So in fact this calculus represents only 17 values.

May be for some classes of networks this will turn out to be useful in practice.
The 17-th value, X can be used for instance to denote initial don't care conditions on pseudo inputs in sequential networks with FLP cut.

In G/F:
? = ⟨?,?⟩
X = ⟨x,x⟩

Note, that the reverse assignment remains the assignment negation. This calculus should be sufficient for ASC testing.

CONCLUSION

We gathered 5 different calculuses with 27 different symbols used in test pattern generation theory in TGS-es.
It turns out, that ten of them are identical to some others and all are included in Akers' calculus /except DC value-whose lack is Akers' main disadvantage/. We proved that.
17 and 16 values calculuses seem to be good enough for ASC test analysis.
It is interesting, that similiar amount of values /16/ we meet in hazard and races analysis calculus designed for ASC by Hławiczka and Badura [7].

References

[1] ROTH J.P.: Diagnosis of Automata Failures: a Calculus and a Method, IBM Journal on Res. and Dev., July 1966

[2] ROTH J.P., BOURICIUS W.G., SCHNEIDER P.R.: Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits, IEEETC, Oct., 1967, t.EC-16, nr 5, s. 567-579

[3] MUTH P.: A Nine Valued Circuit Model to Generate Tests for Sequential Circuits, FTC-5 Digest of Papers, s. 43-49

[4] NOWICKI M., SAPIECHA K.: Algorytm generacji testów diagnostycznych metodą równoległego uczulania ścieżek, Polish FTC-1 Proceedings Wisła 1976, s. 214-221

[5] AKERS S.B.: A Logic System for Fault Test Generation, FTC-5 Digest of Papers, s.37-42

[6] RAULT J.C.: La Detection et la Localisation des Defauts dans les Circuites Logiques, FTC-5, Digest of Papers, s. 17-23

[7] HŁAWICZKA A., BADURA D.: Multi-valued Algebra to Diagnostic of Critical Hazards, Komunikaty Naukowe ISS, Nr. 25, Katowice 1978

[8] SAPIECHA K.: Algebraiczne modele układów przełączających i ich zastosowania w dziedzinie analizy i diagnostyki struktur cyfrowych, Politechnika Warszawska, Prace Naukowe, ELEKTRONIKA NR 35, praca habilitacyjna, Warszawa 1978.

Appendix

I. Four calculuses original symbols presentation:

1. Roth : 0,1,D,$\overline{D}$,X
2. Muth : 0,1,S1,S0,U,G0,G1,F0,F1
3. Novi : 0,1,D,$\overline{D}$,X,0^+,0^-,1^+,1^-
4. Akers: 0,1,0^-,0^+,1^-,1^+,(0),(1),X,$\sim 0^-$,$\sim 0^+$, $\sim 1^-$,$\sim 1^+$,+,-,K

Proposed: 0,1,D,$\overline{D}$,+,-,G0,G1,F0,F1,N0,N1,ND,$N\overline{D}$, ?,X,K

A. ŁUKASIEWICZ

METODA AUTOMATYCZNEJ KONSTRUKCJI ANALIZATORA SKŁADNIOWEGO DLA JĘZYKÓW GENEROWANYCH PRZEZ GRAMATYKĘ AFIKSOWĄ

Streszczenie

Wprowadzone w 1970 r. przez C.H.A. Kostera gramatyki afiksowe są wygodnym aparatem do definiowania języków programowania. Ze względu na dużą moc gramatyk afiksowych /w ogólności można za ich pomocą wygenerować dowolny język rekurencyjnie przeliczalny/ stosowane są one do opisu nie tylko składni bezkontekstowej, ale również i semantyki definiowanego języka. W związku z dużą atrakcyjnością gramatyk afiksowych jako narzędzia do opisu języków, w niniejszej pracy podjęto próbę podania metody automatycznej konstrukcji analizatora składniowego dla języków generowanych przez gramatyki afiksowe. Metoda ta dotyczy szerokiej podklasy klasy gramatyk afiksowych, tzw. "dobrze zdefiniowanych gramatyk afiksowych" /wcześniej proponowane metody C.Kostera i D. Watta obejmowały stosunkowo wąskie podklasy/.

Podano algorytm konstrukcji analizatora składniowego dla danej /dowolnej/ gramatyki dobrze zdefiniowanej. Algorytm ten opiera się na wprowadzonym w pracy pojęciu "dodawania form zdaniowych".

Aby ułatwić czytelnikowi zapoznanie się z proponowaną metodą pracę podzielono na trzy części: przedstawienie metody, dowody podawanych twierdzeń oraz przykłady ilustrujące ważniejsze z wprowadzonych pojęć.

МЕТОД АВТОМАТИЧЕСКОЙ ПОСТРОЙКИ СИНТАКСИЧЕСКОГО ПАРСЕРА ДЛЯ ЯЗЫКОВ ГЕНЕРОВАННЫХ АФФИКСНОЙ ГРАММАТИКОЙ

Резюме

Аффиксные грамматики, определённые в 1970 году Костером, являются удобным средством для формального описания языков программирования. Из-за того, что аффиксные грамматики обладают большой мощью /в общем с их помощью можно определить любой рекурсивно перечислимый язык/ они применяются для описания не только контекстно-свободного синтаксиса, но также для семантики определяемого языка.

В связи с большой привлекательностью аффиксных грамматик как средства описания языков, мы попытались найти метод автоматического построения синтаксического парсера для языков, определённых с помощью аффиксных грамматик. Метод этот относится к широкому подклассу класса аффиксных грамматик, так называемых "правильно определённых аффиксных грамматик" /ранее предлагаемые методы Костера и Ватта относились к довольно узким подклассам/.

Предложено алгоритм конструкции синтаксического парсера для данной любой правильно определённой грамматики - алгоритм этот образуется на введенном в работу понятии "сложения нетерминальных цепочек".

Для облегчения читателю усвоить предлагаемый метод, работу разделено на три части: 1 - иллюстрация метода, 2 - доказательства введенных теорем, 3 - примеры, изображающие более важных из введенных понятий.

THE METHOD OF AUTOMATIC CONSTRUCTION OF THE SYNTACTIC ANALYZER FOR THE LANGUAGES GENERATED BY AFFIX GRAMMAR

Summary

Affix grammars introduced in 1970 by C.H.A. Koster are a suitable means to define formally the meaning of programming languages. This advantage of affix grammars resolved the author to find a new and general method of automatic construction of parsers for languages defined by these grammars. In contradistinction to other methods known from the literature, the proposed methods comprises a broad subclass of the class of affix grammars, named "well defined affix grammars".

This paper contains the parser construction algorithm for arbitrary well defined affix grammars. The algorithm bases on the notion of "adding sentential forms" whose definition precedes the algorithm itself.

To enable the reader to pass throught the work more easily, it has been divided into three parts: presentation of the method, theorem proofs, and application examples.

J.M. KLIMOWICZ

RÓWNOWAŻNOŚĆ WARTOŚCI LOGICZNYCH W RACHUNKACH GENERACJI TESTÓW

Streszczenie

Przedstawiono cztery różne systemy wartości logicznych używane w generacji testów sieci logicznych. Jest oczywiste, że zbiory wartości używanych w tych systemach mają wspólne przecięcia. Pokazano relacje między wartościami w różnych systemach i zestawiono te wartości w tabeli. W przypadkach, które zdaniem autora nie są oczywiste, podano dowody równoważności wartości. Odnotowano możliwość pewnego rozszerzenia rachunku Akersa, choć rozszerzenie takie nie wydaje się konieczne dla testowania ASC.

ОБ ЭКВИВАЛЕНТНОСТИ ЛОГИЧЕСКИХ ЗНАЧЕНИЙ В ГЕНЕРАЦИИ ТЕСТОВ

Резюме

Представлены 4 различные исчисления используемые для генерации тестов цифровых сетей. Очевидно, что у них существуют взаимные пересечения, но не так просто их точно указать. Показываются взаимоотношения между разными логическими значениями и составляется их в таблице. Даются доказательства эквивалентности для случаев, которые автор считает не очевидными.

LOGICAL VALUES EQUIVALENCES IN FAULT TEST GENERATION

Summary

Four different fault test generation logic systems, as mentioned below /1-4/ will be presented. It is immediately obvious that sets of values used in these systems have common intersections. Relations between all the values in different systems will be stated and set up in a tabular form. Equivalence proofs are supplied in cases which - in the opinion of the author are not obvious. The possibility of Akers' calculus extension is noticed, though this does not seem to be necessary for ASC testing.

Wyk."Tłumacz-Tekst",Al.Ujazd.26
U-499 450/II/A4 ic

Cena 60 zł

Druk okładki IMM zam. 125/80 n. 450.