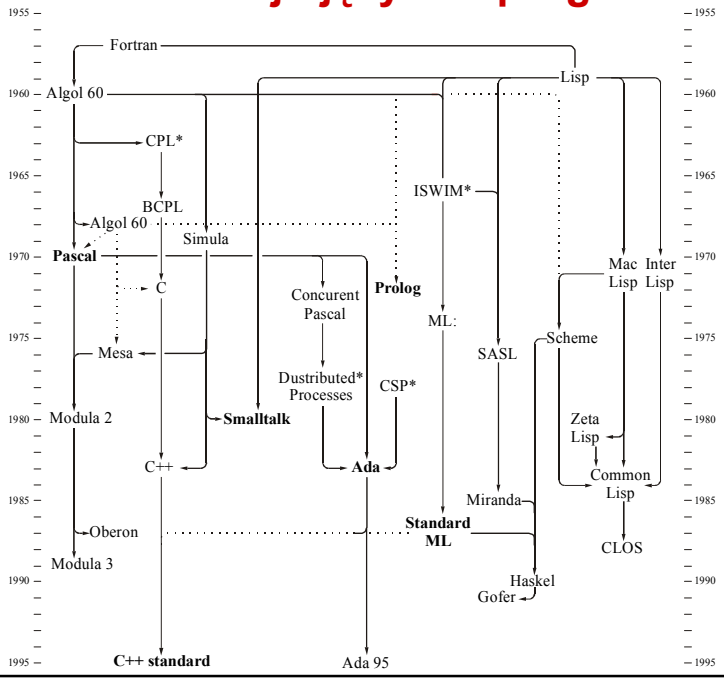
Katedra Informatyki Stosowanej
Politechnika Łódzka

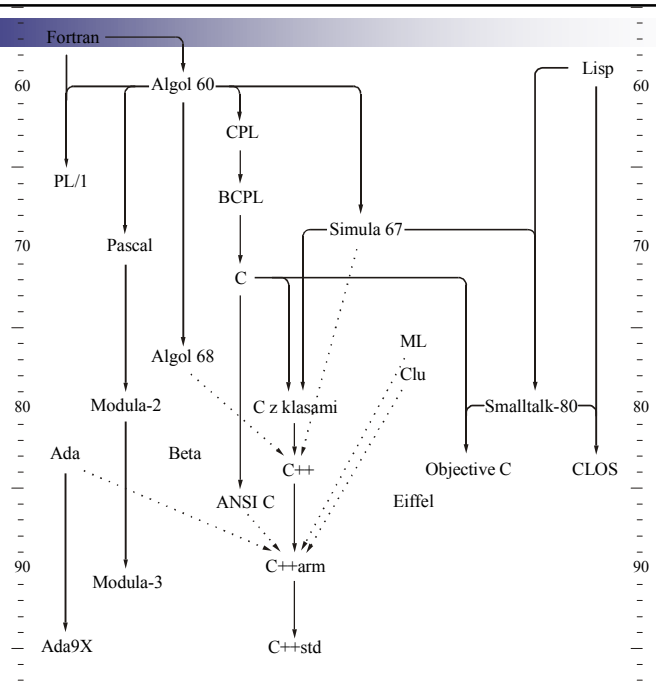
Lingwistyka Matematyczna

Języki formalne i gramatyki
Analiza zdań

© dr hab. inż. Lidia Jackowska-Strumiłło

Historia rozwoju języków programowania





Zależność między językiem C++ a innymi językami programowania w ujęciu historycznym

Definicja języka

- Podstawą każdego języka jest **słownik**. W językach formalnych elementy słownika (słowa) nazywamy **symbolami** (podstawowymi).
- Ciąg słów zbudowany według określonych reguł nazywamy **zdaniem**.
- Zbiór reguł lub formuł, które określają zbiór formalnie poprawnych zdań nazywamy **gramatyką** lub **składnią**. Składnia lub struktura języka pozwala na określenie, czy dany ciąg słów jest zdaniem.
- Składnia i **semantyka** (znaczenie) języka są ze sobą blisko powiązane.

Język formalny

- **Język formalny** to podzbiór zbioru łańcuchów skończonych złożonych z elementów zbioru skończonego nazywanego **alfabetem**.
- **Język formalny** jest generowany przez **gramatykę formalną**, przy czym ten sam język może być generowany przez wiele gramatyk.

PRZYKŁAD 1

<zdanie> ::= <podmiot> <orzeczenie>
<podmiot>::= kwiaty | gwiazdy
<orzeczenie>::= kwitną | świecą

Zdania wyprowadzone z tej gramatyki:

- kwiaty kwitną
- gwiazdy świecą
- gwiazdy kwitną
- kwiaty świecą

Notacja BNF (Backus-Naur-Form)

- **Symbol początkowy**: <zdanie>,
- **symbole pomocnicze** (nieterminalne): <zdanie>, <podmiot>, <orzeczenie>;
- **symbole końcowe** (terminalne): kwiaty, gwiazdy, kwitną, świecą;
- **metasymbole** notacji BNF: <, >, ::=, |
- **produkcje** – reguły, poprzez które można zdefiniować język.

PRZYKŁAD 1

```
S ::= A B
A ::= x | y
B ::= z | w
```

Zdania wyprowadzone z gramatyki:

xz, yz, xw, yw.

*Stosując **regułę kolejnych wyprowadzeń** można zdanie języka wyprowadzić z **symbolu początkowego:***

$S \rightarrow AB \rightarrow xB \rightarrow xw$

$S \rightarrow AB \rightarrow yB \rightarrow yz$ $\qquad S \xrightarrow{*} yz$

Gramatyka Chomsky'ego
– matematyczna definicja języka

1. Język ***L = L(T,N,P,S)*** jest definiowany przez:
   - ***T*** – słownik symboli końcowych;
   - ***N*** – zbiór symboli pomocniczych (kategorii gramatycznych);
   - ***P*** – zbiór produkcji (reguł syntaktycznych);
   - ***S*** – symbol początkowym, należący do ***N,*** nazywany też głową języka.

Gramatyka Chomsky'ego, cd.

2. Język ***L(T,N,P,S)*** jest zbiorem ciągów symboli końcowych ξ, które mogą być wyprowadzone z ***S*** zgodnie z podaną poniżej regułą 3.

   $L = \{ \xi \mid S \overset{*}{\Rightarrow} \xi \quad \text{i} \quad \xi \in T^* \}$

   gdzie: litery greckie oznaczają ciągi symboli,

   T^* oznacza zbiór wszystkich ciągów symboli z ***T***.

Gramatyka Chomsky'ego, cd.

3. Ciąg δ_n może być wyprowadzony z ciągu δ_0 wtedy i tylko wtedy, gdy istnieją ciągi $\delta_1, \delta_2, \ldots, \delta_{n-1}$ takie, że każdy ciąg δ_i może być bezpośrednio wyprowadzony z δ_{i-1} zgodnie z podaną poniżej regułą 4.

$$(\delta_0 \overset{*}{\Rightarrow} \delta_n) \leftrightarrow ((\delta_{i-1} \rightarrow \delta_i) \text{ dla } i = 1, \ldots, n)$$

Gramatyka Chomsky'ego, cd.

4. Ciąg η może być bezpośrednio wyprowadzony z ciągu ξ wtedy i tylko wtedy gdy istnieją ciągi $\alpha, \beta, \xi', \eta'$ takie, że:
   a) $\xi = \alpha\ \xi'\ \beta$
   b) $\eta = \alpha\ \eta'\ \beta$
   c) ***P*** zawiera produkcję $\xi' ::= \eta'$

PRZYKŁAD 2

Gramatyka, w której za pomocą rekursji można wyprowadzić nieskończenie wiele zdań za pomocą skończonego zbioru produkcji.

```
S ::= xA
A ::= z | yA
```

Zbiór zadań jakie mogą być wyprowadzone z symbolu początkowego S:

xz
xyz
xyyz
xyyyz
.....

PRZYKŁAD

Gramatyka, za pomocą której można zdefiniować liczby całkowite:

```
<liczba całkowita> ::= <cyfra>
                     |<liczba całkowita><cyfra>
<cyfra> ::= 0|1|2|3|4|5|6|7|8|9
```

Hierarchia Chomsky'ego

Hierarchia Chomsky'ego dzieli języki na cztery typy.

Języki regularne

Języki regularne – języki formalne generowane przez wyrażenia regularne, tzn. definiowane za pomocą produkcji o postaci:

A ::= a

A ::= aB

A ::= ε

gdzie: ***A*** ∈ ***N, B*** ∈ ***N,*** a ∈ ***T***

Języki bezkontekstowe i kontekstowe

Język bezkontekstowy – język formalny zdefiniowany przez zbiór produkcji bezkontekstowych, tzn. produkcji o postaci:

$A ::= \xi$ i $(A \in N, \xi \in (N \cup T)^*)$

Język kontekstowy – język formalny zdefiniowany przez zbiór produkcji o postaci:

$\alpha A \beta :: = \alpha \xi \beta$ dla niepustego ξ

Języki rekursywne i rekursywnie przeliczalne

Język rekursywny – język formalny, dla którego istnieje algorytm rozstrzygający, czy dany łańcuch należy do tego języka, czy nie.

Język rekursywnie przeliczalny – język formalny, dla którego istnieje algorytm decydujący, czy dany łańcuch należy do tego języka, czy nie; przy tym algorytm ten musi zakończyć działanie z odpowiedzią pozytywną dla łańcuchów należących do języka, zaś dla łańcuchów spoza języka może dać odpowiedź negatywną lub w ogóle nie dać odpowiedzi.

Analiza zdań

- Analiza polega na rozbiorze struktur zdaniowych i zdań.
- Zadaniem teorii **analizy składniowej** jest opracowanie algorytmów rozbioru języka o skomplikowanych regułach strukturalnych.

Rozbiór zstępujący „Top down”

Rozbiór zstępujący lub generacyjny (ang. *Top down*) polega na wyznaczeniu kroków generacyjnych od symbolu początkowego do zdania końcowego.

Przykład 1

<zdanie> ::= <podmiot> <orzeczenie>
<podmiot>::= kwiaty | gwiazdy
<orzeczenie>::= kwitną | świecą

Czy zdanie "gwiazdy świecą" należy do języka?

<zdanie>	*gwiazdy świecą*
<podmiot><orzeczenie>	*gwiazdy świecą*
gwiazdy <orzeczenie>	*gwiazdy świecą*
<orzeczenie>	*świecą*
świecą	*świecą*
- - -	**- - -**

Przykład 2

S ::= xA
A ::= z | yA

Czy zdanie "xyyz" należy do języka?

S	xyyz
xA	xyyz
A	yyz
yA	yyz
A	yz
yA	yz
A	z
z	z
--	--

Przykład 3

S:: = A | B
A:: = xA | y
B:: = xB | z

Dokonaj rozbioru zdania "xxxz"

S	xxxz
A	xxxz
xA	xxxz
A	xxz
xA	xxz
A	xz
xA	xz
A	z

Gramatyki klasy LL(1)

REGUŁA 1:

Dla zadanej gramatyki:

$$A:: = \xi_1 \mid \xi_2 \mid \ldots \mid \xi_n$$

zbiory pierwszych symboli w zdaniach, które mogą być wyprowadzone z ξ_i muszą być rozłączne, tzn.

$pierw(\xi_i) \cap pierw(\xi_j) = \varnothing$ dla wszystkich $i \neq j$.

Gramatyki klasy LL(1)

Zbiór ***pierw(ξ)*** jest zbiorem wszystkich symboli końcowych, które mogą wystąpić na pierwszej pozycji w zdaniach wyprowadzonych z ξ. Zbiór ten może być wyznaczony według następujących reguł:

1. jeśli pierwszy symbol argumentu jest symbolem końcowym, to

   $pierw(a\xi) = \{a\}$

2. jeśli pierwszy symbol jest symbolem pomocniczym i istnieje produkcja

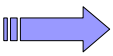
   $A ::= \alpha_1 | \alpha_2 | \ldots | \alpha_n$

   to

   $pierw(A\xi) = pierw(\alpha_1) \cup pierw(\alpha_2) \cup \ldots \cup pierw(\alpha_n)$

Przykład 3

S:: = A | B
A:: = xA | y
B:: = xB | z



S:: = C | xS
C:: = y | z

Dokonaj rozbioru zdania "xxxz"

S	xxxz
xS	xxxz
S	xxz
xS	xxz
S	xz
xS	xz
S	z
C	z
z	z
--	--

Lewostronna faktoryzacja

Produkcję postaci:

$$A :: = \alpha\xi_1 \mid \alpha\xi_2 \mid \ldots \mid \alpha\xi_n$$

należy zastąić produkcjami:

$$A :: = \alpha A'$$
$$A' :: = \xi_1 \mid \xi_2 \mid \ldots \mid \xi_n$$

Przykład 4

Dana jest gramatyka:

$$S :: = Ax$$
$$A :: = x \mid \varepsilon$$

gdzie: ε - pusty ciąg symboli.

Dokonaj rozbioru zdania "x"

S	x
Ax	x
xx	x
x	--

Gramatyki klasy LL(1)

REGUŁA 2:

Dla każdego symbolu $A \in N$, z którego można wyprowadzić pusty ciąg symboli ($A \xrightarrow{*} \varepsilon$), zbiór jego pierwszych symboli musi być rozłączny ze zbiorem symboli, które mogą następować po dowolnym ciągu wyprowadzonym z A, tzn.

$$pierw(A) \cap nast(A) = \varnothing$$

Gramatyki klasy LL(1)

Zbiór ***nast****(A)* wyznacza się następująco:

dla każdej produkcji P_i postaci:

$$X ::= \xi A \eta$$

przez S_i oznaczamy *pierw*(η_i); suma wszystkich zbiorów S_i tworzy zbiór *nast(A).* Jeśli z co najmniej jednego η_i możemy wyprowadzić pusty ciąg symboli, to zbiór *nast(X)* musi być również włączony do *nast(A)*.

Rekurencja

Produkcja:

$$A ::= B \mid AB$$

opisuje zbiór ciągów: B, BB, BBB, …

Jej użycie zgodnie z regułą 1 jest niedozwolone, ponieważ:

$$pierw(B) \cap pierw(AB) = pierw(B) \neq \varnothing$$

Rekurencja

Produkcja:

$$A ::= \varepsilon \mid AB$$

opisuje zbiór ciągów: ε, B, BB, BBB, …

Jej użycie zgodnie z regułą 2 jest niedozwolone, ponieważ:

$$pierw(A) = pierw(B)$$

a zatem

$$pierw(A) \cap nast(A) \neq \varnothing.$$

Eliminacja lewostronnej rekurencji

I i II Reguła gramatyczna nie pozwalają na stosowanie definicji lewostronnie rekurencyjnych.

Problem ten można rozwiązać poprzez:

- zamianę rekurencji lewostronnej na prawostronną

$$A ::= \varepsilon \mid BA$$

- zastąpienie rekursji iteracją.

Zapis {B} w notacji EBNF oznacza iterację czyli powtórzenie symbolu B zero, jeden, dwa, ... lub nieskończenie wiele razy. Produkcja:

$$A ::= \{B\}$$

opisuje zbiór ciągów: ε, B, BB, BBB, …

Przykład 5

Dana jest gramatyka:

S ::= A | S - A
A ::= a | b | c

→ *a – b – c = ((a – b) – c)*

gdzie: {a, b, c, -} $\in$ T

S ::= A | A - S
A ::= a | b | c

→ *(a – (b – c))*

Te dwie gramatyki nie są równoważne semantycznie

Eliminacja lewostronnej rekurencji

Produkcję postaci:

$$A ::= A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m$$

można zastapić produkcjami:

$$A ::= \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_m A'$$
$$A' ::= \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_n A' \mid \varepsilon$$

Przykład 5

Dana jest gramatyka:

S ::= A | S - A
A ::= a | b | c

→ *a – b – c = ((a – b) – c)*

S ::= A {– A}
A ::= a | b | c

→ *a – b – c*

S ::= A S'
S' ::= -AS' | ε
A ::= a | b | c

→ *a – b – c*

Te gramatyki są równoważne semantycznie

Drzewo wyprowadzeń

- Drzewo wyprowadzeń jest to hierarchiczna struktura, która reprezentuje w sposób graficzny wyprowadzenie zdania z gramatyki.
- Korzeń drzewa ma etykietę głowy języka. Gałęzie są symbolami pomocniczymi, a liście symbolami końcowymi lub pustymi. Struktura gałęzi reprezentuje produkcje. Gałąź główna ma etykietę lewej strony produkcji, a gałęzie z niej wyprowadzone etykiety prawej strony produkcji w kolejności od strony lewej do prawej.

Przykład 6

Należy wyprowadzić liczbę 123.456 z gramatyki:

<liczba rzeczywista> :: = <część całkowita> . <ułamek>
<część całkowita> :: = <cyfra> | <część całkowita> <cyfra>
<ułamek> :: = <cyfra> | <cyfra> <ułamek>
<cyfra> :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Niejednoznaczność składni.

Problem niejednoznaczności gramatyki, występuje wtedy, gdy generuje ona zdania, które mają więcej niż jedno drzewo wyprowadzeń.

Przykład 7

Należy wyprowadzić zdanie 1 - 0 - 1 dla gramatyki:

E:: = E – E | 0 | 1

Dla zdania 1- 0 -1 można narysować dwa różne drzewa wyprowadzeń.

Przykład 8

Kolejna niejednoznaczność występuje w języku C i Pascal dla instrukcji warunkowej IF.

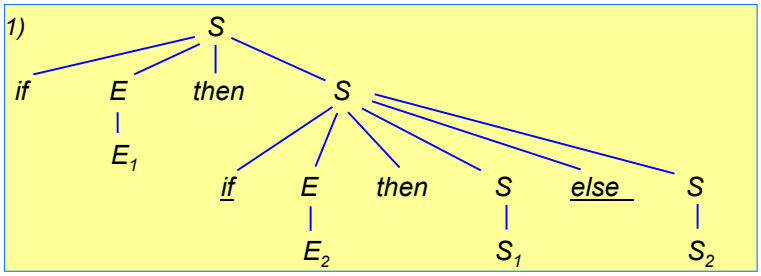
Jest ona spowodowana wystąpieniem produkcji:

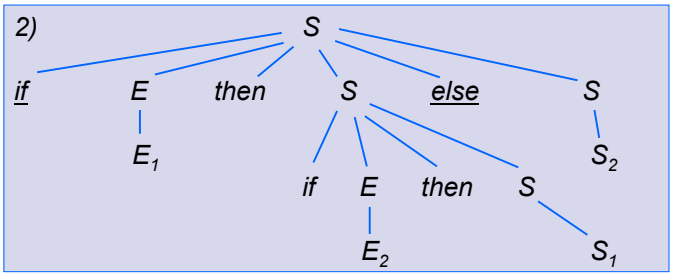
S :: = if E then S
S :: if E then S else S

Pascal

Dla zdania if E_1 then if E_2 then S_1 else S_2 można wyprowadzić dwa różne drzewa.

if E_1 then if E_2 then S_1 else S_2





Eliminacja niejednoznaczności if-else

Problemem niejednoznaczności if-else w językach Pascal i C został rozwiązany w ten sposób, że słowo kluczowe else jest kojarzone z najbliżej występującym słowem kluczowym if.

W tym przypadku rozpatrywane zdanie będzie interpretowane tak jak przedstawia to drzewo 1.

Eliminacja niejednoznaczności if-else w języku Modula-2

```
S ::= <empty>
    | stmt
    | if E then SL end
    | if E then SL else SL end
SL ::= SL; S | S
```

Pascal: *if E_1 then if E_2 then S_1 else S_2*

Modula-2: *1º. if E_1 then if E_2 then S_1 else S_2 end end*
2º. if E_1 then if E_2 then S_1 end else S_2 end

Pascal i Modula-2

Pascal:

```
if E1 then S1
        else if E2 then S2
                    else if E3 then S3
                                else S4
```

Modula-2:

```
if E1 then S1
        else if E2 then S2
                    else if E3 then S3
                                else S4 end
                end
end
```

Modula-2

```
if E1 then S1
        elsif E2 then S2
                    elsif E3 then S3
                                else S4
end
```

S ::= if E then SL { elsif E then SL } [else SL] end
SL ::= SL; S | S

Zestawienie oznaczeń stosowanych w notacji BNF oraz w jej zmodyfikowanej wersji - MBNF

	ZAPIS W MBNF	ZNACZENIE	ZAPIS W BNF (EBNF)
1	=	jest zdefiniowane jako	::=
2	|	lub	|
3	.	zakończenie formuły	brak jawnego oznaczenia
4	[x]	0 lub jednokrotne powtórzenie napisu x	nie używa się metasymboli [ ]
5	{x}	0 lub wielokrotne powtórzenie napisu x	{x} w EBNF
6	(x | y ... | z)	dowolny z napisów: x, y, ··,z	nie używa się metasymboli ( )
7	"a"	symbol terminalny (z alfabetu języka)	nie ujmuje się w cudzysłów
8	ciąg-małych-liter	symbol nieterminalny (zmienna metajęzykowa)	zmienne ujmuje się w nawiasy ostre, nie ogranicza się do małych liter i niekoniecznie używa się łącznika między wyrazami

Separatory i terminatory

Separatory – oddzielają elementy
Terminatory – pojawiają się za każdym elementem, np. zdaniem

Pascal:

```
S ::= <empty>
    | stmt
    | begin SL end
SL ::= SL; S | S
```

Pascal: *begin S_1; S_2; S_3 end*
begin S_1; S_2; S_3; end
begin ;S_1;;;; S_2; S_3;; end

Instrukcja pusta

Pascal:

```
S ::= <empty>
    | stmt
    | if E then S
    | if E then S else S
    | begin SL end
    | while E do S
SL ::= SL; S | S
```

Modula-2:

```
S ::= <empty>
    | stmt
    | if E then SL end
    | if E then SL else SL end
    | begin SL end
    | while E do SL end
SL ::= SL; S | S
```

Pascal: *if E then ; S_1;* *- błąd semantyczny*

if E then S_1; else S_2; *- błąd syntaktyczny*

Modula-2: *if E then S_1; S_2 end*

if E then S_1; S_2 else S_3; S_4 end