Context free grammars - I

Prof. A. Morzenti

LIMITS OF REGULAR LANGUAGES

Simple languages such as $L = \{ a^n b^n ,\ n > 0 \}$

representing basic syntactic structures like

begin begin ... begin ... end ... end end

are ***not*** regular

e.g., $b^+ e^+$ does not satisfy the constraint $\#b = \#e$

$(be)^+$ does not ensure nesting

GRAMMARS – a more powerful means to definefine languages

through ***rewriting rules***

language phrases generated through repeated application of rules

The grammar is characterized by its set of rules

Example – Language of ***palindromes***

$L = \{\ uu^R \mid\ u \in \{a, b\}^*\ \} = \{\varepsilon, aa, bb, abba, baab, \ldots, abbbba, \ldots\ \}$

a palindrome is…

$pal \rightarrow \varepsilon$	an empty palindrome
$pal \rightarrow a\ pal\ a$	a palindrome surrounded by two *a*'s
$pal \rightarrow b\ pal\ b$	a palindrome surrounded by two *b*'s

A chain of ***derivation steps***:

$pal \Rightarrow a\ pal\ a \Rightarrow\ ab\ pal\ ba \Rightarrow abb\ pal\ bba \Rightarrow abb\varepsilon bba = abbbba$

look out: distinguish the two *metasymbols*

$\rightarrow$ separates the left and right part of a rule

$\Rightarrow$ derivation ***relation*** (rewriting)

Example: a non-empty ***list of*** palindromes, ex: *abba bbaabb aa*

$list \rightarrow pal\ list$

$list \rightarrow pal$

$pal \rightarrow \varepsilon \qquad pal \rightarrow a\ pal\ a \qquad pal \rightarrow b\ pal\ b$

non terminal symbols:

- *list* (axiom, or start symbol)
- *pal* (defines the component palindrome substrings)

derivation of the string *abba bbaabb aa*
(spaces added for readability, nonterm to be expanded underlined)

$list \Rightarrow pal\ \underline{list} \Rightarrow pal\ pal\ \underline{list} \Rightarrow \underline{pal}\ pal\ pal \Rightarrow a\ \underline{pal}\ a\ pal\ pal$
$\Rightarrow ab\ \underline{pal}\ ba\ pal\ pal \Rightarrow abba\ \underline{pal}\ pal \Rightarrow abba\ b\ \underline{pal}\ b\ pal$
$\Rightarrow abba\ bb\ \underline{pal}\ bb\ pal \Rightarrow abba\ bba\ \underline{pal}\ abb\ pal$
$\Rightarrow abba\ bbaabb\ \underline{pal}$
$\Rightarrow abba\ bbaabb\ a\ \underline{pal}\ a \Rightarrow abba\ bbaabb\ aa$

CONTEXT-FREE GRAMMAR
(BNF - Backus Normal Form – TYPE 2 – FREE GRAMMAR)

defined by four entities

1. *V, **non terminal alphabet**,* is the set of nonterminal symbols
2. *Σ, **terminal alphabet**,* is the set of the symbols of which phrases/sentences are made
3. *P,* is the set of ***rules*** or ***productions***
4. $S \in V$, is the specific nonterminal, called the ***axiom*** (***Start***), from which derivations start

a rule is an ordered pair (X, α), with $X \in V$ and $\alpha \in (V \cup \Sigma)^*$

$(X, \alpha) \in P$ is usually written as $X \rightarrow \alpha$

rules with the same nonterminal X: $X \rightarrow \alpha_1$, $X \rightarrow \alpha_2$, … $X \rightarrow \alpha_n$

can be written in brief as $X \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$ or $X \rightarrow \alpha_1 \cup \alpha_2 \cup \ldots \cup \alpha_n$

$\alpha_1, \alpha_2, \ldots, \alpha_n$ are called the ***alternatives*** of X

To avoid confusion,

the metasymbols '→', '|', '∪', 'ε' cannot be symbols,

terminal and nonterminal alphabets must be disjointed

NOTATIONS to distinguish terminal and nonterminal symbols

- angle brackets:

  <if-phrase> → *if* *<cond>* *then* < *if-phrase* > *else* < *if-phrase* >

- ***bold italic***:

  if-phrase → ***if*** *cond* ***then*** *if-phrase* ***else*** *if-phrase*

- quotation marks ‘ ’

  if-phrase → *‘if’* *cond* *‘then’* *if-phrase* *‘else’* *if-phrase*

- upper- versus lower-case:

  $F \rightarrow if\ \ C\ \ then\ \ D\ \ else\ \ D$

WE USUALLY ADOPT THESE CONVENTIONS:

- terminal characters - $\{a, b, \ldots\}$
- nonterminal characters - $\{A, B, \ldots\}$
- strings $\in \Sigma^*$ (only terminals) - $\{r, s, \ldots, z\}$
- strings $\in (V \cup \Sigma)^*$ (terminals and nonterminals) - $\{\alpha, \beta, \ldots\}$
- strings $\in V^*$ (only **nonterminals**) - σ

TYPES OF RULES (RP = right part, LP = left part)

Terminal: RP contains only terminals, or the empty string $\rightarrow u \mid \varepsilon$

Empty (*or* ***null***): RP is empty $\rightarrow \varepsilon$

Initial / Axiomatic: LP is the axiom $S \rightarrow$

Recursive: LP occurs in RP $A \rightarrow \alpha A \beta$

Left-recursive: LP is prefix of RP $A \rightarrow A \beta$

Right-recursive: LP is suffix of RP $A \rightarrow \alpha A$

Left- *and* ***right-recursive***: conjunction of two previous cases $A \rightarrow A \beta A$

Copy *or* ***categorization***: RP is a single nonterminal $A \rightarrow B$

Linear: at most one nonterminal in RP $\rightarrow u B v \mid w$

Right-linear (type 3): linear + nonterminal is suffix $\rightarrow u B \mid w$

Left-linear (type 3): linear + nonterminal is prefix $\rightarrow B v \mid w$

Homogeneous normal: *n* nonterminals or just one terminal $\rightarrow A_1 \ldots A_n \mid a$

Chomsky normal (or homogeneous of degree 2): two nonterminals or just one terminal $\rightarrow BC \mid a$

Greibach normal: one terminal possibly followed by nonterminals $\rightarrow a \sigma \mid b$

Operator normal: two nonterminals separated by a terminal (operator); more generally, strings devoid of adjacent nonterminals $\rightarrow A a B$

DERIVATIONS AND GENERATED LANGUAGE

Def. of ***derivation relation*** ‘$\Rightarrow$’

for $\beta, \gamma \in (V \cup \Sigma)^*$ β derives γ for grammar G, $\beta \underset{G}{\Rightarrow} \gamma$, or $\beta \Rightarrow \gamma$, iff

$A \rightarrow \alpha$ is a rule of G, $\beta = \delta A \eta$, $\gamma = \delta \alpha \eta$

the rule $A \rightarrow \alpha$ is applied in that derivation step, and α ***reduces to*** A

power, reflexive and transitive closure of ‘$\Rightarrow$’ $\boxed{\beta_0 \overset{n}{\Rightarrow} \beta_n \quad \beta_0 \overset{*}{\Rightarrow} \beta_n \quad \beta_0 \overset{+}{\Rightarrow} \beta_n}$

If $A \overset{*}{\Rightarrow} \alpha$ $\alpha \in (V \cup \Sigma)$ called ***string form generated by G***

If $S \overset{*}{\Rightarrow} \alpha$ α called ***sentential*** or ***phrase form***

If $A \overset{*}{\Rightarrow} s$ $s \in \Sigma^*$, s is called ***phrase*** or ***sentence***

LANGUAGE GENERATED FROM NONTERMINAL ***A*** OR FROM AXIOM ***S***

$$\boxed{\begin{array}{l} L_A(G) = \left\{ x \in \Sigma^* \mid A \overset{+}{\Rightarrow} x \right\} \\ L(G) = L_S(G) = \left\{ x \in \Sigma^* \mid S \overset{+}{\Rightarrow} x \right\} \end{array}}$$

Example: Grammar G_l generates the structure of a book: it contains

- a front page (f)
- a series (denoted by the nonterm. A) of one or more chapters
- every chapter starts with the title (t) and contains a sequence (B) of one or more lines (l)

$$\boxed{\begin{array}{l} S \rightarrow fA \\ A \rightarrow AtB \mid tB \\ B \rightarrow lB \mid l \end{array}}$$

from A one generates the string form $tBtB$ and the phrase $tlltl \in L_A(G_l)$

from S one generates the phrase forms $fAtlB$, $ftBtB$

The language generates from B is $L_B(G_l) = l^+$

$L(G_l)$, being generated by the context free grammar G_l, is ***context free*** or ***free***

Notice: it is also regular, because it is defined also by the regular expressione $f(tl^+)^+$

Two grammars G and G' are equivalent if they generate the same language, that is, $L(G) = L(G')$

G_l

$$S \to fX$$
$$X \to XtY \mid tY$$
$$Y \to lY \mid l$$

G_{l2}

$$S \to fA$$
$$A \to AtB \mid tB$$
$$B \to Bl \mid l$$

The first rules have the same n.t., only renamed

$Y \overset{n}{\Rightarrow} l^n$ in G_l and $B \overset{n}{\Rightarrow} l^n$ in G_{l2} generate the same language $L_B = l^+$

ERRONEOUS GRAMMARS AND USELESS RULES: all nonterminals must

- be ***reachable*** from S, and hence contribute to the generation of the language
- be ***defined***: eventually generate "something" –we are not interested in the $\varnothing$ lang.

a grammar G is ***clean*** (or ***reduced***) iff the following conditions hold:

1. Every nonterminal A is ***reachable*** from the axiom, that is, there exists a derivation

$$S \stackrel{*}{\Rightarrow} \alpha A \beta$$

2. Every nonterminal A is ***defined***, that is, it generates a non-empty language

$$L_A(G) \neq \varnothing$$

NB: $L_A(G) = \varnothing$ includes also the case when no derivation from A terminates with a terminal string s (i.e. $s \in \Sigma^*$), e.g.: $P = \{ S \to aA,\ A \to bS \}$

GRAMMAR CLEANING: two steps algorithm:

The FIRST PHASE builds the set ***UNDEF*** of undefined nonterminals

The SECOND PHASE builds the set of unreachable nonterminals

PHASE 1- We first build the ***complement*** set $DEF = V \setminus UNDEF$

DEF is ***initialized*** from the *terminal rules* (the n.t. that immediately generate a terminal string)

$$\boxed{DEF := \left\{ A \mid (A \to u) \in P, \text{with } u \in \Sigma^* \right\}}$$

The following ***update*** is repeated until a ***fixed point*** is reached :

$$DEF := DEF \cup \{ B \mid (B \to D_1 D_2 \ldots D_n) \in P \wedge \forall i \; (D_i \in DEF \cup \Sigma) \}$$

Every D_i is already in *DEF* or it is a terminal

In algebra, the ***fixed point*** of a transformation is an object that is transformed into itself

At each iteration, two cases can occur:

1. New nonterm. are found having the RP all with defined nonterm. or term., or
2. No new nonterm. is found, algorithm terminates (a ***fixed point*** has been reached)

nonterminals $\in$ *UNDEF* are eliminated

PHASE 2 - The computation of the set of nonterminals reachable from S consists of finding paths from ***S*** to other nonterminals in the graph of the ***produce*** relation, defined as

$$\boxed{A \text{ produce } B \quad \text{iff} \quad \left(A \to \alpha B \beta\right) \in P, \text{ with } A \neq B \quad \alpha, \beta \text{ any string}}$$

C is ***reachable*** from ***S*** iff there exists, in the graph, a path from ***S*** to ***C***

nonterminals that are not reachable can be eliminated often another requirement is added for cleanness condition of a grammar ***G*** :

3. ***G*** must not allow for ***circular derivations***: they are not essential and introduce ***ambiguity***

if $A \overset{+}{\Rightarrow} A$ then

if the derivation $A \overset{+}{\Rightarrow} x$ is possible

then also $A \overset{+}{\Rightarrow} A \overset{+}{\Rightarrow} x$ and many other similar ones exist

NB: ***circular derivations*** must not be confused with ***recursive rules*** and ***derivations*** **!!**

EXAMPLES OF GRAMMARS THAT ARE NOT CLEAN

1) $\{ S \to aASb, A \to b \}$ (S does not generate any phrase, i.e., $L(S)=\varnothing$)

2) $\{ S \to a, A \to b \}$ (A not reachable) ($\{ S \to a\}$ equiv. clean version)

3) $\{ S \to aASb \mid A, A \to S \mid b \}$ (circular on S and A) ($\{S \to aSSb \mid b \}$ equiv. clean)

circularity can also derive from an empty rule

$$\boxed{X \to XY \mid \ldots \quad Y \to \varepsilon \mid \ldots}$$

NB: even if clean, a grammar can have ***redundant rules*** (leading to ambiguity)

(1,4) and (2,5) generate
the same phrases

$$\boxed{\begin{array}{ll} 1.\ S \to aASb & 4.\ A \to c \\ 2.\ S \to aBSb & 5.\ B \to c \\ 3.\ S \to \varepsilon & \end{array}}$$

RECURSION AND LANGUAGE ***INFINITY***

most interesting languages are infinite

but what determines the ability of a grammar to generate an infinite language?

infinity of the language implies unbounded phrase length

therefore the grammar must be recursive

a ***derivation*** $A \overset{n}{\Rightarrow} xAy \quad n \geq 1$ is ***recursive***

if $n = 1$ it is ***immediately recursive***

A is a ***recursive nonterminal***

if $x = \varepsilon$ then it is ***left recursive*** (l.r. derivation, l.r. nonterminal)

if $y = \varepsilon$ then it is ***right recursive*** (r.r. derivation, r.r. nonterminal)

NB: circularity and recursiveness are (very) different notions

a grammar may be recursive (admit recursive derivations) but not circular

circular $\Rightarrow$ recursive but it is **not** the case that recursive $\Rightarrow$ circular

necessary and sufficient condition for language $L(G)$ to be infinite,

assuming G clean and devoid of circular derivations,

is that G allows for recursive derivations

necessary condition: if no recursive derivation was possible,
then every derivation would have limited length hence $L(G)$ would be finite

sufficient condition:

$A \overset{n}{\Rightarrow} xAy$ implies $A \overset{+}{\Rightarrow} x^m A y^m$

for any $m \geq 1$ with $x, y \in \Sigma^*$ not both empty

Furthermore G clean implies

$S \overset{*}{\Rightarrow} uAv$ (A reachable from S)

and $A \overset{+}{\Rightarrow} w$ (derivation from A terminates successfully)

therefore there exist nonterminals that generate an infinite language

$$S \overset{*}{\Rightarrow} uAv \overset{+}{\Rightarrow} ux^m A y^m v \overset{+}{\Rightarrow} ux^m w y^m v, (\forall m \geq 1)$$

a grammar is devoid of recursive derivations

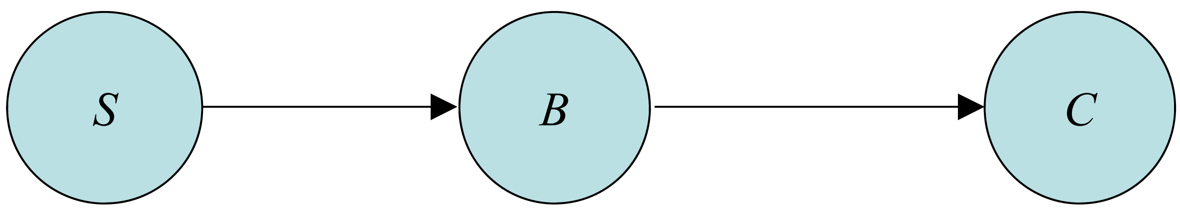
if and only if

the graph of the ***produce*** relation has no circuits

Example

$$\boxed{\begin{aligned} &S \to aBc \\ &B \to ab \mid Ca \\ &C \to c \end{aligned}}$$

finite language: $\{ aabc, acac \}$



Example (arithmetic expressions)

$$G = \left(\underbrace{\{E, T, F\}}_{non\ term.}, \underbrace{\{i, +, *,), (\}}_{term.}, \underbrace{P}_{productions}, \underbrace{E}_{axiom} \right)$$

$$P = \{E \to E + T \mid T, \quad T \to T * F \mid F, \quad F \to (E) \mid i\}$$

$$L(G) = \{i,\ i+i+i,\ \ i*i,\ \ (i+i)*i,\ \ldots\}$$

F (factor) has indirect recursion (non immediate)

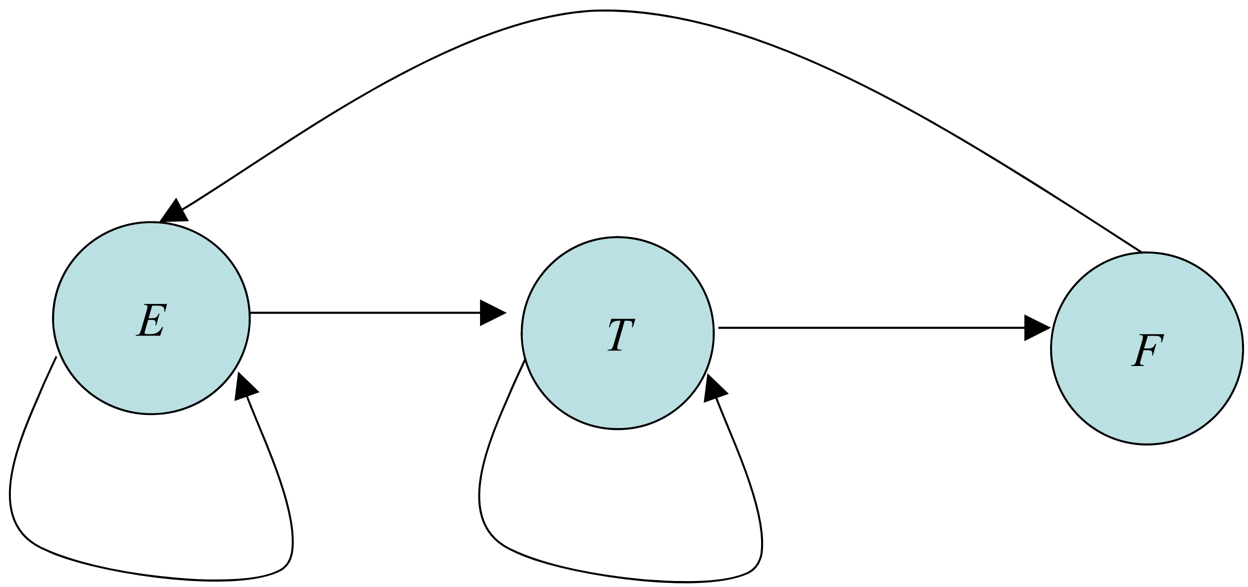
E (expression) has immediate left recursion and non immediate recursion

T (term) has immediate left recursion and non immediate recursion

G is clean, recursive, noncircular, hence the generated language is infinite grammar has recursions

$\Leftrightarrow$

the graph of the produce relation has circuits



$$G = \left(\left\{E, T, F\right\}, \left\{i, +, *,), (\right\}, P, E\right)$$

$$P = \{E \to E + T \mid T, \quad T \to T * F \mid F, \quad F \to (E) \mid i\}$$