1 Basic Formal Grammars

1.1 Types of grammars

1.1.1 Thue and semi Thue systems

The first systems for describing languages date back from around 1920, with the works of the mathematicians Axel Thue (norwegian, 1863-1922) and Emil Post (american, born in Poland, 1897 - 1954) in the purpose of developing tools for mathematical logic, in particular for solving what is called the *word problem* for monoids and semi-groups.

Definition 1 *A* monoid *is a set M provided with a product "." which is associative and has a neutral element. Let us recall what* associative *means :*
"." is said to be associative on M if and only if:

$$\forall x, y, z \in M, (x.y).z = x.(y.z)$$

A neutral element e is an element of M which is such that:

- $\forall x \in M, x.e = e.x = x$

A perfect example of a monoid is given by the set of all words on some alphabet A. An alphabet A can be any non empty finite set ($\{\mathtt{a}, \mathtt{b}, \mathtt{c}, ..., \mathtt{z}\}$ as well as {0, 1, 2, ..., 9} or only {0, 1}, or only $\{|\}$, or $\{\spadesuit, \heartsuit, \diamondsuit, \clubsuit\}$, or {Peter, Paul, Mary} and so on). A *word* on an alphabet A is a finite sequence of elements of A. As a sequence we could write a word as, for instance $(\mathtt{a}, \mathtt{a}, \mathtt{c}, \mathtt{e}, \mathtt{v}, \mathtt{i})$ - on $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, ..., \mathtt{z}\}$- or ($\spadesuit, \spadesuit, \clubsuit, \heartsuit$) - on $\{\spadesuit, \heartsuit, \diamondsuit, \clubsuit\}$, but we prefer to write it simply as : `aacevi` or $\spadesuit\spadesuit\clubsuit\heartsuit$.
The set of all words on an alphabet A is noted A^* (where the star is known as the *Kleene star*) (from the name of another famous logician of the first half of the twentieth century).
On words, a natural operation (a product) is defined: *concatenation.* In terms of sequences, the concatenation ($\frown$) of the sequence σ and the sequence τ in that order (that is σ first and then τ) is the sequence realized by putting all the elements of τ after the last element of σ, in the order they have in τ. For instance:

$$(x_1, x_2, ..., x_n) \frown (y_1, y_2, ..., y_m) = (x_1, x_2, ..., x_n, y_1, y_2, ..., y_m)$$

In formal terms the result is the sequence $(z_1, z_2, ..., z_{n+m})$ such that:

- $\forall i = 1, ..., n : z_i = x_i$

- $\forall j = 1, ..., m : z_{n+j} = y_j$

In the word notation, we have of course:

$$x_1x_2...x_n \frown y_1y_2...y_m = x_1x_2...x_ny_1y_2...y_m$$

that is for instance: `ruta` $\frown$ `baga` = `rutabaga`.
Associativity is expressed by the fact that for every words τ, σ and μ:

$$\tau \frown (\sigma \frown \mu) = (\tau \frown \sigma) \frown \mu$$

This property allows us to speak of the product $\tau \frown \sigma \frown \mu$ without specifying the place of the parentheses, since the result of the product is independent of such places.
Among all the finite sequences of elements of A, there is the sequence with 0 element, that is the *empty sequence*, that we shall call the *empty word* (or *empty string* since sometimes words are also called *strings*), that we shall note ϵ.
It is easy to show that for all word σ on A, we have: $\epsilon \frown \sigma = \sigma \frown \epsilon = \sigma$.
In other terms, ϵ is precisely the neutral element of the monoid $M = (A, .)$.
The so called *word problem* for monoids is the following: suppose we have some rules which are able to transform a word v into a word w, given two words σ and τ belonging to the same monoid, is it possible to transform σ into τ by means of these rules? If it was possible, such a result could be extended to other formal systems and could be translated into: given an axiom A and a formula ϕ is it possible to go from A to ϕ only by using the deduction rules of that system? It is the reason why the word problem appeared so attractive for logicians in the thirties. Unfortunately, it was proven undecidable.
But what is a *Rewriting Rules System*?

Definition 2 *Let R be a binary relation on A^*, that we shall write "$\rightarrow$". An element $(u, v) \in R$ is called a* rewriting rule*, it is noted : $u \rightarrow v$.*

If R is symmetric (that is, if we have $u \rightarrow v$, we have also $v \rightarrow u$), it is called a Thue system. If not, it is only a semi-Thue system (this is the general case).
A rewriting rule system allows to transform words into other words in the following way:

Definition 3 *A word σ is rewritten in one step into a word τ by the rewriting rules system R if and only if:*

- *there is a rule $u \rightarrow v \in R$,*
- *there exists two words : s, t such that:*
  - $\sigma = s \frown u \frown t$
  - $\tau = s \frown v \frown t$

We may then construct the reflexive transitive closure of this relation that we shall write $\rightarrow^*$: it is the relation between words such that $u \rightarrow^* v$ if and only if:

- there exist an integer $k \geq 0$,

- there exists a sequence of words $(v_0, v_2, ..., v_k)$ such that:
  - $v_0 = u$
  - $v_k = v$
  - $\forall i = 0...k-1, v_i \rightarrow v_{i+1}$

Notice that if $k = 0$, the sequence is reduced to (v_0), the first condition becomes : $v_0 = u$ and $v_0 = v$, therefore $u = v$, the condition $0 \leq i \leq k-1$ is always false, thus resulting in $(0 \leq i \leq k-1) \Rightarrow (v_i \rightarrow v_{i+1})$ trivially true. We conclude that a particular case is the case of $u = v$, therefore: $u \rightarrow^* u$, it is why we said this relation is not only the transitive closure but also the reflexive one.

Examples :

1) let A^* the set of words on the latin alphabet $\{\texttt{A}, \texttt{B}, \texttt{C}, ..., \texttt{Z}\}$ and let R the following semi-Thue system:

$$\begin{array}{lcl}
\texttt{RM} & \rightarrow & \texttt{RB} \\
\texttt{AL} & \rightarrow & \texttt{OL} \\
\texttt{OR} & \rightarrow & \texttt{ER} \\
\texttt{RM} & \rightarrow & \texttt{AM} \\
\texttt{AA} & \rightarrow & \texttt{RA} \\
\texttt{BE} & \rightarrow & \texttt{DE} \\
\texttt{AR} & \rightarrow & \texttt{RO} \\
\texttt{CO} & \rightarrow & \texttt{AL}
\end{array}$$

show that:

$$\begin{array}{c}
\texttt{ARMOR} \rightarrow^* \texttt{RODER} \\
\texttt{ARMOR} \rightarrow^* \texttt{RAMER} \\
\texttt{CALOR} \rightarrow^* \texttt{ALLER}
\end{array}$$

2) Another example of semi Thue system is provided by the famous ”MU-puzzle” (Douglas Hofstadter):

- the alphabet is : $\{\texttt{M}, \texttt{I}, \texttt{U}\}$
- the rules are:
  1. Add a `U` to the end of any word ending in `I`
  2. Double any (sub)word after the `M`
  3. Replace any `III` with `U`
  4. Remove any `UU`
- or in terms of rewriting rules (where x and y are variables which may denote any word)
  1. $x\texttt{I} \rightarrow x\texttt{IU}$
  2. $\texttt{M}x \rightarrow \texttt{M}xx$

3. $x\mathtt{III}y \rightarrow x\mathtt{U}y$
4. $x\mathtt{UU}y \rightarrow xy$

For instance, the following vertical list is a *derivation*, that is a rewriting of the first word into the last one step by step by using the rules:

MI
MII
MIIII
MIIIIU
MIIIIUIIIIU
MIUUIIIIU
MIIIIIU
MUIIU

We may write therefore:

$$\mathtt{MI} \rightarrow^* \mathtt{MUIIU}$$

The question is : given two words u and v shall we be always able to determine whether $u \rightarrow^* v$ or not? For instance if $u = \mathtt{MI}$ and $v = \mathtt{IU}$ we can immediately answer ”no”.... simply because a very elementary reasoning makes us to see that starting from a word beginning with M, we cannot arrive at a word not beginning by it. But, more difficult, what happens if $u = \mathtt{MI}$ and $v = \mathtt{MU}$? In other words, do we have $\mathtt{MI} \rightarrow^* \mathtt{MU}$ or not?
An obvious way of trying to answer to that problem consists in starting with MI applying rules repeatidly until we get MU. If we arrive at MU, OK, we shall have won! But what if after a long work we have not yet reached it? Shall we say that $\mathtt{MI} \not\rightarrow^* \mathtt{MU}$, or shall we think that we still need to work? The difficulty of that system comes from the fact that some rules are *extensive* (they expand the left side term) and others are *retracting* (they retract the length of the left side word). If it was not the case, that is if all the rules were extensive (or at least keep the same length), we should necessarily arrive at a situation where all the words of a given length that can be produced would have been obtained. It would remain only to see whether our word MU belongs to that set or not. But here we cannot apply this reasoning.
It is in fact possible to show that MU cannot be obtained from MI, but the solution is not so simple, it relies on the determination of an *invariant* which characterizes all the words obtainable from MI. After determining such an invariant, it simply remains to show that MU does not verify it.
Having seen such a problem here, we begin to understand why the word problem may be a difficult one. Actually, Emil Post showed that it is *undecidable*. That means that there exists *no* general algorithm such that given a rewriting rules system on a monoid M, and given two words u and v, it would be possible to show that $u \rightarrow^* v$, or not, simply by applying this algorithm.

1.1.2 Chomsky's grammars

General Grammars

Among all the thuian systems, there is a kind of rewriting system which revealed to have good applications in linguistics. These particular systems are called *grammars*. They correspond to the famous specification made by Chomsky : a *formal grammar* must be some finite device able to generate an infinite set of expressions. The idea of ”grammar” is based on the splitting of the alphabet into two disjoint subsets: a subset of *terminal* expressions and a subset of *non terminals*. Let us call them respectively V_T and V_N. The notion of ”alphabet” will often be replaced by that of *vocabulary* : in fact, abstractly, both concepts are the same, they simply denote some finite non empty set! But in the first case the elements are interpreted as *letters* (and their sequences are interpreted as *words*) while in the second case, they are interpreted themselves as *words* (or *lexemes*) and their sequences as *sequences of words*. The purpose of a grammar is to discriminate among all the sequences of words a priori possible on a given terminal vocabulary, what are those which correspond to *sentences*.

Nevertheless for the time being, let us still work with an alphabet A, but partitionned into a set of terminal letters A_T and a set of non terminal symbols A_N. Let us consider for example the two following sets :

- $A_T = \{\texttt{a}, \texttt{b}, \texttt{c}\}$
- $A_N = \{\mathrm{S, T, U}\}$

and the set of rules:

$$\begin{array}{lcl}
S & \rightarrow & ST \\
STT & \rightarrow & \texttt{abb} \\
STU & \rightarrow & \texttt{abc} \\
TT & \rightarrow & U \\
UU & \rightarrow & U \\
\texttt{b}T & \rightarrow & \texttt{b} \\
SU & \rightarrow & \texttt{ac}
\end{array}$$

We may have the following sequences of rewriting steps, all starting from S:
$S \rightarrow ST \rightarrow STT \rightarrow STTT \rightarrow STU \rightarrow \texttt{abc}$
$S \rightarrow ST \rightarrow STT \rightarrow \texttt{abb}$
$S \rightarrow ST \rightarrow STT \rightarrow SU \rightarrow \texttt{ac}$
$S \rightarrow ST \rightarrow STT \rightarrow STTT \rightarrow \texttt{abb}T \rightarrow \texttt{abb}$
$S \rightarrow ST \rightarrow STT \rightarrow STTT \rightarrow SUT \rightarrow \texttt{ac}T$
We may notice that all these sequences or *derivations* stop at some point where they can no longer be continued since there is no rule applying beyond that point. Some words obtained by such derivations are only made of terminals, some others (the last one for instance) still contains one non -terminal. If we define the language generated by a grammar as the set of all the words (or sequences of words) which may be obtained by such a derivation and which contain only terminal symbols, we can say that, with regards to this ”grammar”, the generated language contains the words `abc`, `abb`, `ac`. In fact this grammar is not very productive since these words seem to be the only ones to belong to its generated language!

Let us simply add two rules:

$$\begin{array}{lcl}
TU & \rightarrow & \texttt{bc} \\
T\texttt{b} & \rightarrow & \texttt{bb}
\end{array}$$

we may observe that the generated language becomes infinite! Why is it the case? We may see that the first rule may be applied any number of times:

$$S \to ST \to STT \to STTT \to ....$$

thus producing any number of T, then it is always possible to change the last TT into U, if not preceded by S, and then to change TU into $\mathtt{bc}$. Then $T\mathtt{b}$ may be changed into $\mathtt{bb}$. In fact we may generate any word of the form ab....bc, with an arbitrary number of $\mathtt{b}'s$. Of course this grammar is weird and complex: it mixes many kinds of rules, some have long sequences of non terminal symbols on their left, some mix non terminals and terminals either on their left hand side or on their right hand side. It seems appealing to make some housekeeping among these rules!

Context-free grammars

Among all the grammars, one kind was taken to be particularly interesting: the kind where rules are limited to have a left hand side consisting in a single non terminal symbol, that is, are of the form:

$$X \to \phi$$

where $X \in V_N$ and $\phi \in (V_T \cup V_N)^*$. Moreover one non terminal symbol is distinguished: it is the non terminal from which all the derivations start, it is therefore called the *start symbol* or *axiom*.

These grammars are *Chomsky* grammars. They are also called (for reasons we shall examine soon) *Context-Free* grammars, more generally : *Phrase Structure* Grammars.

Context-freeness

These grammars are said to be context-free because they contrast with grammars where rules obey a more general form : $\alpha X \beta \to \alpha u \beta$ where $\alpha, u, \beta \in (V_T \cup V_N)^*$ and $X \in V_N$. This form is more general because we may retrieve from it the particular form of the rules in a Chomsky grammar: it suffices to make $\alpha = \beta = \epsilon$. In fact what says this kind of rule is:

X rewrites as u, but only in the *context* determined by the word α on the left of X and the word β on its right. If we denote such a context by the expression "$\alpha\_\_\beta$", we may say : X rewrites as u in the context $\alpha\_\_\beta$, or:

$$X \to u/_{[\alpha\_\_\beta]}$$

As we may see, the case of a rule like $X \to u$ is the particular case obtained with $\alpha = \beta = \epsilon$, in other terms, when there is no context taken into account. It is the reason why, in this particular case, we say that X rewrites as u independently of any context, and we say that the rule itself is "context-free". In the more general case, the rule is said to be *context sensitive*.

Constituency

Let us take as an example a toy-grammar of a very limited fragment of English.

$$\begin{array}{lcl}
S & \rightarrow & NP^\frown VP \\
NP & \rightarrow & PN \\
NP & \rightarrow & Det^\frown N \\
VP & \rightarrow & Vi \\
VP & \rightarrow & Vt^\frown NP \\
PN & \rightarrow & \texttt{mary} \\
N & \rightarrow & \texttt{cat} \\
Vi & \rightarrow & \texttt{sleeps} \\
Vt & \rightarrow & \texttt{owns} \\
Det & \rightarrow & \texttt{a}
\end{array}$$

Implicitely, this grammar has the following partition of symboles:

$$\begin{array}{lcl}
V_N & = & \{S, NP, VP, PN, N, Vi, Vt\} \\
V_T & = & \{\texttt{mary}, \texttt{cat}, \texttt{sleeps}, \texttt{owns}, \texttt{a}\}
\end{array}$$

and its axiom (start symbol) is S.
Two derivations in this grammar are:

1. $$\begin{array}{l}
S \\
NP^\frown VP \\
PN^\frown VP \\
\texttt{mary}^\frown VP \\
\texttt{mary}^\frown Vi \\
\texttt{mary}^\frown\texttt{sleeps}
\end{array}$$

2. $$\begin{array}{l}
S \\
NP^\frown VP \\
PN^\frown VP \\
\texttt{mary}^\frown VP \\
\texttt{mary}^\frown Vt^\frown NP \\
\texttt{mary}^\frown\texttt{owns}^\frown NP \\
\texttt{mary}^\frown\texttt{owns}^\frown Det^\frown N \\
\texttt{mary}^\frown\texttt{owns}^\frown\texttt{a}^\frown N \\
\texttt{mary}^\frown\texttt{owns}^\frown\texttt{a}^\frown\texttt{cat}
\end{array}$$

There are obviously other possible derivations in this grammar, yielding for instance:
(1) $\texttt{mary}^\frown\texttt{owns}^\frown\texttt{mary}$
(2) $\texttt{a}^\frown\texttt{cat}^\frown\texttt{owns}^\frown\texttt{a}^\frown\texttt{cat}$
(3) $\texttt{a}^\frown\texttt{cat}^\frown\texttt{owns}^\frown\texttt{mary}$
(4) $\texttt{a}^\frown\texttt{cat}^\frown\texttt{sleeps}$
Actually, using no meaning consideration at all, it would be difficult to avoid deriving sentences which are dubious with regards to meaning like (1), (2) or (3). Notice nevertheless that we get

neither

(5) $^*\texttt{mary}^\frown\texttt{sleeps}^\frown\texttt{a}^\frown\texttt{cat}$

nor:

(6) $^*\texttt{mary}^\frown\texttt{owns}$

(5) and (6) would violate elementary grammatical rules according to which verbs enter constructions in conformity with their expected arguments (or complements). *Sleep* takes no complement (it is an intransitive verb), while *owns* must necessarily have a complement. We necessarily own *something*. These facts are taken into account by our grammar.