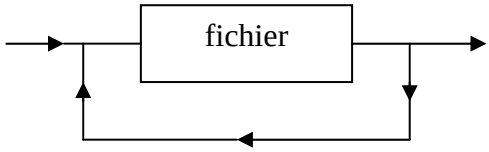
LO01
Langage C

Diagrammes de Conway

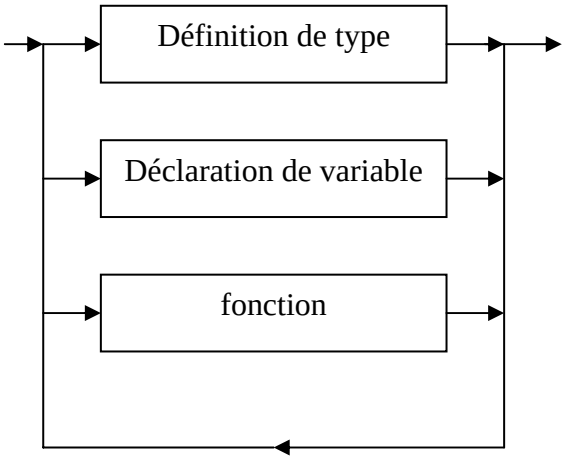
mots réservés :

auto	extern	short
break	float	sizeof
case	for	static
char	goto	struct
continue	if	switch
const	int	typedef
default	long	union
do	register	unsigned
double	return	while
else		

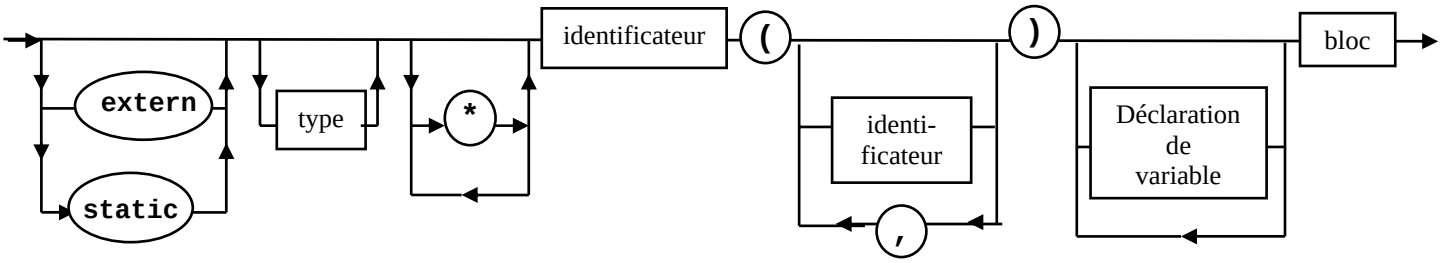
Programme



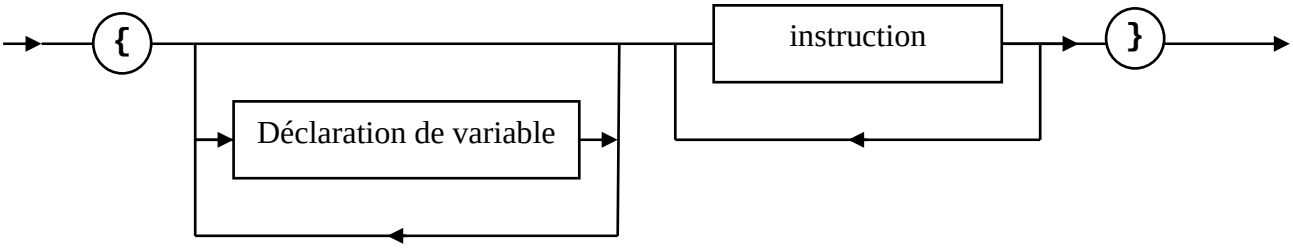
Fichier



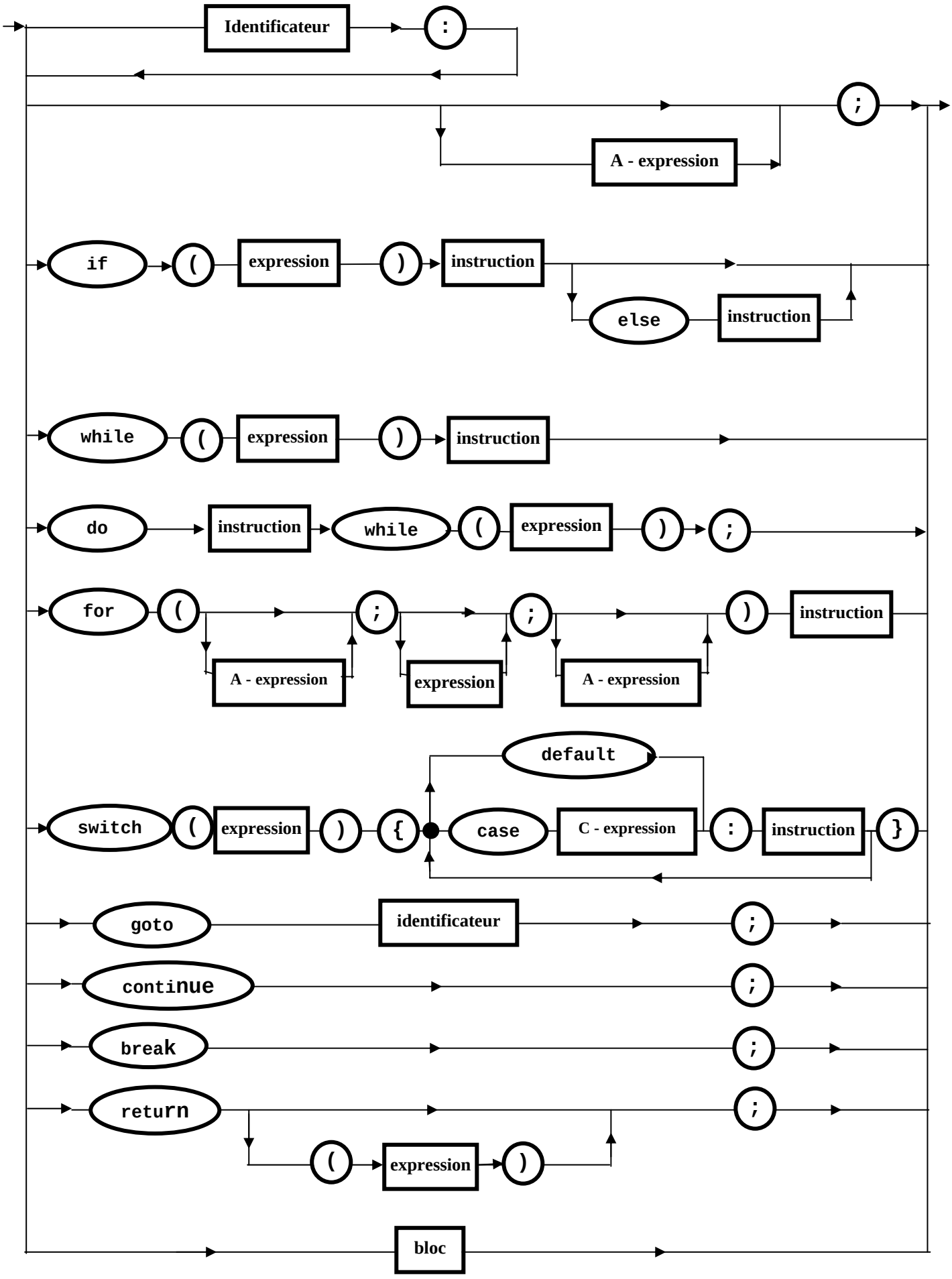
Fonction



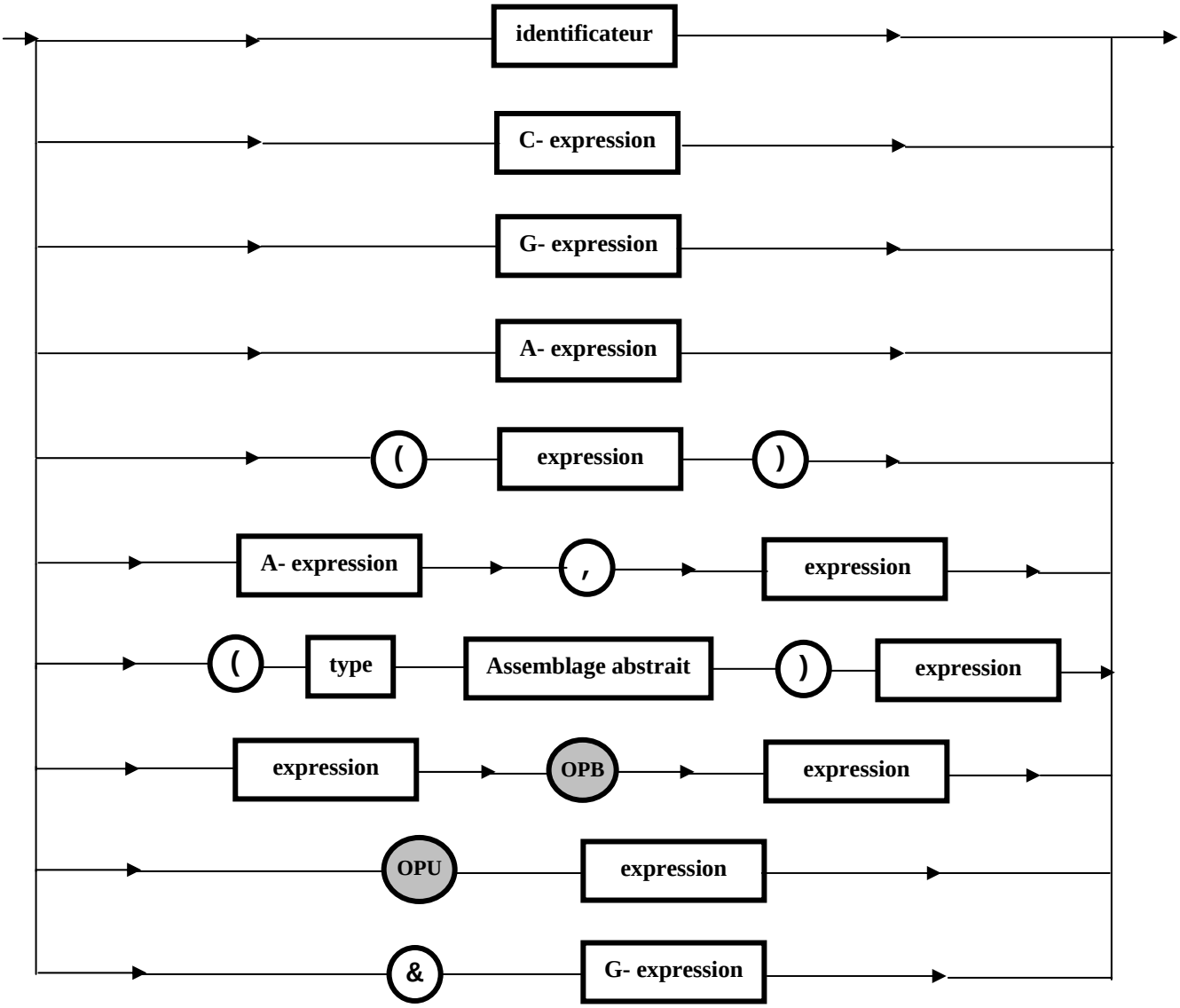
Bloc



Instruction



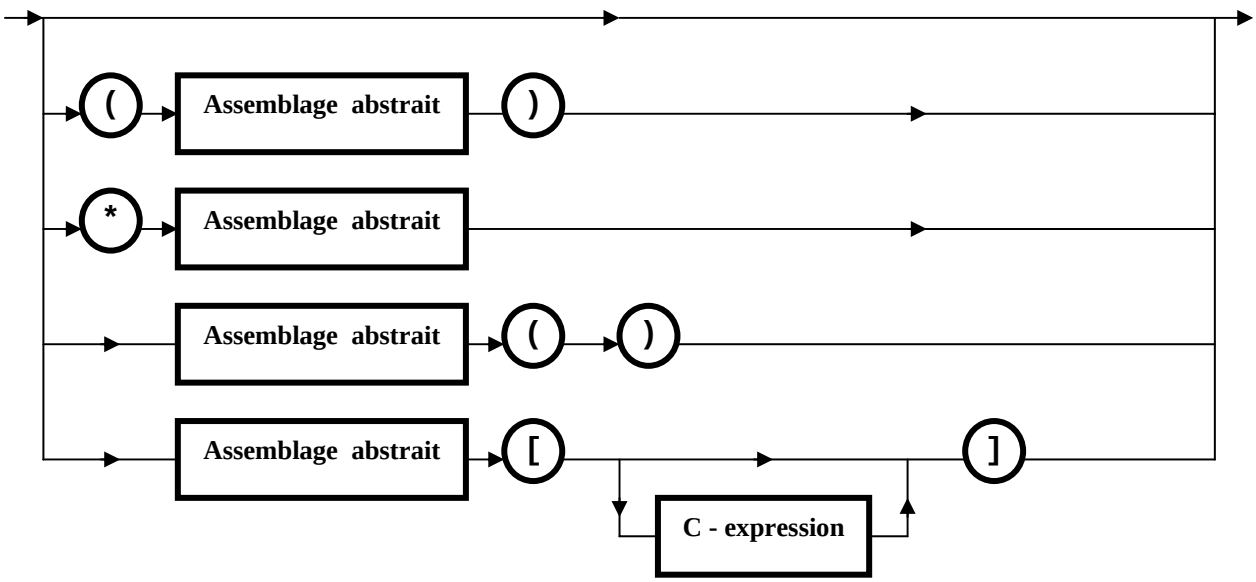
Expression



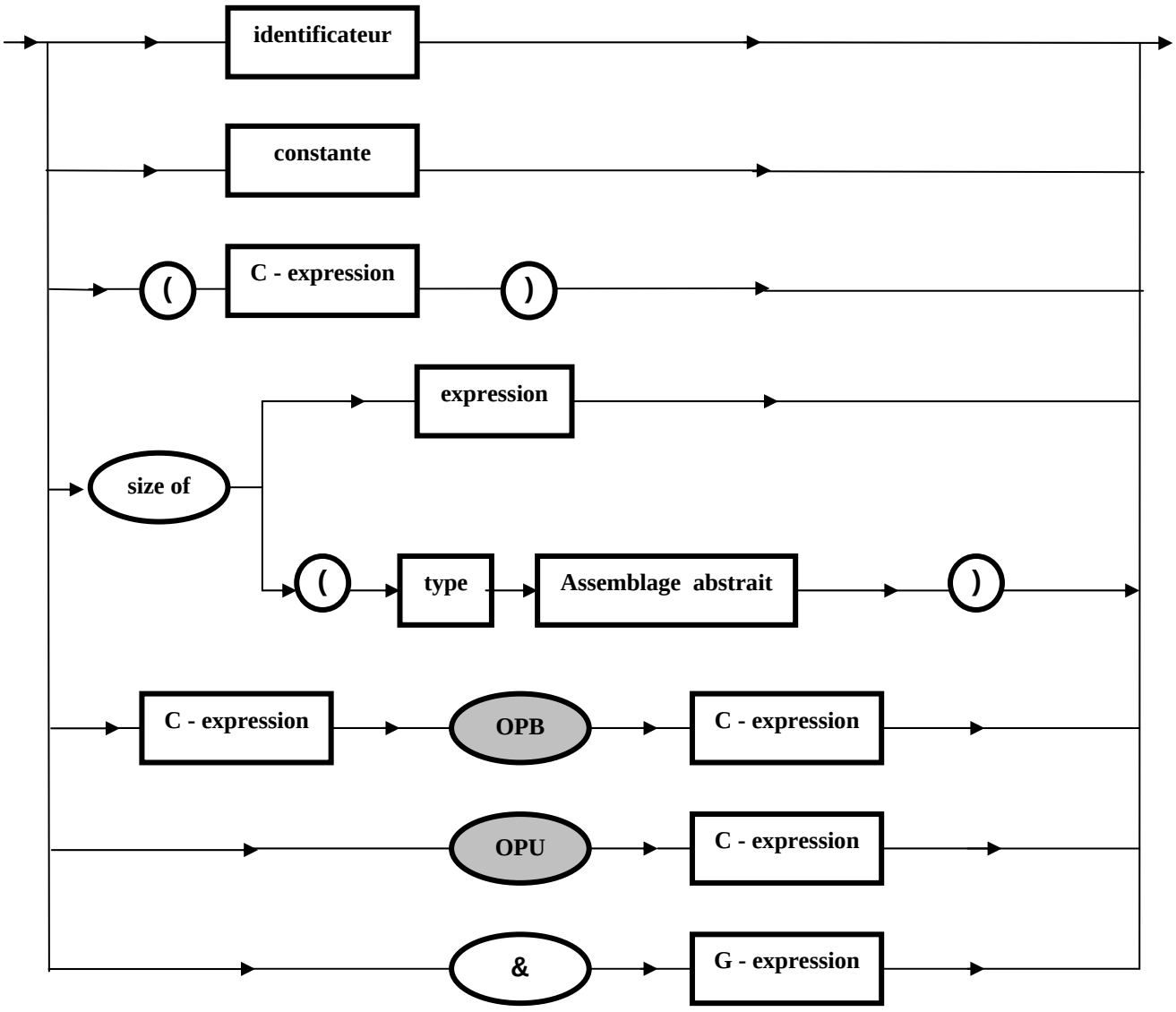
OPB : opérateur binaire
`+ - * / % > < >= <= != == && || >> << & | ^`

OPU : opérateur unaire
`- ! ~`

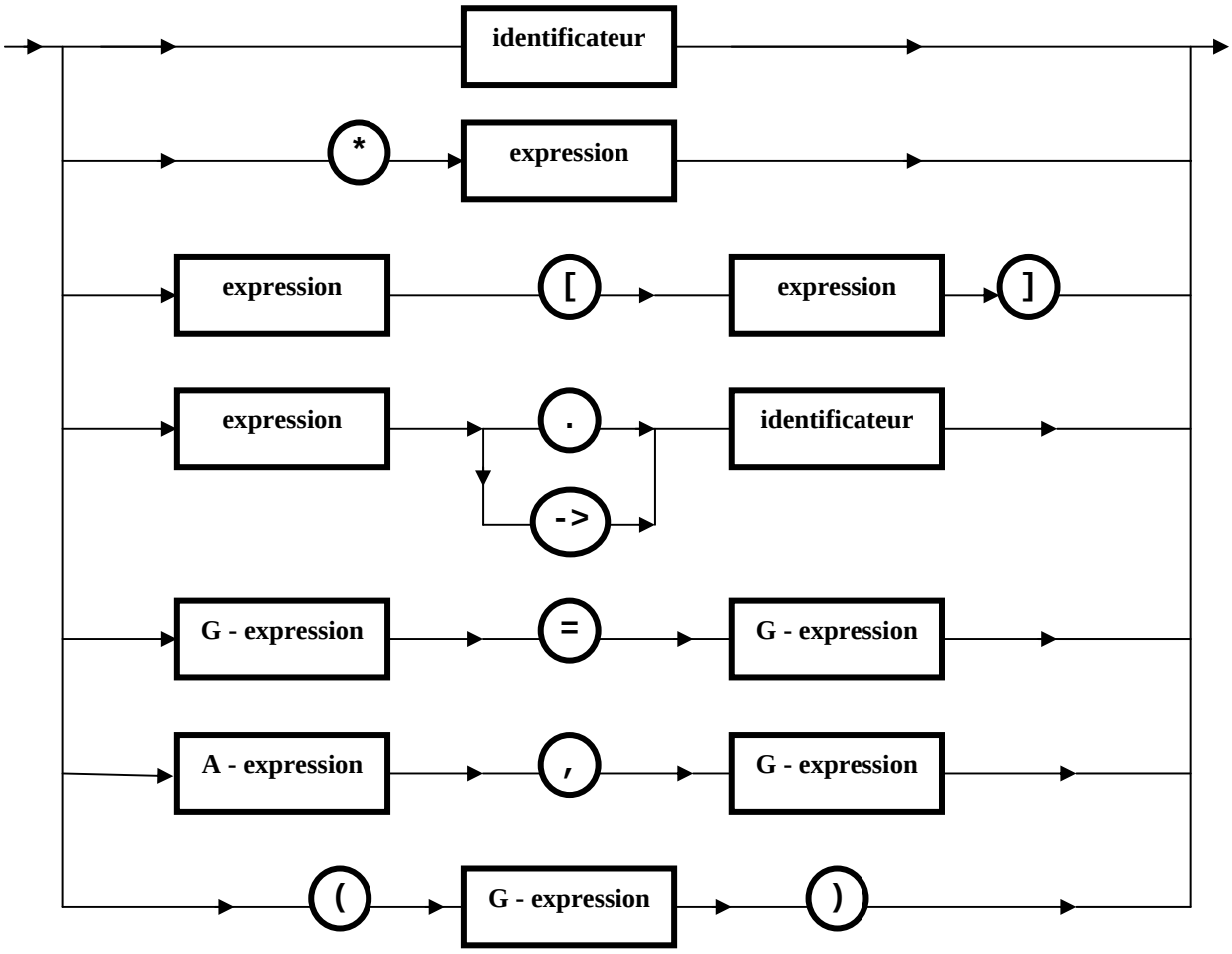
Assemblage - Abstrait



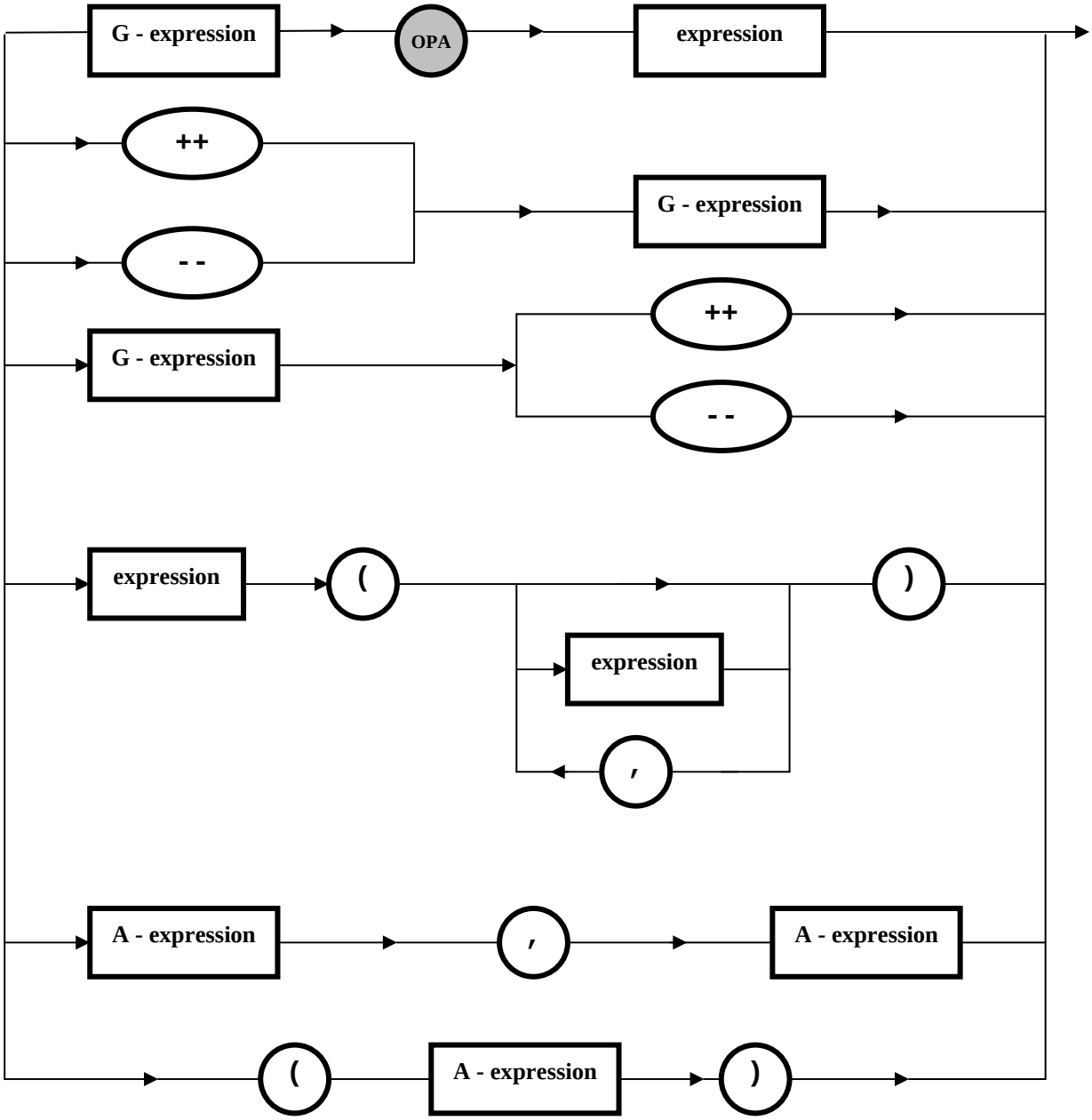
C expression



G expression



A expression



OPA : opérateur d'affectation

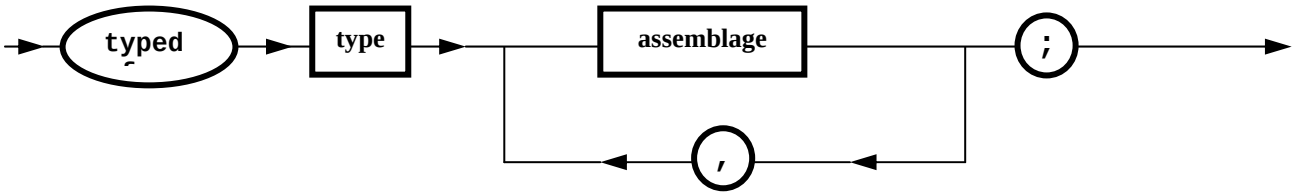
```
=   +=   -=   *=   /=   %=          >>=   <<=   &=   |=   ^=
```

Une *G-expression* est une expression qui définit une adresse mémoire. Ce sont principalement les identificateurs de variable et les expressions construites à partir des opérateurs d'adressage. D'une façon générale, tout ce qui peut se placer à gauche d'un opérateur d'affectation est une *G-expression*

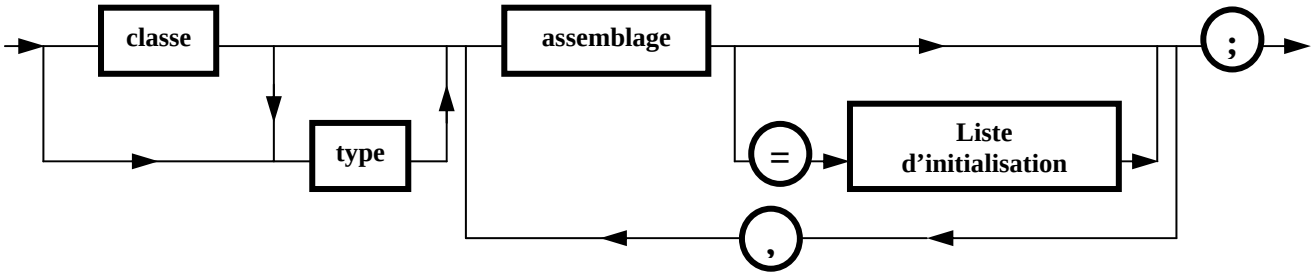
Une *C-expression* est une expression constante dont la valeur est déterminée par le compilateur et l'éditeur de lien. Les *C-expressions* les plus simples sont les constantes, les identificateurs de tableaux et de fonctions. Un opérateur arithmétique dont les opérandes sont des *C-expressions* a pour résultat une *C-expression*.

Une *A-expression* contient une notion d'action : affectation ou appel de fonction.

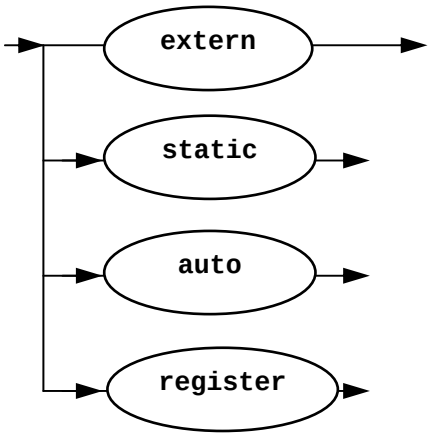
Définition de type



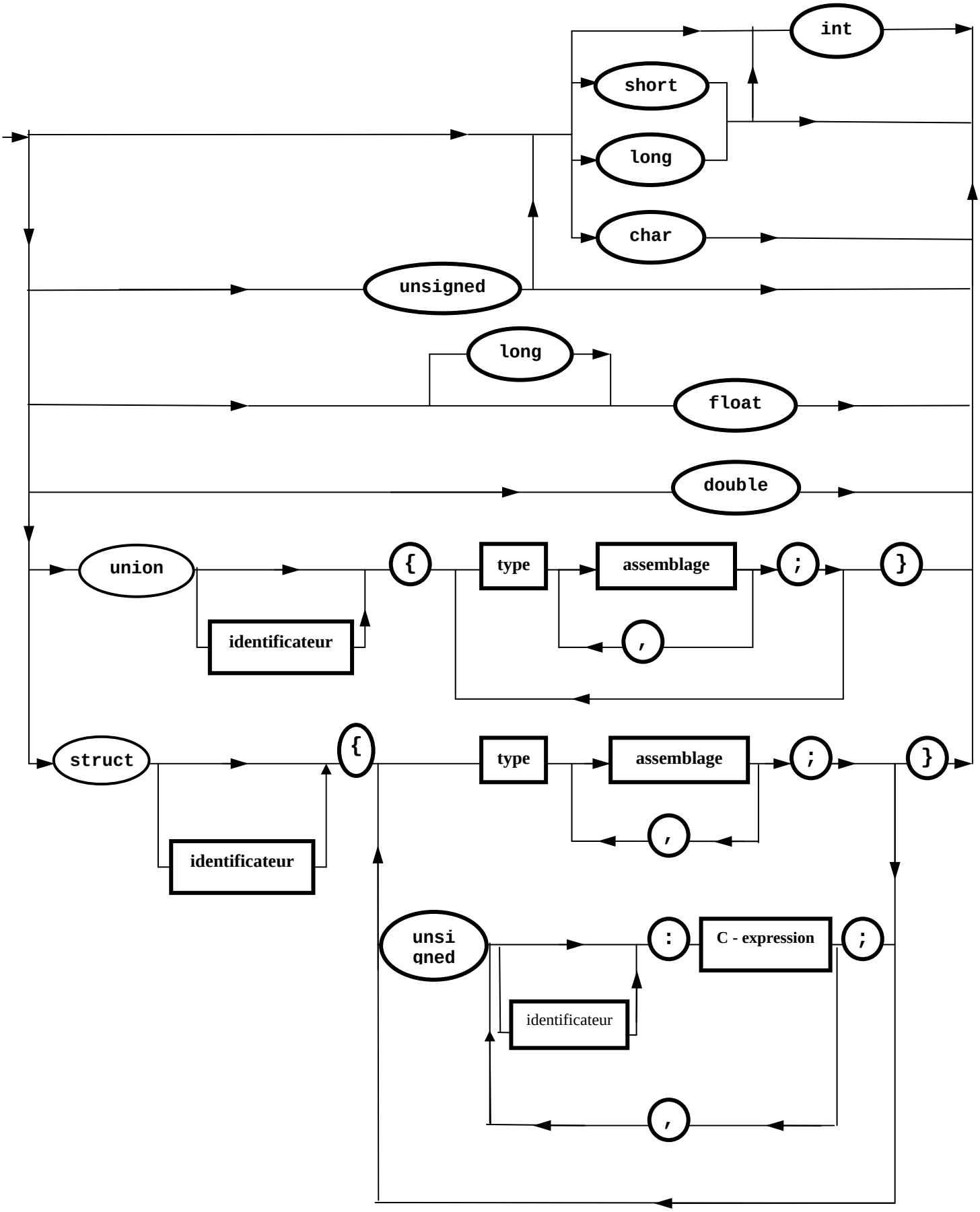
Déclaration d'une variable



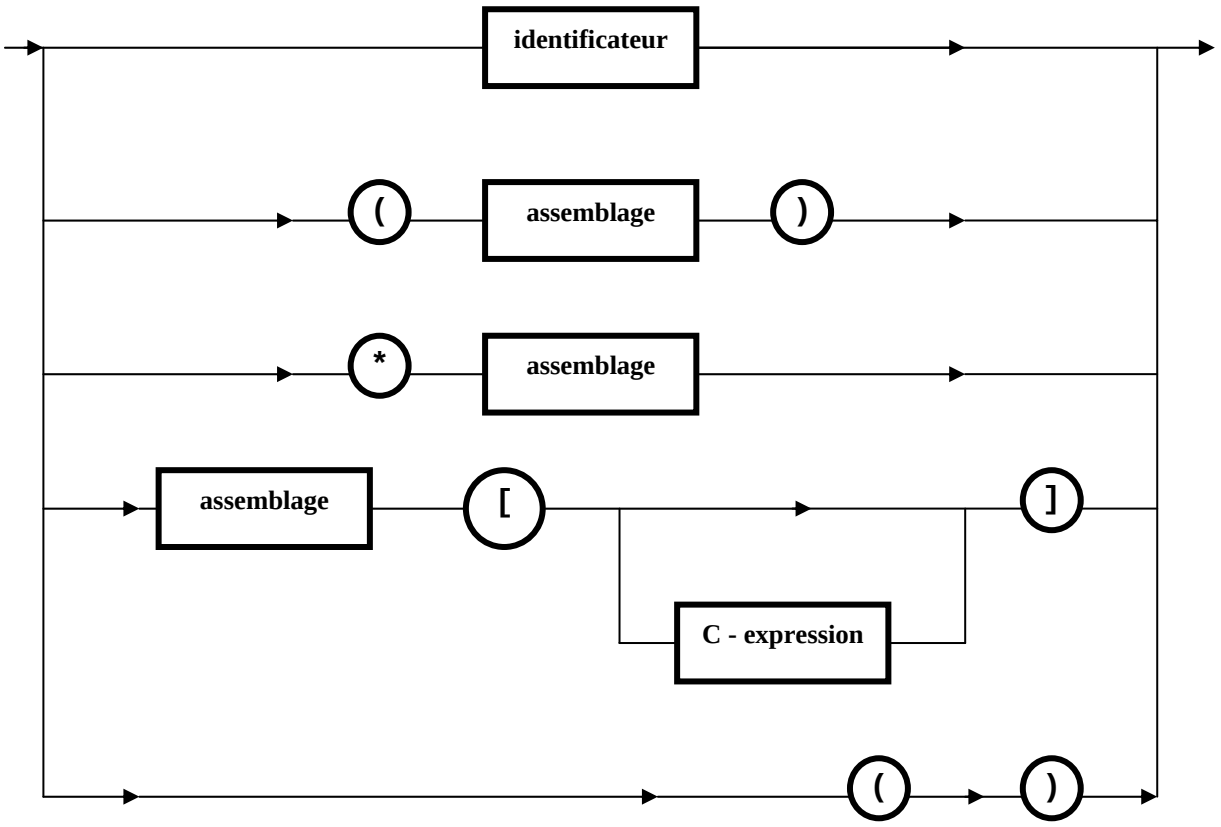
Classe



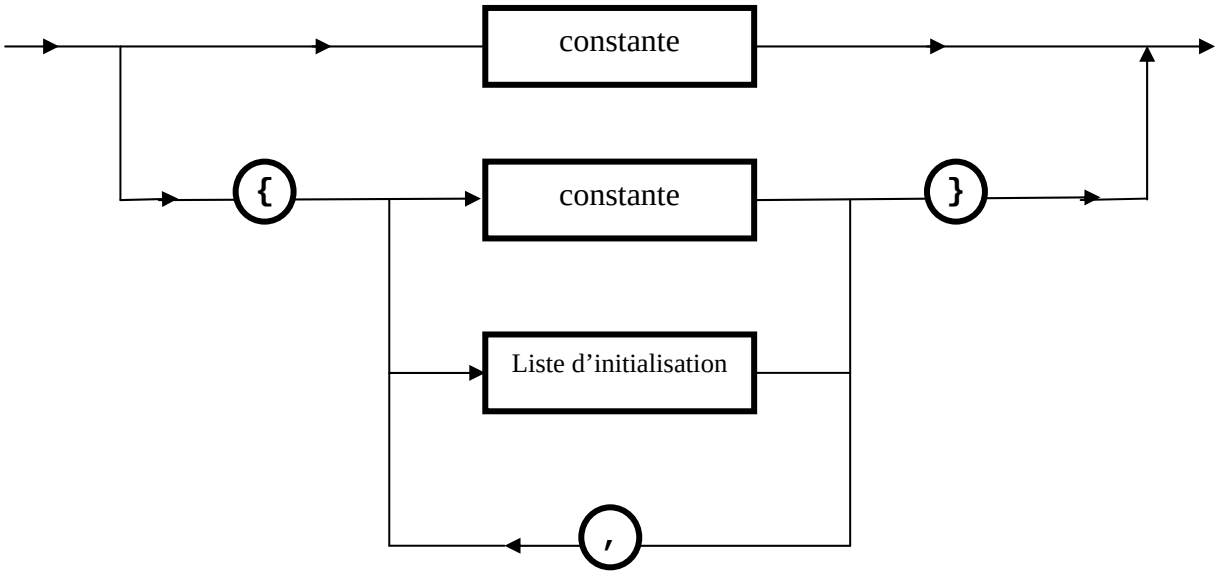
Type


int
short
long
char
unsigned
long
float
double
union
identificateur
{
type
assemblage
,
;
}
struct
identificateur
{
type
assemblage
,
;
}
unsi
gned
:
C - expression
;
identificateur
,

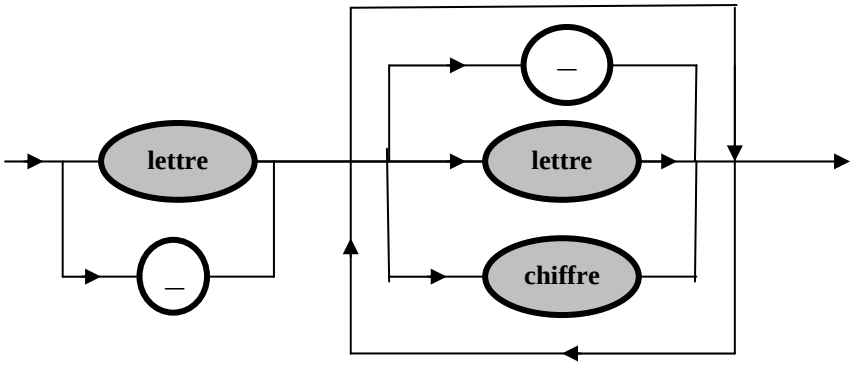
Assemblage



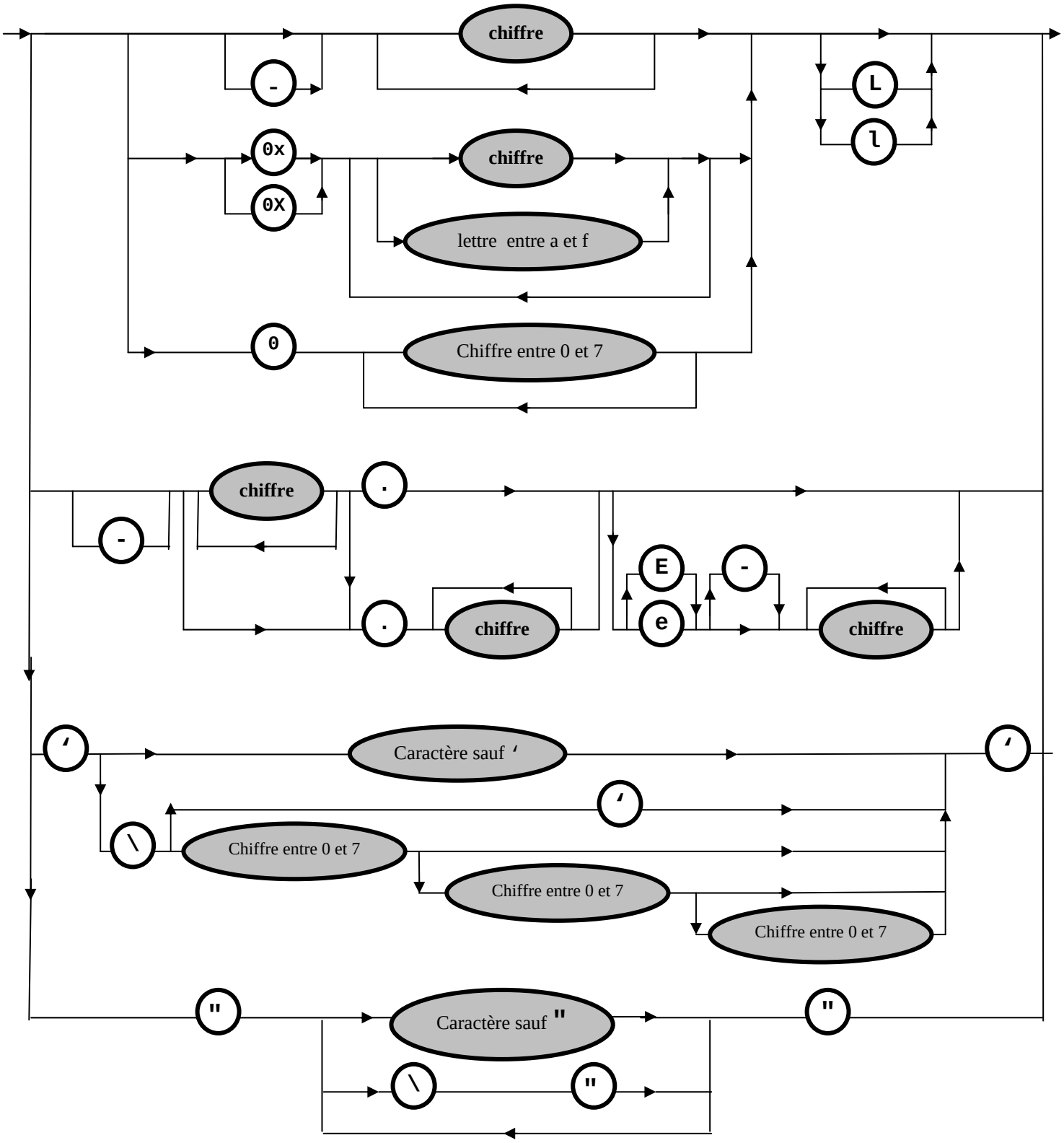
Liste d’initialisation



Identificateur



Constante



Ordre de priorité des opérateurs

forte priorité

Opérateurs	*Evaluation*
`() [] -> .`	gauche -droite
`! ~ ++ -- (<type>) * & sizeof`	droite - gauche
`* / %`	gauche -droite
`+ -`	gauche -droite
`<< >>`	gauche -droite
`< <= > >=`	gauche -droite
`== !=`	gauche -droite
`&`	gauche -droite
`^`	gauche -droite
`|`	gauche -droite
`&&`	gauche -droite
`||`	gauche -droite
`? :`	droite - gauche
`= += -= *= /= %= <<= >>= &= ^= |=`	droite - gauche

faible priorité