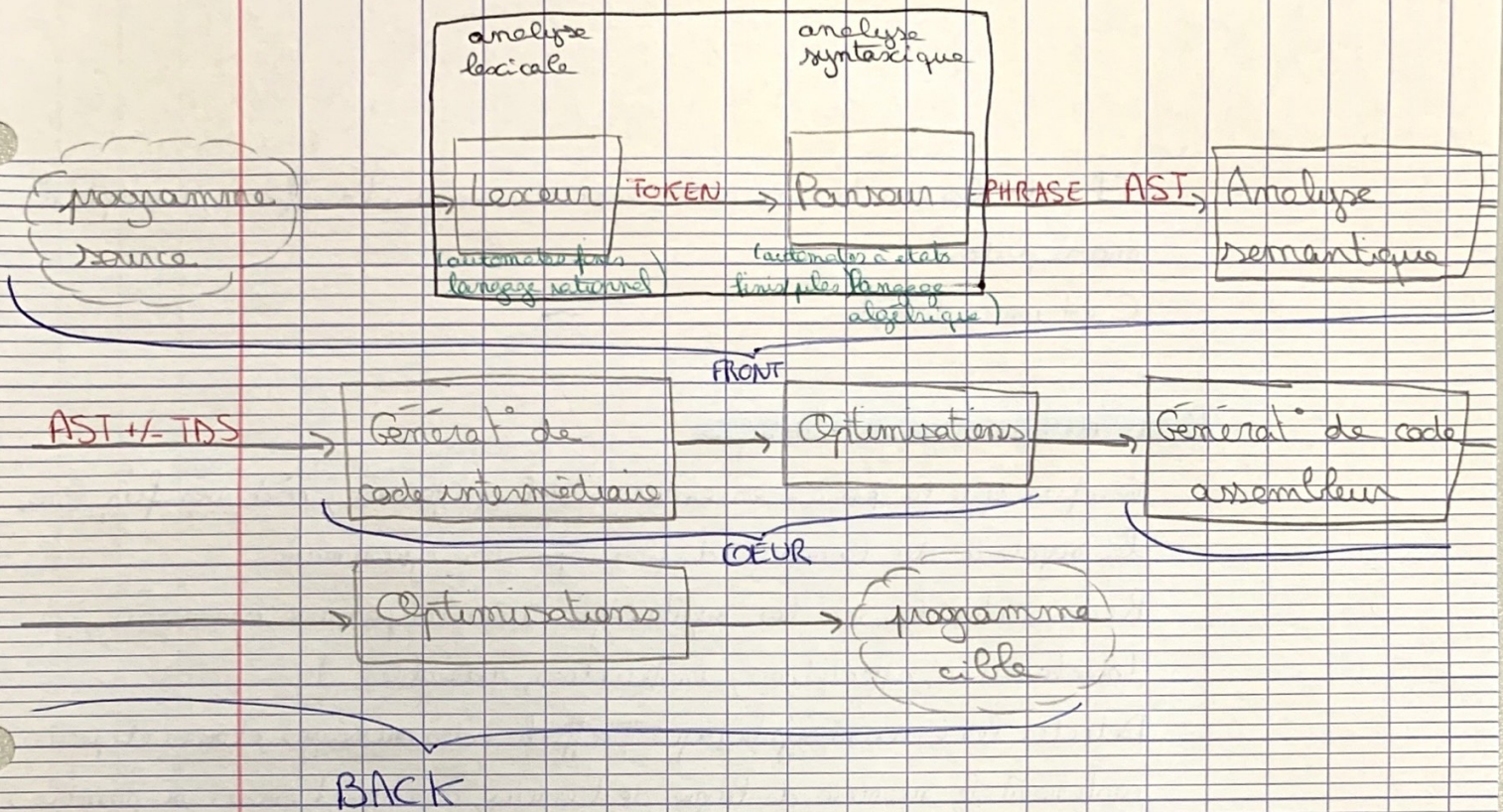

Compilateur : logiciel qui traduit un programme écrit dans un langage de haut niveau en instructions exécutables

Différence entre un compilateur et un interpréteur

* le compilateur lit le code source fourni, le fichier résultat, contenant le code machine (appelé aussi fichier cible), est exécutable une fois pour toutes.
ex : C, C++, fortran...
* L'interpréteur ne crée pas de fichier cible. Il interprète et exécute au fur et à mesure les instructions du programme source
ex : PHP, Python, sh, bash

Analyse lexicale

C'est l'unique module du compilateur qui lit le fichier contenant le code source. Il le lit caractère par caractère.
Il reconnait les unités lexicales (ou tokens) qui sont les mots du langage et les présente à l'analyseur syntaxique.
Il supprime les espaces, les fins de ligne, les tabulations, les commentaires, affiche les erreurs (idf mal formé, caractère inconnu...)

Il code les unités lexicales avec des entiers (+ efficace, + simple à manipuler)
C'est un automate

But de l'analyse sémantique

Vérifier que la phrase en entrée est bien formée, càd est conforme à la syntaxe du langage, définie par une grammaire
Reconnaître les unités syntaxiques décrites par une grammaire (déclaration, affectation, instruction, variables...)
Détecter les erreurs syntaxiques : afficher des messages clairs et précis contenant le numéro de ligne de l'erreur, corriger l'erreur si possible, et poursuivre l'analyse syntaxique pour détecter d'autres erreurs
L'analyse d'un programme correct ne doit pas être ralentie

Analyse descendante

Le code source est lu unité lexicale par unité lexicale
L'arbre syntaxique est construit à partir de l'axiome, en accrochant les parties droites des productions sous la racine (suite d'expansion)
Le texte est lu de gauche à droite
LL(k) : left to right scanning, left most derivation
Une grammaire est LL(k) si on utilise les k unités lexicales suivantes pour décider de la product° à appliquer lors d'une expansion

Algorithme : Analyse syntaxique descendante par fct récursives

A chaque non-terminal on associe une fct qui vérifie que le texte correspond à la syntaxe décrite par les ≠ alternatives
Pour choisir entre ces ≠ alternatives, les fct disposent d'une unité lexicale d'avance

Analyse ascendante

L'analyse syntaxique ascendante ne fonctionne que par table d'analyse déterministe

On part du noyau à analyser

On remplace itérativement des fragments du mot courant qui correspondent à des parties droites de règles de production par le membre gauche de cette règles.

L'analyse est OK si le mot courant final est l'axiome de G

LR(k) : Left to right scanning, Right most derivation

Item

Un item d'une grammaire G est une product° de G avec un marqueur (noté .) repérant une position de sa partie droite.

Fermeture

Soit I un ensemble d'items de G

Fermeture (I) est un ensemble d'items construits à partir de I tq :

- placer chaque item de I dans fermeture (I)
- si $A \to \alpha . B\beta \in$ Fermeture (I) et si $B \to \gamma$ alors ajouter $B \to .\gamma$ à Fermeture (I) (sauf si déjà dedans)
- itérer jusqu'à ne plus trouver d'item à ajouter à Fermeture(I)

Etat de l'automate LR(0)

Les états de l'automate LR(0) sont les ensembles I d'items obtenus par fermeture

L'état initial I_0 est obtenu par fermeture de $S' \to .S$

Transition de l'automate LR(0)

Soit I un ensemble d'items, X un symbole de G ($X \in (N \cup T)$)

On définit transition (I, X) comme étant la fermeture de l'ensemble des items $A \to \alpha X . \beta$ tq $A \to \alpha . X\beta \in I$

Toute grammaire SLR(1) est non ambigüe
Toute grammaire LR(0) sans conflit est SLR(1)

k dans LR(k) représente le nb d'unité lexicale dont on a besoin pour prendre la décision de réduction

Analyse sémantique

vérifie que le programme satisfait un ensemble de règles sémantiques
vérifie que le texte source ou AST est conforme à un ensemble de règles de construction des programmes, définie par le manuel de référence du langage

2 catégories d'erreurs "sémantiques"
statiques : "vérifiables" à la compil

dynamiques "vérifiables" à l'exécution
ex : division par 0, allocat° mémoire, dépassemt tableau

LR(0) → SLR(1) ⟶ LR(1) ⟶ LALR(1)

on prend en compte les suivants	on prend en compte le contexte et les suivants	optimisat° fusion d'états

Technique de vérification

le nb de réduction c'est le nb d'État Terminal par contexte
le nb de déplacements c'est le nb de terminaux sur les flèches
le nb de transitions c'est le nb de non terminaux sur les flèches

Pour LALR(1) on regarde si on peut fusionner les états équivalents