5.3.1 Interprétation

Interprétation avec ETF

$E \to E + T$	*empiler(dépiler(P) + dépiler(P))*
$E \to T$	
$T \to T \times F$	*empiler(dépiler(P) × dépiler(P))*
$T \to F$	
$F \to (E)$	
$F \to a$	*empiler(P,valeur(a))*

Les actions associées aux règles permettent d'évaluer (calculer) l'expression en même temps qu'on la reconnaît (en analyse descendante).

Ce qu'on vient de réaliser, ce n'est pas un *compilateur*, mais un *interpréteur*. Mais sur la même idée, on peut construire un vrai compilateur :

Compilation avec ETF

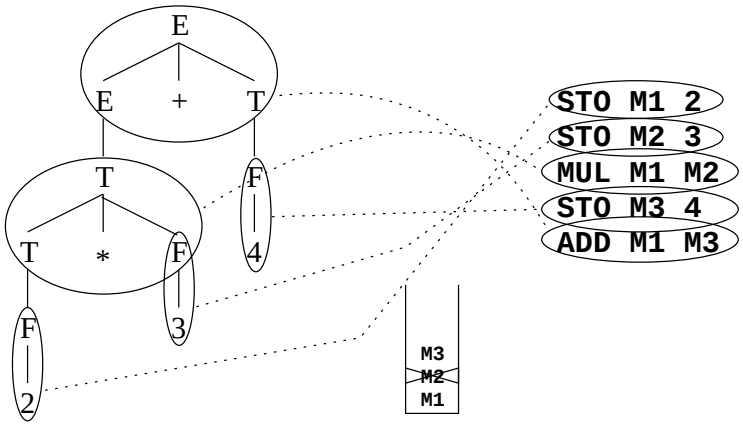
$E \to E + T$	*Engendrer l'opération :* – additionner le contenu des mémoires dont les adresses sont en sommet de pile[5] ; – mettre le résultat dans la première des deux ; – supprimer un élément de la pile[5]
$E \to T$	
$T \to T \times F$	*Engendrer l'opération :* – multiplier le contenu des mémoires dont les adresses sont en sommet de pile[5] ; – mettre le résultat dans la première des deux ; – supprimer un élément de la pile[5]
$T \to F$	
$F \to (E)$	
$F \to a$	*Engendrer l'opération :* – mettre a dans une case libre[5] ; – mettre l'adresse de cette case dans la pile[5]

Résumé Avec une grammaire et des « actions », on peut faire :

Analyse syntaxique Action : imprimer n° de règle
Traduction dans $X^* \times Y^*$ Action : appliquer la règle dans $\mathcal{G}_2$.
Interprétation Action : calculer sur des valeurs
Compilation Action : générer du code

[5] On suppose que le compilateur gère des cases mémoire libres, ainsi qu'une pile, qui ne sera pas nécessaire *at run-time*.

FIG. 5.2 – Exemple de compilateur ETF



5.3.2 Génération de code avec une grammaire d'attributs

L'un des attributs contient le code généré pas à pas (synthétisé), et on ajoute d'autres attributs pour les informations nécessaires *at compile time*.

Exemple 1 : ETF

On introduit deux attributs : `nb` pour le nombre de registres nécessaires, et `code` pour le code lui-même. Ils sont tous deux synthétisés.[6]

$$
\begin{array}{lcll}
F \to a_i & : & F.\texttt{code} & := \texttt{LOAD } i \texttt{ R1} \\
 & & F.\texttt{nb} & := 1 \\
F \to (\,E\,) & : & F.\texttt{code} & := E.\texttt{code} \\
 & & F.\texttt{nb} & := E.\texttt{nb} \\
T \to F & : & T.\texttt{code} & := F.\texttt{code} \\
 & & T.\texttt{nb} & := F.\texttt{nb} \\
E \to T & : & E.\texttt{code} & := T.\texttt{code} \\
 & & E.\texttt{nb} & := T.\texttt{nb} \\
E \to E + T & : & E^0.\texttt{code} & := \left[\begin{array}{l} (E^1.\text{nb} \geq T.\texttt{nb}) \;\; ? \\ E^1.\texttt{code}\,;\, \mathcal{T}(T.\texttt{code})\,;\, \texttt{ADD R1 R2} \;\; : \\ \text{T.code}\,;\, \mathcal{T}(E^1.\texttt{code})\,;\, \texttt{ADD R1 R2} \end{array}\right] \\
 & & E^0.\texttt{nb} & := \Big[(E^1.\text{nb} == T.\texttt{nb})\;?\; E^1.\texttt{nb} + 1 :\; max(E^1.\text{nb}, T.\texttt{nb})\Big] \\
T \to T \times F & : & T^0.\texttt{code} & := \left[\begin{array}{l} (T^1.\text{nb} \geq F.\texttt{nb}) \;\; ? \\ T^1.\texttt{code}\,;\, \mathcal{T}(F.\texttt{code})\,;\, \texttt{MUL R1 R2} \;\; : \\ \text{F.code}\,;\, \mathcal{T}(T^1.\texttt{code})\,;\, \texttt{MUL R1 R2} \end{array}\right] \\
 & & T^0.\texttt{nb} & := \Big[(T^1.\text{nb} == F.\texttt{nb})\;?\; T^1.\texttt{nb} + 1 :\; max(T^1.\text{nb}, F.\texttt{nb})\Big]
\end{array}
$$

Code « translaté » : Décalage des numéros de registre.
Par exemple, $\mathcal{T}(\texttt{LDA A R1}) = \texttt{LDA A R2}$.

[6] `LOAD` = `LDA` = `STO`.