ENSEEIHT

Sémantique et TDL

Projet

Compilateur du langage $\mu C^{\#}$

auteurs :
ACHARD - VINCENT
BARRAS - THOMAS
BOUQSIMI - AMINE
RIGONDAUD - CLÉMENT

Table des matières

Chapitre 1

Structure

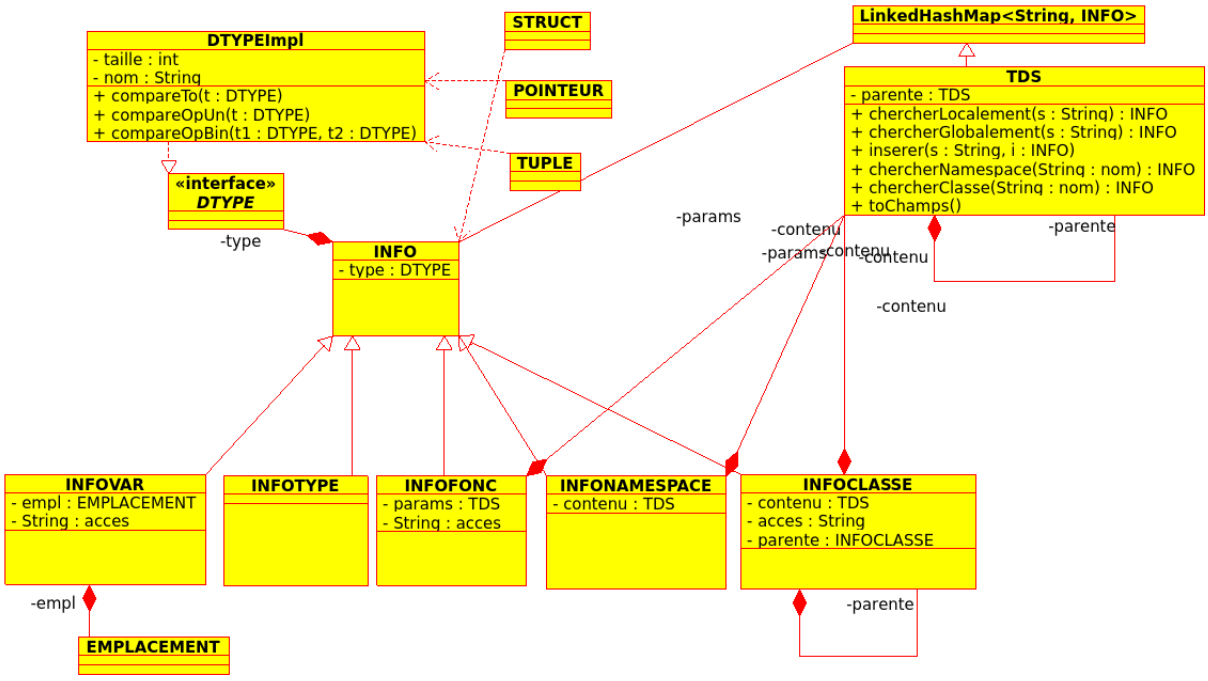

FIGURE 1.1 – UML global

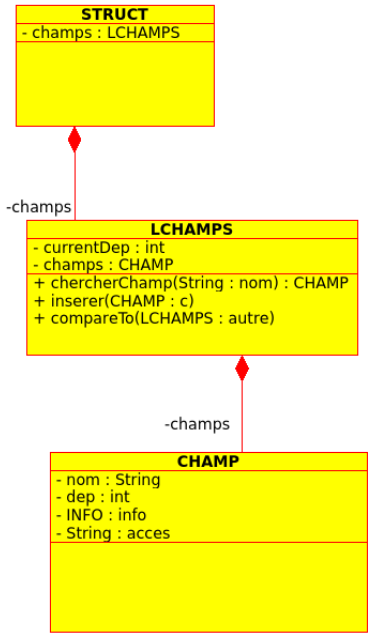

FIGURE 1.2 – Précision STRUCT

Les principaux choix de conception sont liés à la gestion de la TDS. Afin de distinguer les éléments présents dans celle-ci, plutôt que de donner un attribut à une Info pour distinguer sa vraie nature, nous avons choisi d'utiliser le typage. C'est pour cette raison que nous avons créé les classes Infofonc, Infonamespace, Infoclasse, et Infovar qui héritent toutes de la classe INFO.

INFOFONC : Représente une fonction. Le type de cette fonction correspond à son type de retour. Une Infofonc possède une TDS contenant les paramètres de la fonction.

INFONAMESPACE : Représente un Namespace. Une Infonamespace possède une TDS contenant l'ensemble des définitions effectuées au sein de ce dernier.

INFOCLASSE : Représente une classe objet. Une Infoclasse possède un contenu, ce qui correspond aux attributs et aux méthodes de la classe. La distinction entre un attribut et une méthode s'effectue grâce au typage. Acces correspond à l'accessibilité de la classe (private, public, protected). Et enfin, parente est la « super-classe » de la classe en question.

INFOVAR : Représente une variable quelconque. Elle possède un attribut accès qui représente son accessibilité (public, private), utilisée uniquement dans le cadre de lobjet. Lattribut emplacement représente la position dans la pile, par rapport à un registre donné. Lattribut mode correspond au mode de passage de la variable dans les paramètres dune fonction. Si la variable en question nest pas un paramètre, son mode par défaut sera « ref » : on peut la lire et laffecter.

En ce qui concerne le système de type, nous avons simplement rajouté les méthodes compareOpUn et compareOpBin permettant de tester respectivement la compatibilité entre un d'un opérateur binaire avec son argument et celle d'un opérateur binaire avec ses deux arguments.
La classe CHAMPCLASSE permet de représenter les attributs et les méthodes d'une classe de la même manière que l'on utilise la classe CHAMP pour représenter un élément d'un Struct. La différence est l'existence d'un attribut de type Info permettant de faire la différence entre une méthode et un attribut, différence impossible à effectuer uniquement à partir des informations contenues par CHAMP. Chaque CHAMPCLASSE possède également une accessibilité.

Chapitre 2

Choix de conception

— Assembleur Inline : Nous avions au départ modifié la classe Infovar de manière à gérer la position dans la pile sans passer par la classe Emplacement, et donc créé nos propres règles pour la gestion de l'assembleur Inline, à cause d'erreurs liées à des conflits avec les règles déjà existantes. Nous sommes ensuite revenu sur notre décision, et avons utilisé la classe Emplacement, encore une fois à cause de conflits avec le traitement initialement prévu pour l'assembleur Inline.

— Nous avons choisi de représenter un objet sous la forme d'un pointeur vers un Struct.

— L'utilisateur doit indiquer toutes les variables globales en premier, c'est à dire avant les fonctions/Namespaces, pour une raison de complexité de traitement.

— Le déplacement courant au sein dun Bloc est conservé non pas grâce à un attribut, mais directement par la machine. Cela simplifie grandement lécriture des règles. La même classe Java soccupe de la modification de la valeur de ce déplacement.

— Au début, nous avions choisi dutiliser une classe Infoparam pour représenter le fait quune variable soit le paramètre dune fonction. Nous insérions en effet les paramètres comme des variables normales, puis au moment de leur affectation, nous testions leur appartenance aux paramètres pour savoir si le déplacement par rapport à LB était négatif. Maintenant, nous donnons directement aux paramètres une valeur de déplacement négative, conformément à la solution vu en TD.

— L'attribut mode est utilisé à la fois pour gérer l'affectation des paramètres des fonctions, mais aussi pour éviter que des expressions autres que identifiants puissent être affectées. Pour cela, on recherche le seul cas aboutissant à une expression comportant seulement un identifiant, à savoir celui où aucun opérateur n'apparaît dans l'expression. Il existe cependant un cas où une expression autre quun ident peut se trouver à droite dun opérateur daffectation, à savoir lorsquil sagit dune condition. Doù la présence dun attribut permettant de vérifier si lon se trouve dans ce cas.

— Au départ, nous avions choisi de créer une classe Champclasse héritant de Champ, et possédant les attributs nom, dep, type, acces, et info. Le problème avec cette implantation, c'est que l'attribut info contient déjà les informations sur le type et l'accès. Il était donc plus logique de remplacer l'attribut type de la classe champ pour le remplacer par une info. Cela ne pose pas de problème car toutes les variables ont une accessibilité, set par défaut à « default », avoir comme attribut une info pour un champ de struct n'est donc pas un problème.

Chapitre 3

Limitations

Supposons que l'on ait deux fonctions du même nom dans deux Namespaces différents, et que l'on se trouve dans un troisième Namespace dans lequel on utilise les deux précédents. Si l'on souhaite appeler ladite fonction, qui par ailleurs n'existe pas dans le dernier Namespace, il est impossible de choisir laquelle appeler, il s'agira de la première trouvée par la recherche (vraisemblablement celle du premier Namespace).