OPTIMISATION GLOBALE

Des optimisations qui dépendent des interactions entre plusieurs blocs de base

Une optimisation est possible si quelque chose est arrivée avant l'entrée de ce bloc ou va arriver après la fin

Nécessite une analyse des chemins possibles dans le graphe

[Supposition pas forcément vraie mais sans danger que chaque chemin dans le graphe correspond à une suite d'exécutions de blocs]

Cette analyse serait triviale si les programmes n'avaient pas de boucles ...

LES OPTIMISATIONS

- ne pas stocker les valeurs dans des registres à la fin d'un bloc de base que si les variables sont vivantes

- éviter les calculs redondants de sous-expressions communes

- supprimer (propager) les copies

- invariantes de boucles composées de plusieurs blocs de base

- calculer une expression dès qu'il est certain qu'elle va être utilisée (en plusieurs blocs de base)

LES ENSEMBLES A CALCULER

- Les variables vivantes
- Les expressions disponibles
- Les définitions: chaînes u-d: la liste de définitions (affectations) pouvant donner la valeur utilisée
- Les utilisations: chaînes d-u:
- Les variables forcément utilisées

UTILISER CES ENSEMBLES

- Pour sous-expressions: les expressions disponibles

- Pour invariantes: les chaînes u-d

- Pour code mort: les chaînes d-u

- Pour propagation de copies: u-d et d-u

- Pour supprimer les copies (à la fin d'un bloc): les variables vivantes

LES EQUATIONS

- Un ensemble au début et un autre à la fin de chaque bloc de base
- une équation pour chaque bloc qui relie les deux ensembles selon les instructions du bloc
- une équation pour chaque bloc qui relie un de ses ensembles à ceux de ses prédécesseurs ou successeurs
- deux directions de dépendance
- deux opérateurs de combinaison
- quelle solution?
  (dépend de l'opérateur)

UN EXEMPLE: LES VARIABLES VIVANTES

D_i: l'ensemble de variables vivantes au début de bloc numéro i
F_i: idem à la fin
U_i: les variables utilisées en bloc i (avant d'être définies)
C_i: les variables changées en bloc i (avant d'être utilisées)

$$F_i = \cup_{j \textit{ successeur de } i} D_j$$

$$D_i = F_i \cup U_i - C_i$$

Nous voulons la solution minimale

RESOUDRE LES EQUATIONS

- Solution itérative

- Commencer avec tout ensemble = ($\emptyset$ ou tout) selon la solution désirée (donc $\emptyset$ dans l'exemple)

- Plus vite si la direction est celle des dépendances (de bas en haut dans l'exemple)

- Assez vite avec représentation des ensembles par les vecteurs de bits

LES COMPLICATIONS

- Procédures
- Tableaux
- Pointeurs

On doit prendre un point de vue conservateur: ne faire que les optimisations qui ne pourraient pas changer l'effet d'un programme (tant que cet effet est défini?!)

- une affectation à un élément d'un tableau peut changer n'importe quel élément (sauf indice connu constant)

- une affectation à `*p` peut changer n'importe quoi (sauf restrictions de type dans le langage...)

- dans une procédure, deux paramètres (ou un paramètre et une variable externe) peuvent être des alias

- les effets d'un appel d'une procédure peuvent être déterminés (toujours de façon conservatrice) par l'analyse de l'intérieur de la procédure et une analyse des paramètres.