DM Module OPC, 2018-2019

21 janvier 2019

L'objectif de ce devoir maison est de vous faire étudier et utiliser une analyse statique de programme, basée sur le modèle polyhédrique, qui permet de calculer ou tout du moins d'estimer le nombre de cache miss dans un nid de boucles. La première partie du homework peut-être vue comme un ensemble de questions de cours qui vous serviront d'échauffement pour attaquer la seconde partie, dans laquelle vous serez amené à étudier en profondeur une contribution issue d'article de recherche. Notez qu'il n'est pas nécessaire d'aller jusqu'au bout du pour avoir une bonne note.

Le devoir devra être rendu pour le 1er février au plus tard, est devra être envoyé à l'adresse suivante sderrien@irisa.fr. Assurez vous d'avoir obtenu un accusé de réception de ma part. N'hésitez pas non plus à m'envoyer un mail si vous avez des question ou besoin de précisions sur le sujet (par contre n'attendez pas le dernier moment !).

1 Préambule

1.1 Une boite à outil pour la compilation polyhédrique

L'outil iscc est une interface de commande offrant toutes les fonctionnalités nécessaire à l'analyse et à la transformation de nids de boucle dans le formalisme dit "polyédrique". L'outil peut facilement est compilé/installé sur une machine linux, et est également disponible via une interface web [1]. La documentation de l'outil correspond au premier chapitre du manuel de la librairie barvinok [2] et vous êtes très fortement invités à la consulter avant de répondre aux questions ci-après.

Question 1 :

A quelles opérations polyédriques correspondent les commande ci-dessous ? Expliquez.

```
A := [N] -> { [i,j] : 0<i<N and 15<=j<34 and not(exists k:2k=j+i and 0<k<3) };
B := [N] -> { [i,j] : 0<=i<j and 0<=j<N };
C := A*B;
print C;

D := card(C);
print D;

E := { [i,j] -> [i',j'] : i'=i/16 and j'=i%16};
F := E(B);
print F;
```

Question 2 :

A quoi correspond le résultat de la séquence de commandes ci-dessous (on s'intéresse à sa signification et non pas au résultat) ?

```
Domain  := [n] -> {
    S[k] : k <= -2 + 2n and k >= 0;
T[i, j] : i >= 0 and i <= -1 + n and j <= -1 + n and j >= 0;
};
```

1. http://compsys-tools.ens-lyon.fr/iscc/
2. http://barvinok.gforge.inria.fr/barvinok.pdf, http://barvinok.gforge.inria.fr/tutorial.pdf

```
Schedule := [n] -> {
T[i, j] -> [1, j, i+j];
S[k] -> [0, k, 0];
};
codegen (Schedule * Domain);
```

Question 3 :

Soit le domaine $\mathcal{D}$ défini ci-dessous.

```
D := [N] -> { [i,j,k] : 0<=i<j and 0<=j<N and i<k<j};
```

On souhaite déterminer la projection de ce domaine sur les dimension i et k, puis calculer le nombre de points dans cette projection. Donnez la séquence de commande iscc nécessaires (vous justifierez chaque opération).

Question 4 :

Soit le nid de boucles ci-dessous, N est un paramètre dont la valeur n'est pas connue à la compilation (on sait simplement que $1 < N < 512$).

```
float T[1024];
float Y[1024];

for (i=0 ;i<N ; i++) {
S0: Y[i]=0 ;
    for (j=1 ;j<i ;j++) {
S1:    Y[i] = Y[j] + T[i-j];
    }
}
```

Donnez, en suivant la syntaxe d'iscc :
- Les définitions suivantes des domaines d'itérations $\mathcal{D}_{S0}$ et $\mathcal{D}_{S1}$, des instructions `S0` et `S1`.
- Les vecteurs d'ordonnancement des instructions du programme original
- Les relations de dépendance *RAW, WAR* et *WAW* liant les instructions S0(i) et S1(i,j).

Question 5 :

On se propose d'appliquer une transformation de boucle basée sur les fonctions d'ordonnancements ci-dessous :
- $\Theta_{S0}(i) = (i, 0, 0)$
- $\Theta_{S1}(i, j) = (j, 1, i + j)$

Donner la séquence de commande iscc permettant d'obtenir une version du code transformé, ainsi que le code obtenu.

Question 6 :

Cette transformation préserve-t-elle la sémantique du programme original ? Vérifiez sa légalité à l'aide des techniques étudiées en cours (vous utiliserez pour cela les fonctions d'iscc) en détaillant votre raisonnement (par on peut compter le nombre d'itération pour lesquelles on viole une dépendance, et s'assurer que ce nombre vaut 0).

Question 7 :

Donnez la séquence de commandes iscc permettant de calculer le nombre total d'exécutions de ces instructions au cours de l'exécution de la boucle (indice : ce résultat sera une fonction du paramètre `N`). Vous veillerez à bien expliquer chacune des étapes.

Question 8 :

Formulez, en suivant la syntaxe d'iscc, la relation liant l'ensemble des cases du tableau `T[p]` accédées (en lecture ou écriture) lors d'une exécution de la boucle. Déduisez-en la formule donnant le nombre d'emplacements distincts de ce tableau accédés au cours de l'exécution de la boucle (vous donnerez et expliquerez la séquence d'opérations permettant d'arriver à ce résultat).

Question 9 :

Calculez le nombre nombre d'emplacements distincts du tableau `T[p]` accédés (en lecture ou écriture) lors d'une exécution d'une itération de la boucle externe i. Notez que cette fois-ci le résultat sera une expression fonction à la fois de `N` et `i`. Ici encore, vous donnerez et expliquerez la séquence d'opérations permettant d'arriver à ce résultat.

Question 10 :

Formulez la relation $Read_T(p)$ qui, étant les coordonnées (p) d'un emplacement du tableau `T[p]`, détermine l'ensemble des itérations $S1(i, j)$ accédant à cette case. Déduisez en en la fomule permettant de calculer le nombre d'accès à un emplacement donné `T[p]` (indice : le résultat sera une fonction de N et de p).

Question 12 :

On suppose maintenant que le programme est encours d'exécution, et que l'on s'apprête à exécuter l'instruction $S1(p, q)$, on considèrera donc p, q comme des paramètres du programme. Donnez les définitions des domaines des itérations restant encore à exécuter pour les instruction `S0` et `S1` (vous les noterez $\mathcal{D}_{S0}[p]$ et $\mathcal{D}_{S1}[p, q]$). Déduisez-en la formule permettant de calculer le nombre d'exécution de `S0` et `S1` restant à exécuter d'ici à la fin de la boucle (indice : le résultat sera une fonction des variables p et q).

2 Etude de l'article de Chatterjee et al., PLDI 2001

La deuxième partie du homework porte sur l'analyse d'un article proposant une analyse statique de boucles permettant de déterminer analytiquement le nombre de défauts de cache lors de l'éxécution d'un nid de boucle. Une version électronique de cet article est disponible via l'URL en pied de page [3].

Dans la suite de ce devoir, il vous est demandé d'appliquer la modélisation proposée dans l'article au nid de boucles étudié dans la section précédente. Notez que les objets mathématiques manipulés par iscc sont les mêmes que ceux utilisés dans l'article (dans les deux cas, on manipule des formules de Presburger), mais les notations utilisées dans ce dernier sont différentes.Il est bien évidement attendu que vous reformuliez le modèle et présentiez vos résultat en utilisant les notations d'iscc.

Pour cette exercice, on s'interesse à une organisation du cache et une implantation mémoire spécifiée ci-après.

— Le cache considéré fonctionne par correspondance directe (direct mapped), avec une politique LRU, et est organisé en lignes de $B = 4$ octets. Il contient S =512 entrées pour une capacité totale de $4 * 512 = 2$ko. On rappelle que le type float désigne un flottant codé en 32 bits, soit 4 octets.

— On donne la fonction d'allocation mémoire pour le tableau Y[p] qui s'écrira $\mathcal{L}_Y(p) = \mu_Y + 4p$, de même la fonction d'allocation mémoire pour le tableau T[p][q] s'écrit $\mathcal{L}_T(p, q) = \mu_T + 4096p + 4q$. Quels sont les rôles des paramètre μ_Y et μ_T ? Quel intérêt y-a-t-il à les représenter par des paramètres plutôt que par des valeurs constantes ?

Question 1 :

D'après la sémantique opérationnele du langage C, dans quel ordre relatif s'effectuent les accès en lecture `Y[j]`, `T[i-j][j]`en partie droite de S1 ? Expliquer en quoi cet ordre peut influencer le nombre de défauts de cache ?

Question 2 :

A quoi correspond la fonction Map(m, w, s) définie dans l'équation 3 ? Quel est la signification des variables $s$et w ?

Question 3 :

Expliquez, en les reformulant, comment les conditions exprimées en 3.2.1 permettent de déterminer si un accès est un *interior miss*.

3. https://apps.dtic.mil/dtic/tr/fulltext/u2/a441135.pdf

Question 4 :

Pour les questions 4, 6 et 7, on supposera, pour simplifier la formulation, que seuls les accès au tableau T utilisent le cache, les accès au tableau Y accèderont toujours à la mémoire externe et ne modifieront pas l'état du cache (ceci est possible via un mécanisme de bypassing). Utilisez l'equation (6) pour construire, à l'aide d'iscc, l'ensemble des itérations (i, j) provoquant un *interior miss* pour l'accès `T[i-j]`. Le plus grand soin devra être apporté aux explications, et vous indiquerez la séquence de commandes que vous avez utilisé pour obtenir le résultat.

Question 5 :

Expliquez, en les reformulant, comment les conditions exprimées en 3.2.2 permettent de déterminer si un accès est un *boundary miss*.

Question 6 :

Pour cette question, seuls les accès au tableau T utilisent le cache. Utilisez ensuite l'equation (7), section 3.2.2 pour construire, à l'aide d'iscc, l'ensemble des itérations (i, j) provoquant un *boundary miss* pour pour l'accès `T[i-j]`. Le plus grand soin devra être apporté aux explications, et vous indiquerez la séquence de commandes que vous avez utilisé pour obtenir le résultat.

Question 7 :

Pour cette question, seuls les accès au tableau T utilisent le cache. Construisez, à partir des réponses précédentes, la formule qui donne le nombre de défaut de cache (au pire cas) pour l'exécution de ce nid de boucles, en fonction de N, μ_Y et μ_T

Question 8 :

Reprenez la question 4, en considérant cette fois-ci un cache associatif à deux voies, organisé en lignes de $B = 16$ octets qui contient S =256 entrées pour une capacité totale de $16 * 256 * 2 = 8$ko. Expliquez ensuite pourquoi l'approche proposée ne permet pas de gérer des niveau d'associativité élevés.

Question 9 :

Dans l'exercice, on fait l'hypothèse que les accès mémoire se font dans l'ordre spécifié par le code source C. Cette hypothèse est-elle réaliste ? Quelle sont les étapes du back-end de compilation pouvant remettre en cause cette hypothèse ?

Question 10 :

Dans de nombreux programmes, les tableaux multi-dimensionnels sont alloués dynamiquement en fonction de la taille du problème à traiter. Par exemple, pour une matrice $m \times m$, on va allouer à l'exécution, dès que m est connu, un tableau de unidimensionel de taille m^2 et accéder à une case $T[p][q]$ via la fonction d'accès $\mathcal{L}(p, q) = \mu + m.p + q$. Pensez-vous que l'approche proposée dans cet article permette de gérer ce cas ? Expliquez.