CMP2 – Construction des compilateurs

EPITA – Apprentis promotion 2011
Tous documents (notes de cours, polycopiés, livres) autorisés
Calculatrices et ordinateurs interdits.

Juillet 2009 (1h30)

Une copie synthétique, bien orthographiée, avec un affichage clair des résultats, sera toujours mieux notée qu'une autre demandant une quelconque forme d'effort de la part du correcteur. Une argumentation informelle mais convaincante, sera souvent suffisante. Une lecture préalable du sujet est recommandée.

Écrivez votre nom en haut de la première page du sujet, et rendez-le avec votre copie.

1 Généralités

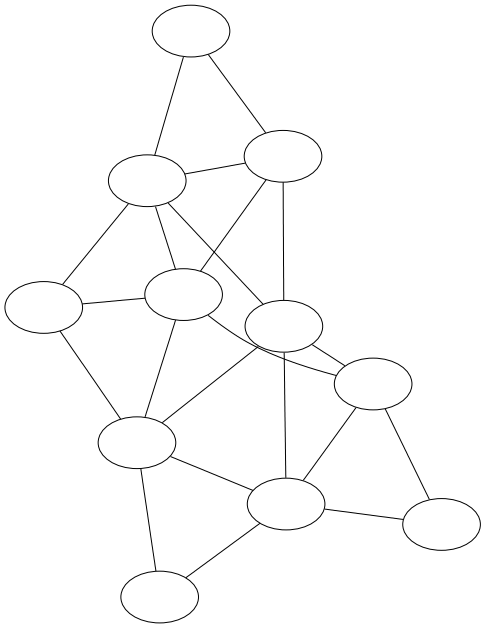
1. Qu'est-ce que le sucre syntaxique ?
2. En donner un exemple (dans le langage de votre choix).
3. Qu'est-ce que le sel syntaxique ?
4. En donner un exemple (dans le langage de votre choix).
5. À quoi sert `make` ? (Note : se contenter de répondre « à compiler » ne rapportera pas de point.)
6. À quoi peut bien servir le bout de code C++ ci-dessous ?

```
template <typename T>
struct fun
{
  typedef T ret;
};

template <typename T>
struct fun<T*>
{
  typedef typename fun<T>::ret ret;
};
```

(Donner un exemple d'utilisation est une bonne idée.)

2 Allocation de registres



Colorer ce graphe d'exclusion mutuelle en 3 registres : R1, R2, R3.

Rendre le sujet en ayant annoté chaque nœud d'un R1, R2, R3.

L'utilisation de stylos de couleurs différentes n'est pas requise, mais sera appréciée.

N'oubliez pas d'écrire votre nom en haut de la première page.

3 Élimination de fonctions inutilisées

Une opération simple qu'un compilateur peut effectuer sur un programme est l'élimination de fonctions inutilisées. Par exemple, dans le programme C++ suivant, `bar` n'est pas utilisée et on pourrait l'éliminer sans changer le comportement du programme.

```
static void foo () {}
static void bar () {}
int main () { foo(); }
```

1. Citer les différentes phases de la partie frontale (*front end*) d'un compilateur comme `tc`.
2. Même question avec la partie terminale (*back end*).
3. À quel(s) endroit(s) dans le compilateur pourrait-on place placer une étape d'élimination de fonctions inutilisées ?
4. Quel(s) avantage(s) y a-t-il à effectuer une telle opération ? (Il y a au moins deux bonnes réponses, mais une seule suffit.)
5. Dans quel cadre de travail spécifique on ne voudrait surtout pas utiliser une telle optimisation ? (Il n'est pas impossible qu'un indice soit présent dans ce sujet.)
6. Si l'on voulait implémenter l'élimination de fonctions inutilisées dans `tc`, quels seraient les types de nœuds de l'AST à considérer ?
7. Comment pourrait-on détecter qu'une fonction peut être éliminée dans `tc` ?
8. Surprise (à moitié) ! Les plus attentifs auront remarqué que cette option est *déjà* présente dans le compilateur Tiger de référence, et vous pouviez l'implémenter dans le cadre d'une extension optionelle du projet. Dans le sujet, ce travail facultatif figurait dans la section « TC-I, Function inlining ». Rappelez en quoi consiste l'optimisation de mise en ligne du corps des fonctions (*inlining of function bodies*).
9. À votre avis, pourquoi l'élimination de fonctions inutilisées est-elle située dans la même section que l'inlining ?
10. Dans quel(s) cas est-il impossible de mettre en ligne une fonction ?

11. Il est possible de supprimer encore plus de code inutilisé. Considérez l'exemple ci-dessous.

```
static void foo () {}
static void bar () { foo(); }
int main () {}
```

Ici, `foo` est bien utilisée par `bar`, mais cette dernière n'est pas utilisée. On pourrait donc supprimer ces deux fonctions et obtenir un programme équivalent après compilation. Proposez un algorithme pour généraliser cette approche.

4 À propos de ce cours

Pour terminer cette épreuve, nous vous invitons à répondre à un petit questionnaire. Les renseignements ci-dessous ne seront bien entendu pas utilisés pour noter votre copie. Ils ne sont pas anonymes, car nous souhaitons pouvoir confronter réponses et notes. En échange, quelques points seront attribués pour avoir répondu. Merci d'avance.

Sauf indication contraire, vous pouvez cocher plusieurs réponses par question. Répondez sur la feuille de QCM qui vous est remise. N'y passez pas plus de dix minutes.

Le cours

1. Quelle a été votre implication dans les cours CMP1, CMP2 et TYLA ?
   - **A** Rien.
   - **B** Bachotage récent.
   - **C** Relu les notes entre chaque cours.
   - **D** Fait les annales.
   - **E** Lu d'autres sources.
2. Ce cours
   - **A** Est incompréhensible et j'ai rapidement abandonné.
   - **B** Est difficile à suivre mais j'essaie.
   - **C** Est facile à suivre une fois qu'on a compris le truc.
   - **D** Est trop élémentaire.
3. Ce cours
   - **A** Ne m'a donné aucune satisfaction.
   - **B** N'a aucun intérêt dans ma formation.
   - **C** Est une agréable curiosité.
   - **D** Est nécessaire mais pas intéressant.
   - **E** Je le recommande.
4. La charge générale du cours en sus de la présence en amphi (relecture de notes, compréhension, recherches supplémentaires, etc.) est
   - **A** Telle que je n'ai pas pu suivre du tout.
   - **B** Lourde (plusieurs heures par semaine).
   - **C** Supportable (environ une heure de travail par semaine).
   - **D** Légère (quelques minutes par semaine).

Les formateurs

5. L'enseignant
   A N'est pas pédagogue.
   B Parle à des étudiants qui sont au dessus de mon niveau.
   C Me parle.
   D Se répète vraiment trop.
   E Se contente de trop simple et devrait pousser le niveau vers le haut.
6. Les assistants
   A Ne sont pas pédagogues.
   B Parlent à des étudiants qui sont au dessus de mon niveau.
   C M'ont aidé à avancer dans le projet.
   D Ont résolu certains de mes gros problèmes, mais ne m'ont pas expliqué comment ils avaient fait.
   E Pourraient viser plus haut et enseigner des notions supplémentaires.

Le projet Tiger

7. Vous avez contribué au développement du compilateur de votre groupe (une seule réponse attendue) :
   A Presque jamais.
   B Moins que les autres.
   C Équitablement avec vos pairs.
   D Plus que les autres.
   E Pratiquement seul.
8. La charge générale du projet Tiger est
   A Telle que je n'ai pas pu suivre du tout.
   B Lourde (plusieurs jours de travail par semaine).
   C Supportable (plusieurs heures de travail par semaine).
   D Légère (une ou deux heures par semaine).
   E J'ai été dispensé du projet.
9. Y a-t-il de la triche dans le projet Tiger ? (Une seule réponse attendue.)
   A Pas à votre connaissance.
   B Vous connaissez un ou deux groupes concernés.
   C Quelques groupes.
   D Dans la plupart des groupes.
   E Dans tous les groupes.

Questions 10-16 Le projet Tiger vous a-t-il bien formé aux sujets suivants ? Répondre selon la grille qui suit. (Une seule réponse attendue par question.)

A Pas du tout
B Trop peu
C Correctement
D Bien
E Très bien

10. Formation au C++.
11. Formation à la modélisation orientée objet et aux *design patterns*.
12. Formation à l'anglais technique.
13. Formation à la compréhension du fonctionnement des ordinateurs.
14. Formation à la compréhension du fonctionnement des langages de programmation.
15. Formation au travail collaboratif.
16. Formation aux outils de développement (contrôle de version, systèmes de construction, débogueurs, générateurs de code, etc.)

Questions 17-29 Comment furent les étapes du projet ? Répondre selon la grille suivante. (Une seule réponse attendue par question ; ne pas répondre pour les étapes que vous n'avez pas faites.)

A Trop facile.
B Facile.
C Nickel.
D Difficile.
E Trop difficile.

17. Rush `.tig` : mini-projet en Tiger (Bistromatig).
18. `TC-0`, Scanner & Parser.
19. `TC-1`, Scanner & Parser, Tâches, Autotools.
20. `TC-2`, Construction de l'AST et pretty-printer.
21. `TC-3`, Liaison des noms et renommage.
22. `TC-E`, Calcul des échappements.
23. `TC-4`, Typage.
24. Désucrage des constructions objets (transformation Tiger → Panther).
25. `TC-5`, Traduction vers représentation intermédiaire.
26. Option `TC-A`, Surcharge des fonctions.
27. Option `TC-D`, Suppression du sucre syntaxique (boucles `for`, comparaisons de chaînes de caractères).
28. Option `TC-B`, Vérification dynamique des bornes de tableaux.
29. Option `TC-I`, Mise en ligne du corps des fonctions.