全体の講評[2010 年度プログラミング言語処理系論]

2010/08/17

東大工学系・電気系専攻（情報基盤センター）
佐藤周行

みなさんへ

レポート、楽しく読みました。使い慣れない manaba を強制したこと、ごめんなさい。真昼間から端末を前ににこにこさせてもらいました。このことの是非はともかく、笑うと寿命が延びるそうなので許してください。

今回のレポートは 14 題のうちの 2 問選択でした。そのうち 1 問は 2 問分の扱いでした。解いた問題数で統計をとると以下のようになりました。

問題＃	1	2	3	4
解答人数	1	3	2	0

5	6	7	8	9
1	3	2	１５	２５

１０	１１	１２	１３	１４
２８	4	1	1	1

問４が問５の前哨戦であることを考えると、実質として、全部の問題に誰かが答えたことになります。問題たちも成仏できるでしょう。めでたいことです。

解答人数をさらに見ると、問９，１０への異常な人気と、教えてもいない８の人気が読み取れます（教えていないからといって、採点は手加減しません)。８はデータ依存性解析＋アーキテクチャへのマッピングという 2 点で HPC の入門的な問題だったのですが、みなさん、興味あるんですか？変な人から変なこと刷り込まれていませんか？

閑話休題。例年、レポートを見ていると、飛びぬけてよくできる人に出くわします。今回はこの 2 人を選びました。権威なんかないし、履歴書に書けるかどうかはなはだ怪しいですがお受け取りください。おめでとうございます。

1 位　山崎健生　君
2 位　原健太朗　君

以下、全体の講評です。個別のものについては、manaba の自分のところを見てください。短く「OK」というのが、おそらく最高のコメントです。長々しく褒め称えるようなコメントはしません。

1．ＸＭＬの十分大きなサブセットが LL(1)であり、手でもパーサが書けることを示す問題です。文法を書いて、predictive parsing table を構成して、その後にコードを書きます。いきなりコードを見せられても困ります。教えられ方も問題かな。理論を教えずにいきなりコードを書く方法を教えられることがあるとしたらそういう暴挙は許されないことです。

2．dc.output には、bison が構築した受理機械の動作が書かれています。それがかけていればＯＫです。実際の入力への動作を書く必要はありません。学部レベルの標準的なコンパイラの本で LALR を調べてください。

3．問題に間違いがありました。 expr と lines の間には¥n が入らなければなりません。ごめんなさい。それを指摘した上で解いた人がいました。教師の間違いを黙って正してくれるというのはありがたいことです。正解は、「シフトが延々と続き、受理機械のスタックを浪費するから」です。もともとのものは reduce が早い段階で次々と起こって、スタックを浪費しません。もちろん、BNF を拡張して expr* のように書けるようにしておくのが良いのですが、動作を知っておくに越したことはありません。授業では「できたパーサの詳細は知らなくても良い。大切なのはパーサができることだ」といったって？いいかげん中年になるとこの程度の矛盾は気にしません。

4．関数引数を扱うには環境を変化させることが必要になります。具体的には名前表をどう変更するかということです。これができれば、自分で本格的なプログラミング言語がかけます。

5．ということで、5 を正解とするには、４．が実装されていなければなりません。がんばってください。さらに、言語を他の人に使ってもらうためには、「言語仕様」を決めて公表しなければなりません。一般論ですが、これができない人がとても多い。実はここまでやり遂げた人が 1 人いました。拍手！でも、言語の仕様を書くのにＸＭＬの規格を例に出したのはちょっとずれていましたね。次回以降の佐藤の宿題としましょう。

6．実行環境のうち、仮想機械をどう定めるかということです。まとめるポイントは問題の通りです。オブジェクトをファーストクラスとして扱い、複数スレッドが書ける言語の実行環境なら、これくらいの論点が必要でしょう。論点を自分で抽出できるようになることも実は必要です。このためには、複数のＶＭの比較解析をするのが常道でしょう。６と６’を並列して出したのはそういう意図もありました。

7．Calling convention を自分で調べる課題です。アセンブリコードを読めることが必須要件です。できたでしょうか。いきなりアセンブリコードに取り掛からずに、ドキュメントを調べた人がいました。立派です。フレームといった場合、ローカル変数の管理

と関係するレジスタの管理も含みます。ここらへんも忘れずに。現在は x86 が支配的で、引数は間違いなくスタックに積みますが、Power 系のものとか、Sparc 系のものとか（あまり近くにはないと思いますが）違いを調べると面白いです。こうやってコンパイラのデザイナーの気持ちがわかっていく…といいな。

８．問題文は行列積になっていませんが、授業中に訂正しましたので、温情はかけません。c[i,j]をひとつ固定すると、そこでの値の変化はループをどう交換しても変わらないことを示してください。データ依存性解析を勉強すれば「依存性は(=,=,<)になるので、ＯＫ」で終わりです。和が可換だからというのは実は間違いです。可換でなくても（例えば割り算でも）ループの順番には依存しません。次はどれが一番速いかですが、理由をちゃんと書いた人がほとんどだったのは立派なことです。「わからないから、全ての場合を計測して答えを出します」という態度はサイエンスの対極にあるものです。さて、ここまできたら是非実測しましょう。実測なしで終わったレポートが一部に見られました。

次に、ループアンロールの話です。これは実測なしのものがほとんどでした。具体的なコードを見せなかったからでしょうか。命令スケジューリングの可能性が広がることが書けていればおおよそ正解です。ここらへんは、命令スケジューリングの最適化を勉強しなければ実感わかないです。

９．Value numbering の問題です。i=g, x=h, v=u までは簡単に計算できます。一部の人が v=u だから、h=u-v=0 と計算していました。変数への代入が起きると、value number がアップデートされます。2 行目の u, v と、6 行目の u,v に与えられている number が異なることに気づいてください。さて、アルゴリズムをきちんと書き直した人、プログラムを書いた人がいました。そこまでやると満点です。

１０．　Dominator tree と natural loop の問題です。Dominator の関係は授業で説明したとおり tree を作ります。Dominator tree を作るアルゴリズムには高速なものが開発されています。調べましょう。Dominator は、いろいろな最適化、特に SSA 変換の基礎です。Natural loop ですが、バックエッジはみなさんわかっているようですが、natural loop の定義くらいは書きましょう。定義に従って、loop を構成するノードを全部見つけるプログラムを書くと満点です。直感は時として人をだまします。

１１．　連立方程式といいいましたが、もとネタは以下の liveness 解析です。

```
    a←0
L1: b←a+1
    c←c+b
    a←b*2
    cmp a, N
    jl L1
    return c
```

in[n] = use[n] ∪ (out[n]-def[n])

out[n]=∪s ∈succ[n] in[s]

この手の方程式の解き方は説明しました。ただし、方程式が生で与えられると、これを満たすものは全部解になります。解は一般に複数になることにも注意しましょう。

１２． 「解は一般に複数になる」が、その解の集合の中で、授業中の解き方で求めたものには、解の最小性（または最大性）が保証されることの証明です。繰り返しの一回一回をＦとすると、in, out を集合の配列と考えて

in, out → F(in, out) になり、これを新たに in, out とすると、Ｆ, F(F), F(F(F)),...と続きます。相手は有限集合ですから、単調に変化すれば、いつかは入力と出力が同じになります。ここまで書ければＯＫです。この F が、x ⊂ y ならば F(x) ⊂ F(y)を満たすとき、最大性、最小性が保証されます。φ, φ ⊂F(φ,φ)、または全体集合を W としたときに W, W ⊃ F(W, W)という関係からはじめましょう。

１３． Knoop の論文のサマリーを書くときは、LCM が、いくつかのデータフロー方程式を解くことで定式化されることと、各々のデータフロー方程式の意味を解説する 2 段階で十分です。証明まではまぁ、いいや。逆に言えば、この２つをスキップすることは、論文、わからなかったか、ちゃんと読んでいないということですね。どうでしょうか。

１４． SSA 変換をするには、dominance frontier を求めることが必要で、そのためには dominator を理解しなくちゃ…ということで、一般のフローグラフを相手にするときは結構敷居が高いです。一方、問題のようにフローグラフそのものが帰納的に構成できるときは、SSA 変換も帰納的にかけちゃいます。プログラミングの手間が劇的に減るということで、まず簡単な言語を作って実験して…ということが研究の方法論として有効であることの証拠になっています。もっともちゃんとしたところに論文書くときはフルセットの言語でのベンチマークが求められ、簡単な言語での実験は結局は日の目をみない可能性が大なのですが、フルセットの言語を相手に手も足も出ない状態よりは、さっさと実験環境を整えることの方がずっと良いでしょう。

最後になりましたが、manaba という e-learning システムは、今回のレポート管理をとても楽にしてくれました。使う機会をいただいた朝日ネット、また NTT コミュニケーションズの方々に感謝いたします。