Errata and Additions for the Third Printing of Advanced Compiler Design and Implementation

Steven S. Muchnick

14 November 1998

Change History

Date	Change
9 Sept 1998	Added correction for p. 185, line 18
14 Nov 1998	Added corrections for pp. 86, 539, 540, and 542; deleted corrections for pp. 274, 286, and 317

Chapter 3. Symbol Table Structure

P. 63, Figure 3.15 *(continued)*: Change lines -11 through -2 to

```
if Short_Const(symdisp) & stflag then
   ||  generate load or store
   Gen_Inst(Opcode,regc ⊕ ",[" ⊕ reg1 ⊕ "+" ⊕ Int_Char(symdisp) ⊕ "]"); return
elif Short_Const(symdisp) then
   ||  generate sethi and load or store
   Gen_Inst(Opcode,"[" ⊕ reg1 ⊕ "+" ⊕ Int_Char(symdisp) ⊕ "]," ⊕ regc); return
fi
Gen_Inst("sethi","%hi(" ⊕ Int_Char(symdisp) ⊕ ")," ⊕ reg1)
if stflag then
   Gen_Inst(Opcode,regc ⊕ ",[" ⊕ reg1 ⊕ ⊕ "+%lo(" ⊕ Int_Char(symdisp) ⊕ ")]")
else
   Gen_Inst(Opcode,"[" ⊕ reg1 ⊕ ⊕ "+%lo(" ⊕ Int_Char(symdisp) ⊕ ")]," ⊕ regc)
fi
```

Chapter 4. Intermediate Representations

P. 86, Figure 4.9: In the "`Inst[4]`" entry, change "`kind:if`" to "`kind:binif`".

Chapter 6. Producing Code Generators Automatically

P. 154, Figure 6.14, line 11: Change "`r1.rt ∉ R`" to "`r1.rt↓1 ∉ R`".

P. 185, line 18: Change "$Dfn(v_i) > Dfn(w)$" to "$Dfn(v_i) < Dfn(w)$".

Chapter 7. Control-Flow Analysis

P. 205, Figure 7.38, line 4: Change "`Structof`" to "`StructOf`".

P. 207: In line 9 of text, change "`DFS_Preorder( )`" to "`DFS_Postorder( )`", and, in line 10 change "preorder" to "postorder".

P. 210, Figure 7.44, line 12: Change "`nset`" to "`NodeSet`".

P. 211, line -6: Append the following sentence "The function `Path(`n`,`m`,`I`)` returns `true` if there is a path from n to m such that all the nodes in it are in I and `false` otherwise." to this line.

Chapter 9. Dependence Analysis & Dependence Graphs

P. 274, Figure 9.6: In line 4, delete "`set of`".

P. 286: In Figure 9.20, interchange the labels on the dashed edges from `B1` to `R1` and `R2`.

Chapter 12. Early Optimizations

P. 342: In line 8, change "left side" to "first of them" and, in line 9, change "right" to "second".

P. 356, final paragraph: Making the algorithm run in less than in $O(n^2)$ time depends on the implementation chosen for the set `ACP`. If it is implemented as a linearly searched array, the algorithm is order n^2; if it is implemented as a balanced tree, it is order $n \cdot \log n$; and if it is hashed, the time can be reduced to order n.

P. 357, Figure 12.24: Change line -5 to the following:

```
if Block[m][i].kind = valasgn & Block[m][i].opd.kind = var
   & Block[m][i].left ≠ Block[m][i].opd.val then
```

Chapter 13. Redundancy Elimination

P. 392, Figure 13.11, line 8: Change "`set of BinExp`" to "`BinExpIntPairSet`".

Chapter 15. Procedure Optimizations

P. 462, line 1 of the text: Change "reegisters" to "registers".

Chapter 17. Code Scheduling

P. 539, Figure 17.5, line -3: Delete the initial "`(`".

P. 539, line -9 of the text: Insert "of" after "pairs".

P. 540, Figure 17.6, line 5: Change "`Node`" to "`integer`".

P. 542, line 5: Change "`Etime(i)`" to "`Etime[i]`".

Chapter 19. Interprocedural Analysis & Optimization

P. 653, Table 19.12: In the caption, change "formal-parameters" to "formal parameters".

Technical Index

Also, add the following:

arybinasgn, 87 aryref, 87 aryunasgn, 87 aryvalasgn, 87 binasgn, 83
binexp, 85 binif, 83 bintrap, 83 call, 83 callasgn, 84
callreg, 89 callreg2, 89 callreg3, 90 callregasgn, 90 castasgn, 83
condasgn, 83 const, 84, 90 else, 87 eltasgn, 83 endif, 87
goto, 83, 89 gotoaddr, 89 indasgn, 83 indeltasgn, 83 label, 83, 89
listexp, 85 loadmem, 89 noexp, 85 receive, 83 regbin, 89
regbinif, 89 regvalif, 89 regbintrap, 89 regvaltrap, 89 regcond, 89
regelt, 89 regun, 89 regunif, 89 reguntrap, 98 regval, 89
return, 84, 90 retval, 84, 90 sequence, 84, 90 strbinif, 87 strunif, 87
strvalif, 87 stormem, 89 type, 84, 90 unasgn, 83 unexp, 85
unif, 83 untrap, 83 valasgn, 83 valif, 83 valtrap, 83