

- transformation of C programs. In *CC '02: Proceedings of the 11th International Conference on Compiler Construction*, pages 213–228, London, UK. Springer-Verlag.
- Norman Ramsey and João Dias. 2005 (September). An applicative control-flow graph based on Huet's zipper. In *ACM SIGPLAN Workshop on ML*, pages 101–122.
- David A. Schmidt. 1998. Data flow analysis is model checking of abstract interpretations. In ACM, editor, *Conference Record of the 25th Annual ACM Symposium on Principles of Programming Languages*, pages 38–48.
- Raja Vallée-Rai, Etienne Gagnon, Laurie J. Hendren, Patrick Lam, Patrice Pominville, and Vijay Sundaresan. 2000. Optimizing Java bytecode using the Soot framework: Is it feasible? In *CC '00: Proceedings of the 9th International Conference on Compiler Construction*, pages 18–34, London, UK. Springer-Verlag.
- David B. Whalley. 1994 (September). Automatic isolation of compiler errors. *ACM Transactions on Programming Languages and Systems*, 16 (5):1648–1659.

A. Code for fixpoint

```

data TxFactBase n f
  = TxFB { tfb_fbase :: FactBase f
          , tfb_rg   :: RG n f C C -- Transformed blocks
          , tfb_cha  :: ChangeFlag
          , tfb_lbls :: LabelSet }
-- Set the tfb_cha flag iff
-- (a) the fact in tfb_fbase for or a block L changes
-- (b) L is in tfb_lbls.
-- The tfb_lbls are all Labels of the *original*
-- (not transformed) blocks

updateFact :: DataflowLattice f -> LabelSet -> (Label, f)
            -> (ChangeFlag, FactBase f)
            -> (ChangeFlag, FactBase f)
updateFact lat lbls (lbl, new_fact) (cha, fbase)
  | NoChange <- cha2      = (cha, fbase)
  | lbl `elem` LabelSet lbls = (SomeChange, new_fact)
  | otherwise              = (cha, new_fact)
where
  (cha2, res_fact)
    = case lookupFact fbase lbl of
        Nothing -> (SomeChange, new_fact)
        Just old_fact -> fact_extend lat old_fact new_fact
    new_fact = extendFactBase fbase lbl res_fact

fixpoint :: forall n f. Edges n
          => Bool -- Going forwards?
          -> DataflowLattice f
          -> (Block n C C -> FactBase f
              -> FuelMonad (RG n f C C, FactBase f))
          -> FactBase f -> [(Label, Block n C C)]
          -> FuelMonad (RG n f C C, FactBase f)
fixpoint is_fwd lat do_block init_fbase blocks
  = do { fuel <- getFuel
        ; tx_fb <- loop fuel init_fbase
        ; return (tfb_rg tx_fb,
                  tfb_fbase tx_fb `delFromFactBase` blocks) }
  -- The outgoing FactBase contains facts only for
  -- Labels *not* in the blocks of the graph
where
  tx_blocks :: [(Label, Block n C C)]
             -> TxFactBase n f -> FuelMonad (TxFactBase n f)
  tx_blocks [] tx_fb = return tx_fb
  tx_blocks ((lbl,blk):bs) tx_fb = tx_block lbl blk tx_fb
                                     >>= tx_blocks bs

tx_block :: Label -> Block n C C
          -> TxFactBase n f -> FuelMonad (TxFactBase n f)
tx_block lbl blk tx_fb@(TxFB { tfb_fbase = fbase
                               , tfb_lbls = lbls
                               , tfb_rg   = blks
                               , tfb_cha  = cha })
  | is_fwd && not (lbl `elem` LabelSet fbase)
  = return tx_fb -- Note [Unreachable blocks]
  | otherwise
  = do { (rg, out_facts) <- do_block blk fbase
        ; let (cha', fbase')
              = foldr (updateFact lat lbls) (cha, fbase)
                (factBaseList out_facts)
        ; return (TxFB { tfb_lbls = extendLabelSet lbls lbl
                        , tfb_rg   = rg `RGCatC` blks
                        , tfb_fbase = fbase'
                        , tfb_cha  = cha' }) }

loop :: Fuel -> FactBase f -> FuelMonad (TxFactBase n f)
loop fuel fbase
  = do { let init_tx_fb = TxFB { tfb_fbase = fbase
                                , tfb_cha  = NoChange
                                , tfb_rg   = RGNil
                                , tfb_lbls = emptyLabelSet }
        ; tx_fb <- tx_blocks blocks init_tx_fb
        ; case tfb_cha tx_fb of
            NoChange -> return tx_fb
            SomeChange -> setFuel fuel >>
                loop fuel (tfb_fbase tx_fb) }

```

B. Index of defined identifiers

This appendix lists every nontrivial identifier used in the body of the paper. For each identifier, we list the page on which that identifier is defined or discussed—or when appropriate, the figure (with line number where possible). For those few identifiers not defined or discussed in text, we give the type signature and the page on which the identifier is first referred to.

Some identifiers used in the text are defined in the Haskell Prelude; for those readers less familiar with Haskell, these identifiers are listed in Appendix D.

Add :: Operator not shown (but see page 7).
 ag let- or λ -bound on page 10.
 analyzeAndRewriteFwd defined on page 10.
 arbNode defined on page 10.
 ARF defined on page 9.
 arfBlock defined on page 10.
 arfBody defined on page 10.
 arfGraph defined on page 10.
 arfNode defined on page 10.
 Assign defined in Figure 1 on page 3.
 b1 let- or λ -bound on page 4.
 b2 let- or λ -bound on page 4.
 BCat defined in Figure 2 on page 3.
 BFirst defined in Figure 2 on page 3.
 Binop :: Operator -> Expr -> Expr -> Expr not shown (but see page 7).
 Blast defined in Figure 2 on page 3.
 blk let- or λ -bound on page 14.
 blks let- or λ -bound on page 14.
 Block defined in Figure 2 on page 3.
 blocks let- or λ -bound on page 14.
 BMiddle defined in Figure 2 on page 3.
 body let- or λ -bound on page 10.
 body' let- or λ -bound on page 10.
 Branch defined in Figure 1 on page 3.
 bs let- or λ -bound on page 4.
 bs1 let- or λ -bound on page 4.
 bs2 let- or λ -bound on page 4.
 C defined in Figure 2 on page 3.
 cha let- or λ -bound on page 14.
 cha' let- or λ -bound on page 14.
 cha2 let- or λ -bound on page 14.
 ChangeFlag defined in Figure 4 on page 5.
 CondBranch defined in Figure 1 on page 3.
 DataflowLattice defined in Figure 4 on page 5.
 delFromFactBase :: FactBase f -> [(Label,f)] -> FactBase f not shown (but see page 14).
 do_block let- or λ -bound on page 14.
 Edges defined in Figure 2 on page 3.
 elemFactBase :: Label -> FactBase f -> Bool not shown (but see page 14).
 elemLabelSet :: Label -> LabelSet -> Bool not shown (but see page 14).
 emptyLabelSet :: LabelSet not shown (but see page 14).
 entryLabel defined in Figure 2 on page 3.
 ex let- or λ -bound in Figure 2 on page 3.
 exit let- or λ -bound on page 10.
 exit' let- or λ -bound on page 10.
 Expr defined on page 3.
 extendFactBase :: FactBase f -> Label -> f -> FactBase f not shown (but see page 14).
 extendLabelSet :: LabelSet -> Label -> LabelSet not shown (but see page 14).
 Fact defined in Figure 4 on page 5.

FactBase defined in Figure 4 on page 5.
factBaseLabels :: FactBase f -> [Label] not shown (but see page 14).
factBaseList :: FactBase f -> [(Label, f)] not shown (but see page 14).
fact_bot defined in Figure 4 on page 5.
fact_extend defined in Figure 4 on page 5.
fb let- or λ -bound on page 10.
fbase let- or λ -bound on page 10.
fbase' let- or λ -bound on page 14.
fixpoint defined on page 11.
forwardBlockList defined on page 10.
fp_lattice defined in Figure 4 on page 5.
fp_rewrite defined in Figure 4 on page 5.
fp_transfer defined in Figure 4 on page 5.
Fuel defined on page 10.
fuel let- or λ -bound on page 14.
FuelMonad defined on page 8.
FwdPass defined in Figure 4 on page 5.
FwdRes defined in Figure 4 on page 5.
FwdRewrite defined in Figure 4 on page 5.
FwdTransfer defined in Figure 4 on page 5.
fx let- or λ -bound on page 10.
getFuel :: FuelMonad Fuel not shown (but see page 14).
GMany defined in Figure 2 on page 3.
GNil defined in Figure 2 on page 3.
Graph defined in Figure 2 on page 3.
graphOfAGraph defined on page 10.
gSplice defined on page 4.
GUnit defined in Figure 2 on page 3.
init_fbase let- or λ -bound on page 14.
init_tx_fb let- or λ -bound on page 14.
is_fwd let- or λ -bound on page 14.
Just0 defined in Figure 2 on page 3.
Label defined in Figure 2 on page 3.
LabelMap (a type) not shown (but see page 14).
LabelSet (a type) not shown (but see page 14).
lat let- or λ -bound on page 14.
lbl let- or λ -bound on page 14.
lbls let- or λ -bound on page 14.
lookupFact :: FactBase f -> Label -> Maybe f not shown (but see page 14).
loop defined on page 14.
Maybe0 defined in Figure 2 on page 3.
mb_g let- or λ -bound on page 10.
mkFactBase :: [(Label, f)] -> FactBase f not shown (but see page 5).
mkFRewrite' defined in Figure 4 on page 5.
mkFTransfer' defined in Figure 4 on page 5.
new_fact let- or λ -bound on page 14.
new_fbase let- or λ -bound on page 14.
NoChange defined in Figure 4 on page 5.
Node defined in Figure 1 on page 3.
node defined on page 6.
NoFwdRes defined in Figure 4 on page 5.
normalizeBody defined on page 9.
Nothing0 defined in Figure 2 on page 3.
0 defined in Figure 2 on page 3.
old_fact let- or λ -bound on page 14.
out_facts let- or λ -bound on page 14.
pairFwd defined on page 7.
pass let- or λ -bound on page 10.
pass' let- or λ -bound on page 10.
res_fact let- or λ -bound on page 14.
RG defined in Figure 6 on page 9.

rg let- or λ -bound on page 10.
RGCatC defined in Figure 6 on page 9.
RGCat0 defined in Figure 6 on page 9.
RGNil defined in Figure 6 on page 9.
RGUnit defined in Figure 6 on page 9.
rw let- or λ -bound on page 10.
setFuel :: Fuel -> FuelMonad () not shown (but see page 14).
SomeChange defined in Figure 4 on page 5.
stdMapJoin :: Ord k => JoinFun v -> JoinFun (Map.Map k v) not shown (but see page 8).
Store defined in Figure 1 on page 3.
successors defined in Figure 2 on page 3.
tfb_cha defined on page 14.
tfb_fbase defined on page 14.
tfb_lbls defined on page 14.
tfb_rg defined on page 14.
thing let- or λ -bound on page 9.
toAGraph defined on page 8.
transfer defined on page 9.
transfer_fn defined on page 6.
tx_block defined on page 14.
tx_blocks defined on page 14.
TxFactBase defined on page 14.
TxFB defined on page 14.
tx_fb let- or λ -bound on page 14.
updateFact defined on page 14.
Var defined on page 3.
withFuel defined on page 10.

C. Undefined identifiers

addBlock (p4), AGraph (p8),
analyzeAndRewriteFwdBody (p5), Body (p5), BodyCat (p4),
BUnit (p10), changeIf (p5), constFactAdd (p8),
constLattice (p8), constProp (p8), constPropPass (p8),
f' (p6), facts (p10), FwdTransfers (p6), HasConst (p8),
iterFwdRew (p7), iterFwdRw (p7), mapE (p8), mapUnion (p4),
Monad (p7), noFwdRw (p7), PElem (p5), rewriteE (p8), rw1 (p7),
rw1a (p7), rw2 (p7), s_exp (p8), simplify (p8),
thenFwdRew (p7), thenFwdRw (p7), Top (p8), varHasConst (p8),
WithTop (p5).

D. Identifiers defined in Haskell Prelude or a standard library

!, \$, &, &&, *, +, ++, -, ., /, ==, >, >=, >>, >>=, Bool, const,
curry, Data.Map, drop, False, flip, fmap, foldl, foldr,
fst, head, id, Int, Integer, Just, last, liftM, map,
Map.empty, Map.insert, Map.lookup, Map.Map, mapM_,
Maybe, not, Nothing, otherwise, return, snd, String, tail,
take, True, uncurry, undefined.