


A. Code for fixpoint

```haskell
fixpoint :: forall n f. Edges n
data TxFactBase n f
  = TxFB { tfb_fbase :: FactBase f
         , tfb_rg :: RG n f 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

fixpoint is_fwd lat do_block init_fbase blocks
updateFact lat lbls (lbl, new_fact) (cha, fbase)
  = do { fuel <- getFuel
         ; (rg, out_facts) <- do_block blk fbase
         | otherwise = (cha, new_fbase)
         -- 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
         where
             (cha2, res_fact)
             = case lookupFact fbase lbl of
                 Nothing -> (SomeChange, new_fbase)
                 Just old_fact -> fact_extend lat old_fact new_fact
             new_fbase = extendFactBase fbase lbl res_fact

fixpoint = forall n f. Edges n
  = Bool
  -- Going forwards?
        = DataflowLattice f
        = FactBase f
        = FactBase f
        = FactBase f
        = FactBase f
        = DataflowLattice f

fixpoint is_fwd lat do_block init_fbase blocks
  = do { fuel <- getFuel
         ; tx_fb <- loop fuel init_fbase
         ; return (txf_rb_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])
             -- Not [Unreachable blocks]s
             -- otherwise
             = 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' })
         | is_fwd && not (lbl 'elemLabelSet' lbls)
             = return tx_fb
             -- Some identifiers used in the text are defined in the Haskell Prelude;
             -- for those readers less familiar with Haskell, these identifiers are
             -- 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 Haskell Prelude;
             -- for those readers less familiar with Haskell, these identifiers are
             -- defined or discussed in text, we give the type signature and the page
             -- on which the identifier is first referred to.
             -- Add :: Operator not shown (but see page 7).
             -- ag let- or lambda-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 lambda-bound on page 4.
             -- b2 let- or lambda-bound on page 4.
             -- Block defined in Figure 2 on page 3.
             -- blocks let- or lambda-bound on page 4.
             -- BMiddle defined in Figure 2 on page 3.
             -- body let- or lambda-bound on page 10.
             -- body' let- or lambda-bound on page 10.
             -- Branch defined in Figure 1 on page 3.
             -- bs let- or lambda-bound on page 4.
             -- bs1 let- or lambda-bound on page 4.
             -- bs2 let- or lambda-bound on page 4.
             -- C defined in Figure 2 on page 3.
             -- cha let- or lambda-bound on page 14.
             -- char let- or lambda-bound on page 14.
             -- char' let- or lambda-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 -> FactBase f not shown (but see page 14).
             -- do_block let- or lambda-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.
             -- exit let- or lambda-bound on page 10.
             -- exit' let- or lambda-bound on page 10.
             -- Expr defined on page 3.
             -- extendFactBase :: FactBase f -> FactBase f not shown (but see page 14).
             -- factBaseList 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.

Fact instruct defined in Figure 4 on page 5.

Fact lattice defined in Figure 4 on page 5.

Fact_rewrite defined in Figure 4 on page 5.

Fact transfer defined in Figure 4 on page 5.

Fuel defined on page 10.

Fuel let- or \lambda-bound on page 10.

Fuel fact defined on page 10.

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 \lambda-bound on page 10.

GetFuel :: FuelMonad Fuel not shown (but see page 14).

GMap 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.

InitFbase let- or \lambda-bound on page 14.

InitTx_FB let- or \lambda-bound on page 14.

Is_fwd let- or \lambda-bound on page 14.

Just0 defined in Figure 2 on page 3.

Label defined in Figure 2 on page 3.

LabelMap (a tuple) not shown (but see page 14).

LabelSet (a tuple) not shown (but see page 14).

Lat let- or \lambda-bound on page 14.

Lbl let- or \lambda-bound on page 14.

LBl let- or \lambda-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.

Mbg let- or \lambda-bound on page 10.

mkFactBase :: [(Label, f)] -> FactBase f not shown (but see page 5).

mkFTransfer' defined in Figure 4 on page 5.

new_fact let- or \lambda-bound on page 14.

new_fbase let- or \lambda-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.

Old_fact let- or \lambda-bound on page 14.

Out_facts let- or \lambda-bound on page 14.

PairFwd defined on page 7.

Pass let- or \lambda-bound on page 10.

Pass' let- or \lambda-bound on page 10.

Res_fact let- or \lambda-bound on page 14.

RG defined in Figure 6 on page 9.

rg let- or \lambda-bound on page 10.

RGCatC defined in Figure 6 on page 9.

RGCatO defined in Figure 6 on page 9.

RGNil defined in Figure 6 on page 9.

RGNUnit defined in Figure 6 on page 9.

rw let- or \lambda-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.

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_Feb let- or \lambda-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.