# Test Report Transitive Closure

By Luka Hale and Christian Konig

To test our implementation of the trClos function we were able to reuse an example from the book: on page 175 is a function transR that evaluates is a relation is a transitive closure. The only difficulty was, that the example relies on an implementation of relations as sets, while our task was about implementing relations as lists.

We were able to work around this mismatch by creating a helper method called relToSet, that creates an ordered set from a list.

isTrans :: Ord a => Rel a -> Bool

isTrans x = isTrans2 (relToSet x)

isTrans2 :: Ord a => Set (a,a) -> Bool

isTrans2 (Set []) = True

isTrans2 (Set s) = and [trans pair (Set s) | pair <- s] where

trans (x,y) (Set r) = and [inSet (x,v) (Set r) | (u,v) <- r, u ==y]

relToSet :: Ord a => Rel a -> Set (a,a)

relToSet x = relToSetR x emptySet

relToSetR :: Ord a => Rel a -> Set (a,a) -> Set (a,a)

relToSetR [] y = y

relToSetR (x:xs) y = relToSetR xs (insertSet x y)

At first we used this function for some manual testing like this:

s1 = [(1,2),(2,3),(3,4),(4,5)]

manualTest = (not (isTrans s1) && isTrans (trClos s1))

In order to make use of automated random testing, we created a function to generate random relations

getRandomRelation :: Int -> IO (Rel Int)

getRandomRelation 0 = return []

getRandomRelation n = do fstX <- getRandomInt 9 -- x range

fstY <- getRandomInt 9 -- y range

randPairs <- getRandomRelation (n-1)

return ((fstX, fstY) : randPairs)

Wiring both parts together to run a given number of random test is as simple as this:

autoTest :: Int ->IO Bool

autoTest testCount = if testCount > 0

then do

randRel <- getRandomRelation 9 -- relation pair count

let tr = trClos randRel

testRes <- autoTest (testCount -1)

testResR <- autoTestR tr

return (testRes && testResR)

else return True

The impementation of autoTestR was tricky. Not all generated relations make a possible input, meaning that the output is not transitive. We came up with the test condition that reads like “*If the trClos function added elements to the relation, the output has to be transitive.*” The cases we don't look at are therefore inputs that are already transitive closure and inputs for which a transitive closure cannot be produced.

autoTestR :: Rel Int -> IO Bool

autoTestR x = return ((length x == length (trClos x)) || (isTrans (trClos x)))

We ran the automated tests several times with parameters up to 10.000. The calculation takes only a few seconds. The result was True for every run.