Scribe April - Week 1
Program Analysis (CS6843)

April 12, 2017

Akshay Potey CS16M038

Introduction

1. Identifying shape of data structures in the program like trees,lattices,etc.
2. Say , a data structure is height balanced , we can then safely say that the running time is logarithmic in nature.
3. Another example is identifying a data structure as list.
The advantages includes improving cache performance by fetching the next node in the list.
4. Identifying a tree data structure an help us to identify two different sub-trees which can enable parallel execution of operations.

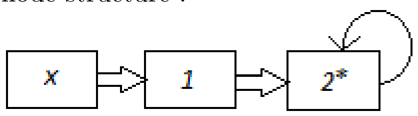
Limitations of pointer analysis

Can we identify shape of the data structure using pointer analysis? To answer the question let's try to identify the structure from the following program.

```
$listReverse(List\ x)\{$
    $assert(\text{"}x\ is\ acyclic\ singly\ linked\ list\text{"});$
    $for(y = null; x; )\{$
        $t = y;$
        $y = x;$
        $x = x \rightarrow next;$
        $y \rightarrow next = t$
    $\}$
    $x = y;$
    $t = null;$
    $y = null;$
$\}$
```

The following information can be deduced from the analysis :
1. x is pointing to one node at the end.
2. Number of temporaries is not fixed.
3. Loop termination is not sure. Although considering the assertion we can ensure loop termination but even then the number of iterations cant be predicted.

The solution for this: Use one variable for rest of the list.We model the list as a two node structure :



Lets perform pointer analysis again with the updated structure.
The following are the points to set for various variables :-

$x \rightarrow \{1, 2^*, null\}$
$y \rightarrow \{1, 2^*, null\}$
$t \rightarrow \{1, 2^*, null\}$
Since we have a finite representation for the list , we know that the loop terminates i.e, we reach a fixed point.

Precision

Precision reduces along slist,tree, DAG and cycle.

Challenges

1. The size of the data structure is statically unknown.
2. Same pointers iterates through various nodes.
3. Shape needs to be determined from program variables.

Shape Analysis

What do we mean by shape analysis?
Shape analysis tries to identify the shape of the pointer i.e. we want to identify the data structure reachable from the pointer ,in our case a singly list or a tree or a DAG or a cycle.The shape we infer is the shape we can reach from that pointer.
The analysis is field insensitive analysis.

Tree ,DAG or Cycle?

To identify structure the proposed analysis maintains three data structures :-

1. **Interference Matrix** Encodes common reachability.

2. **Direction Matrix** Encodes direct reachability.

3. **Shape** The shape reachable from that pointer.

Interference Matrix

- A 2X2 symmetric matrix, $I[n][n]$, taking values over {0,1},where n is the number of pointers.
- $I[p][q] = 1$ if the two pointers p and q have path to common memory node.

Direction Matrix

- A 2X2 non-symmetric matrix ,$D[n][n]$, taking values over {0,1}, where n is the number of pointers.
- $D[p][q] = 1$ if whatever q is pointing to , p has a path to it.

Shape

- A n-array where n is the number of pointers , taking values over the set $\{tree, DAG, cycle\}$.The set may include other structures as well,like singly list etc., depending on the shape analysis.
- Stores the shape reachable from that pointer.
- We give a guarantee on must not analysis, say, we have a DAG then it is definitely not a cycle.

Inference Rules

The statements of following formats are of interest to us :

- $p = malloc(..)$
- $p = q$
- $p = q \rightarrow f$
- $p = \&(q \rightarrow f)$
- $p = q\ op\ k$
- $p = null$
- $p \rightarrow f = q$
- $p \rightarrow f = null$

1. **p = malloc(..)**
   - $I[p][p] = 1$, $D[p][p] = 1$ and $p.shape = tree(most\ precise\ information)$
   - Kill other information for p, as any old information(if any) is lost.

2. **p = q**
   - $I[p][p] = 1, D[p][p] = 1$ if $D[q][q] = 1$ and $I[q][q] = 1$.
   - $D[p][s] = 1$ if $D[q][i] = 1$.
   - $D[s][p] = 1$ if $D[s][q] = 1$.
   - $I[p][s] = 1$ if $I[q][s] = 1$.
   - Kill any other information for p.
   - $p.shape = q.shape$.

3. **p = q $\rightarrow$ f**
   - $D[s][p] = 1$ if $I[s][q] = 1$ and s $\neq$ q.
   - $D[p][s] = 1$ if $D[q][s] = 1$ and s $\neq$ p and s $\neq$ q.
   - $D[p][q] = 1$ if $q.shape =$ cycle.
   - $D[p][p] = 1$ if $D[q][q] = 1$.
   - $I[p][s] = 1$ if $I[q][s] = 1$ and s $\neq$ p.
   - $I[p][p] = 1$ if $I[q][q] = 1$.
   - p.shape = q.shape.