

# Program Structure Trees

## A comparison of different algorithms

Tobias Großer

University Passau

January 25, 2010

## 1 Introduction

## 2 Components

- Control Flow Graph
- Region
- Refined Region
- Program Structure Tree

## 3 Different algorithms

- Regions in Structured Programs
- The Program Structure Tree
- The Refined Program Structure Tree
- Dominance Tree based RPST

## 4 Region detection in static program analysis

- Convert an CFG to an CFA

# Outline

## 1 Introduction

## 2 Components

- Control Flow Graph
- Region
- Refined Region
- Program Structure Tree

## 3 Different algorithms

- Regions in Structured Programs
- The Program Structure Tree
- The Refined Program Structure Tree
- Dominance Tree based RPST

## 4 Region detection in static program analysis

- Convert an CFG to an CFA

# Outline

## 1 Introduction

## 2 Components

- Control Flow Graph
- Region
- Refined Region
- Program Structure Tree

## 3 Different algorithms

- Regions in Structured Programs
- The Program Structure Tree
- The Refined Program Structure Tree
- Dominance Tree based RPST

## 4 Region detection in static program analysis

- Convert an CFG to an CFA

# Control Flow Graph (CFG)

## Definition (Basic Block - BB)

- Contains a list of statements.
- Statements are executed linear.
- The last statement is a branch statement.

## Definition (Control Flow Graph - CFG)

- $CFG = (V, E)$ .
- $V$  is a set of basic blocks (bbs).
- $E$  is a set of edges.
- Execution starts at the entry bb.
- Execution ends at any “return”.

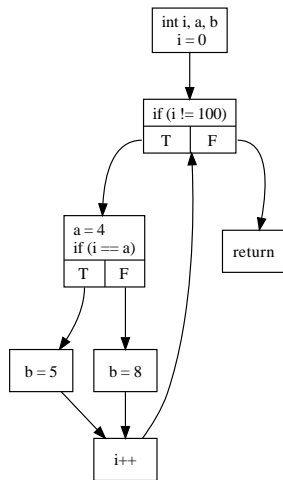


Figure: Control Flow Graph

# Control Flow Graph (CFG) - Example

```
void foo() {  
    int i, a, b;  
    for (i=0; i!=100; i++) {  
        a=3;  
        if (i==a)  
            b=5;  
        else  
            b=8;  
    }  
    return;  
}
```

Figure: C Source Code

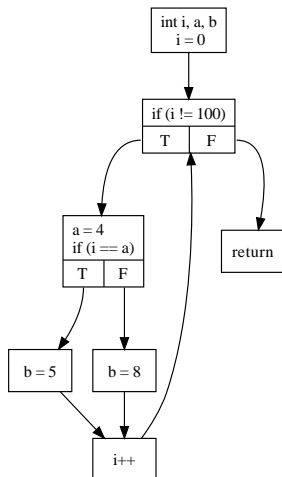


Figure: Control Flow Graph

# Region

## Definition (Region)

- Subgraph of the CFG
- Only two connections to the remaining CFG.
- Incoming edge is called entry edge.
- Outgoing edge is called exit edge.

## Definition (Canonical Region)

A region that cannot be built out of a set of smaller regions.

## Definition (Trivial Region)

A region that contains just one basic block.

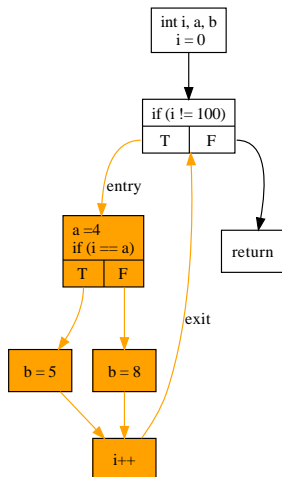


Figure: Region in the CFG

# Region - Move into Function

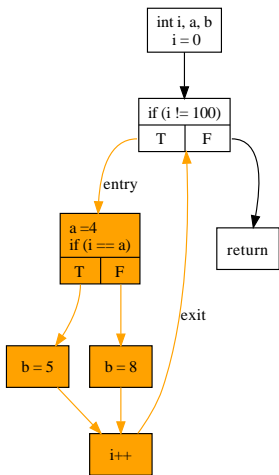


Figure: Original CFG

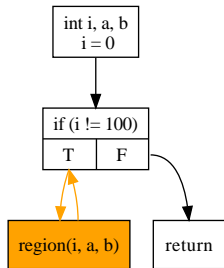


Figure: call region()

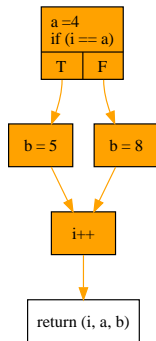


Figure: region()

# Refined Region

## Definition (Merge block)

An empty basic block that joins several edges.

## Definition (Refined Region)

A connected subgraph of the CFG, that can be transformed to a region, by inserting merge bbs.

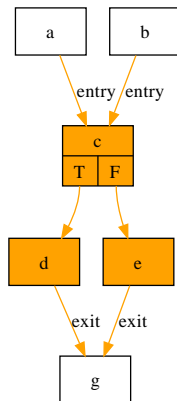


Figure: Refined region

## Refined Region - Add Merge Blocks

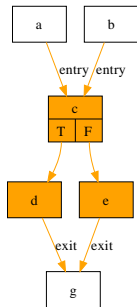


Figure: Refined Region

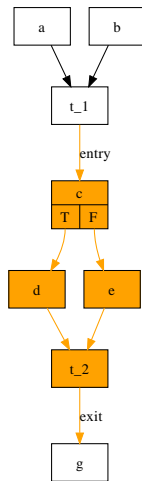


Figure: Region with merge blocks

# The Program Structure Tree

## Definition (Program Structure Tree - PST)

- A tree built out of the canonical regions of a function.
- Region A is an ancestor of B, if A is contained in B.

## Definition (Refined PST - RPST)

- A PST build with refined regions.

# Program Structure Tree - Example

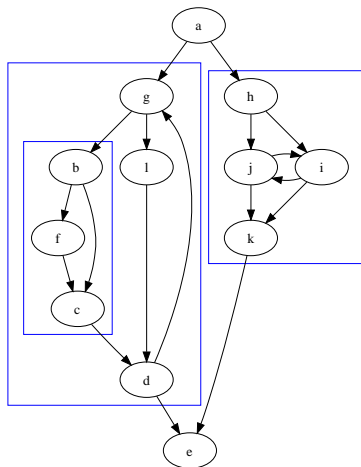


Figure: Program Structure Tree

# Refined Program Structure Tree - Example

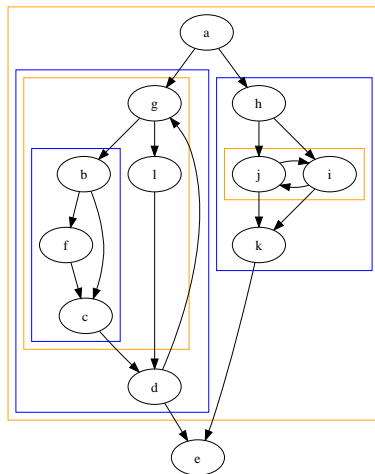


Figure: Refined Program Structure Tree

# Outline

## 1 Introduction

## 2 Components

- Control Flow Graph
- Region
- Refined Region
- Program Structure Tree

## 3 Different algorithms

- Regions in Structured Programs
- The Program Structure Tree
- The Refined Program Structure Tree
- Dominance Tree based RPST

## 4 Region detection in static program analysis

- Convert an CFG to an CFA

# Regions in Structured Programs

Developed n/a

Generality Structured control flow

Precision All Simple Regions

Implementation Afford Easy

Prerequisite Program structure information.

Runtime  $O(V)$

# The Program Structure Tree

**Developed** 1994 / Keshav Pingali, Richard Johnson, David Pearson

**Generality** Any control flow

**Precision** All Simple Regions

**Implementation Afford** Middle

**Prerequisite** no

**Runtime**  $O(V + E)$

# The Refined Program Structure Tree

**Developed** January 2009 / Jussi Vanhatalo, Hagen Volzer, Jana Koehler

**Generality** Any control flow

**Precision** All Refined Regions

**Implementation** Afford Middle

**Prerequisite** Triconnected components

**Runtime**  $O(V + E)$

# Dominance Tree Based RPST

Developed Winter 2009 / Tobias Groß

Generality Any control flow

Precision All Refined Regions

Implementation Afford Middle

Prerequisite (Post)Dominance trees.

Runtime  $O((V + E)^2)$  or better.

# Comparison

Analysis	<b>PST</b>	<b>RPST</b>	<b>DRPST</b>
Applicable	All CFGs	All CFGs	All CFGs
Coverage	Simple Regions	Refined Regions	Refined Regions
Runtime	$O(V + E)$	$O(V + E)$	$O((V + E)^2)$ or better
Prerequisites	None	Triconnected components	(Post)Dominance trees
Representation	2 edges	All edges in region	2 basic blocks
BB in Region		extra mapping required	based on dominance info

Table: Comparison of different region detection algorithms

# Outline

## 1 Introduction

## 2 Components

- Control Flow Graph
- Region
- Refined Region
- Program Structure Tree

## 3 Different algorithms

- Regions in Structured Programs
- The Program Structure Tree
- The Refined Program Structure Tree
- Dominance Tree based RPST

## 4 Region detection in static program analysis

- Convert an CFG to an CFA

# Convert an CFG to an CFA

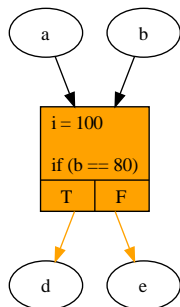


Figure: CFG

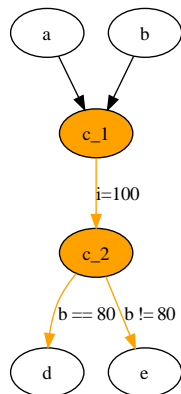


Figure: Resulting CFA

The End

Questions?

## Proportion Simple/Refined Regions

Program	Simple	Refined	Simple / Refined
ac.s	65	77	0.84
aermod.s	1069	5262	0.20
air.s	255	486	0.52
capacita.s	102	232	0.44
channel.s	56	69	0.81
doduc.s	241	702	0.34
fatigue.s	44	133	0.33
gas_dyn.s	76	219	0.35
induct.s	66	239	0.28
linpk.s	44	99	0.44
mdbx.s	84	272	0.31
nf.s	57	122	0.47
protein.s	93	313	0.30
rnflow.s	291	786	0.37
test_fpu.s	242	650	0.37
tfft.s	24	58	0.41
Sum	2809	9719	0.29

Table: Regions in the polyhedron.com benchmark

# Bibliography I



Thomas Ball.

What's in a region?: or computing control dependence regions in near-linear time for reducible control flow.

*ACM Lett. Program. Lang. Syst.*, 2(1-4):1–16, 1993.



Carsten Gutwenger and Petra Mutzel.

A linear time implementation of spqr-trees.

In *GD '00: Proceedings of the 8th International Symposium on Graph Drawing*, pages 77–90, London, UK, 2001. Springer-Verlag.



John E. Hopcroft and Robert Endre Tarjan.

Dividing a graph into triconnected components.

*SIAM J. Comput.*, 2(3):135–158, 1973.



Richard Johnson, David Pearson, and Keshav Pingali.

The program structure tree: Computing control regions in linear time.  
pages 171–185, 1994.

# Bibliography II



Jussi Vanhatalo, Hagen Völzer, and Jana Koehler.

The refined process structure tree.

In *BPM '08: Proceedings of the 6th International Conference on Business Process Management*, pages 100–115, Berlin, Heidelberg, 2008. Springer-Verlag.