

PAG/WWW

Jörg Herter

2013/01/08

... a YACC for program analysis?

- PAG supports instances of monotone frameworks
- Input: concise specifications (of program analyses)
- Output: ANSI C code (of a program analyzer)
- Advantages:
 - 1 rapid implementation
 - 2 integrated debugging facilities
 - 3 short specification

A web interface to PAG.

`http://www.program-analysis.com`

Main differences to PAG:

- Restricted features
- Simplified specification language
- Fixed input language

WHILE: `http://www.program-analysis.com/while.html`

PAG/WWW

- 1 for WHILE programs only
- 2 restricted specification languages
- 3 restricted syntax
- 4 intended for educational purposes
- 5 restricted to a certain iteration algorithm

PAG

- 1 full system not bound to a specific language
- 2 additional specification features
- 3 more complicated syntax
- 4 intended for industrial and for research purposes
- 5 different interprocedural iteration algorithms

Reaching definitions

"Which variable definitions/assignments reach which program points."

Want to compute for each program point u the set:

$\{(v, p) \mid v \in \mathcal{VARS} \wedge p \in \mathcal{LOCS} \wedge \text{definition of } v \text{ at } p \text{ reaches } u\}$

Reaching definitions

Lattice (where the analysis results are from):

$$\{(v, p) \mid v \in \mathcal{VARS} \wedge p \in \mathcal{LOCS}\} = \mathcal{VARS} \times \mathcal{LOCS}$$

Least upper bound (how we combine information from diff. paths):

U

Edge effects (how information changes when traversing a cfg-edge):

???

Reaching definitions (textual results)

`http://www.program-analysis.com`

How to interpret the textual results:

- Labels program automatically
- Shows entry and exit information
- Procedure parameters are underlined

Reaching definitions (graphical results)

<http://www.program-analysis.com>

How to interpret the graphical results:

- A picture for each computation step
- Exit information at outgoing edges
- Entry information not displayed
- Color legend:
 - ▶ Red: information is about to be changed
 - ▶ Blue: nodes in the worklist
- Node labels: numbered elementary statements

Different parts of an analysis specification:

- 1 TYPE: define the analysis lattice
- 2 PROBLEM: define analysis parameters
- 3 TRANSFER: define the transfer functions
- 4 SUPPORT: define additional functions

Specification in a specialized functional language FULA (ML like)

http://www.program-analysis.com/fula_grammar.html

Predefined datatypes:

<code>snum</code>	signed integer
<code>bool</code>	boolean
<code>str</code>	string
<code>Label</code>	program label
<code>Var</code>	program variable
<code>Proc</code>	program procedure
<code>Expression</code>	non-trivial program expression

Lattice construction:

<code>set (<ld>)</code>	Set over <ld>
<code>list (<ld>)</code>	List over <ld>; NOT a lattice!
<code><ld1> * <ld2></code>	Tuple space
<code><ld1> -> <ld2></code>	Function space
<code>flat (<ld>)</code>	Flat lattice
<code>lift (<ld>)</code>	Lifted lattice

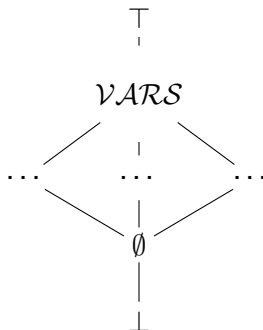
Predefined sets:

<code>LabelSet</code>	<code>set(Label)</code>
<code>VarSet</code>	<code>set(Var)</code>
<code>ProcSet</code>	<code>set(Proc)</code>
<code>ExpressionSet</code>	<code>set(Expression)</code>
<code>ExpressionList</code>	<code>list(Expression)</code>

- 1 In PAG, the worklist is initialized *only* with the start node.
- 2 Algorithm stops when worklist is empty, new nodes are added when a node value changes
- 3 The programmer must ensure that each node of interest is visited at least once.

⇒ Add an additional, artificial bottom element if the bottom element of the lattice has a meaningful value

Remarks: Fixed-point Iteration cont'd



Live Variables Analysis

"Which variables are live at a program point p , i.e., may be read before written on a program path from p to program exit."

Want to compute for each program point p the set

$$\{v \mid v \text{ is live at } p\}$$

Lattice: $2^{\mathcal{VARS}}$, where \mathcal{VARS} is the set of all program variables

Least upper bound: \cup ; MOP: $\mathcal{L}[u]^* = \cup\{\llbracket \pi \rrbracket \# \emptyset \mid \pi : u \rightarrow^* \text{end}\}$

Edge effects: ???

PROBLEM Specification

direction	forward or backward
carrier	the analysis lattice
init	the initial value
init_start	value for the extremal node
combine	combination function

Live variables (problem specification)

`http://www.program-analysis.com`

In the TRANSFER section,

- define the exit value in terms of the entry value(@), or vice versa for backward problems
- give a definition by cases for the WHILE statements
- optional matching of the edge type

Live variables (transfer section specification)

`http://www.program-analysis.com`

In the SUPPORT section,

- define additional functions
- where each function needs a type declaration
- and definition by cases possible

Live variables (support section specification and analysis demo)

`http://www.program-analysis.com`

Available Expressions Analysis and Constant Propagation

`http://www.program-analysis.com`