Using Dependence Graphs for Slicing Functional Programs

Dr. Vadim Zaytsev aka @grammarware

IFL 2015

Using Dependence Graphs for Slicing Functional Programs

Dr. Vadim Zaytsev aka @grammarware

IFL 2015

Slicing

```
read(text);
read(n);
lines = 1;
chars = 1;
subtext = "";
c = getChar(text);
while (c != ‘\eof’)
      if (c == ‘\n’)
      then lines = lines + 1;
           chars = chars + 1;
      else chars = chars + 1;
           if (n != 0)
           then subtext = subtext ++ c;
                n = n – 1;
      c = getChar(text);
write(lines);
write(chars);
write(subtext);
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Slicing

```
read(text);
read(n);
lines = 1;
chars = 1;
subtext = "";
c = getChar(text);
while (c != ‘\eof’)
      if (c == ‘\n’)
      then lines = lines + 1;
           chars = chars + 1;
      else chars = chars + 1;
           if (n != 0)
           then subtext = subtext ++ c;
                n = n – 1;
      c = getChar(text);
write(lines);
write(chars);
write(subtext);
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Slicing

```
read(text);
read(n);
lines = 1;
chars = 1;
subtext = "";
c = getChar(text);
while (c != ‘\eof’)
      if (c == ‘\n’)
      then lines = lines + 1;
           chars = chars + 1;
      else chars = chars + 1;
           if (n != 0)
           then subtext = subtext ++ c;
                n = n – 1;
      c = getChar(text);
write(lines);
write(chars);
write(subtext);
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Slicing

```
module IFL15
where

main = do
   text <- readFile "…"
   n <- readLn :: IO Int
   print $ 1 + (length . filter (=='\n')) text
   print $ length text
   print $ take n (filter (/='\n') text)
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Slicing

```
module IFL15
where

main = do
   text <- readFile "…"
   n <- readLn :: IO Int
   print $ 1 + (length . filter (=='\n')) text
   print $ length text
   print $ take n (filter (/='\n') text)
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Slicing

```
module IFL15
where

main = do
   text <- readFile "…"
   n <- readLn :: IO Int
   return $ triple text n

triple :: String -> Int -> (Int, Int, String)
triple text n = (lines, chars, subtext) where
   lines = 1 + (length . filter (=='\n')) text
   chars = length text
   subtext = take n (filter (/='\n') text)
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Slicing

```
module IFL15
where

main = do
   text <- readFile "…"
   n <- readLn :: IO Int
   return $ triple text n

triple :: String -> Int -> (Int, Int, String)
triple text n = (lines, chars, subtext) where
   lines = 1 + (length . filter (=='\n')) text
   chars = length text
   subtext = take n (filter (/='\n') text)
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Slicing

```
module IFL15
where

main = do
   text <- readFile "…"
   n <- readLn :: IO Int
   return $ triple text n

triple :: String -> Int -> (Int, Int, String)
triple text n = (lines, chars, subtext) where
   lines = 1 + (length . filter (=='\n')) text
   chars = length text
   subtext = take n (filter (/='\n') text)
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Slicing

```
module IFL15
where

main = do
   text <- readFile "…"
   n <- readLn :: IO Int
   return $ triple text n

triple :: String -> Int -> (Int, Int, String)
triple text n = (lines, chars, subtext) where
   lines = 1 + (length . filter (=='\n')) text
   chars = length text
   subtext = take n (filter (/='\n') text)
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Related concepts

- ✓ Forward/backward slicing
- ✓ Dynamic/conditioned slicing
  - ✓ constraints on input
- ✓ Chopping
  - ✓ discover connection between I & O
- ✓ Amorphous slicing
  - ✓ . . .

Amorphous Slicing

```
f :: Int -> Int
f cx = add cx 1
```

```
inc :: Int -> Int
inc a = add a 1

add :: Int -> Int -> Int
add a b = a + b
```

M. Harman, S. Danicic, Amorphous program slicing. IWPC 1997.

Amorphous Slicing

```
f :: Int -> Int
f cx = add cx 1

inc :: Int -> Int
inc a = add a 1

add :: Int -> Int -> Int
add a b = a + b
```

M. Harman, S. Danicic, Amorphous program slicing. IWPC 1997.

Amorphous Slicing

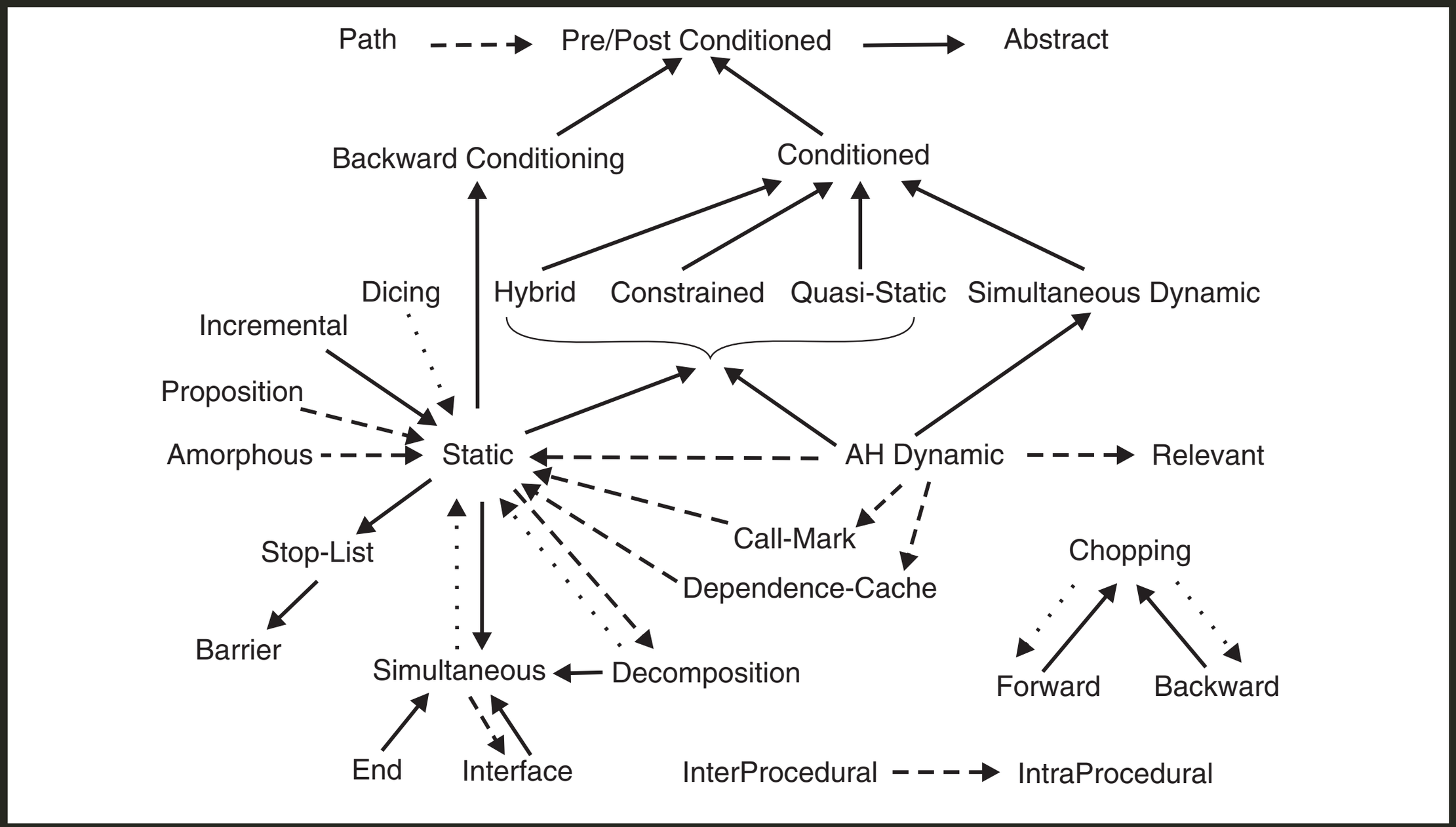
```
f :: Int -> Int
f cx = cx + 1
```

```
inc :: Int -> Int
inc a = add a 1

add :: Int -> Int -> Int
add a b = a + b
```

M. Harman, S. Danicic, Amorphous program slicing. IWPC 1997.

Forms of Slicing



J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

Uses for slicing

- ✓ Debugging
  - ✓ cf. Weiser CACM 1982
- ✓ Cohesion measurement
  - ✓ cf. Ott&Bieman IST 1998
- ✓ Comprehension
  - ✓ cf. De Lucia&Fasolino&Munro IWPC 1996
- ✓ Maintenance
  - ✓ e.g. reuse
- ✓ Re-engineering
  - ✓ e.g. clone detection

http://www0.cs.ucl.ac.uk/staff/mharman/sf.html

Goal: Library

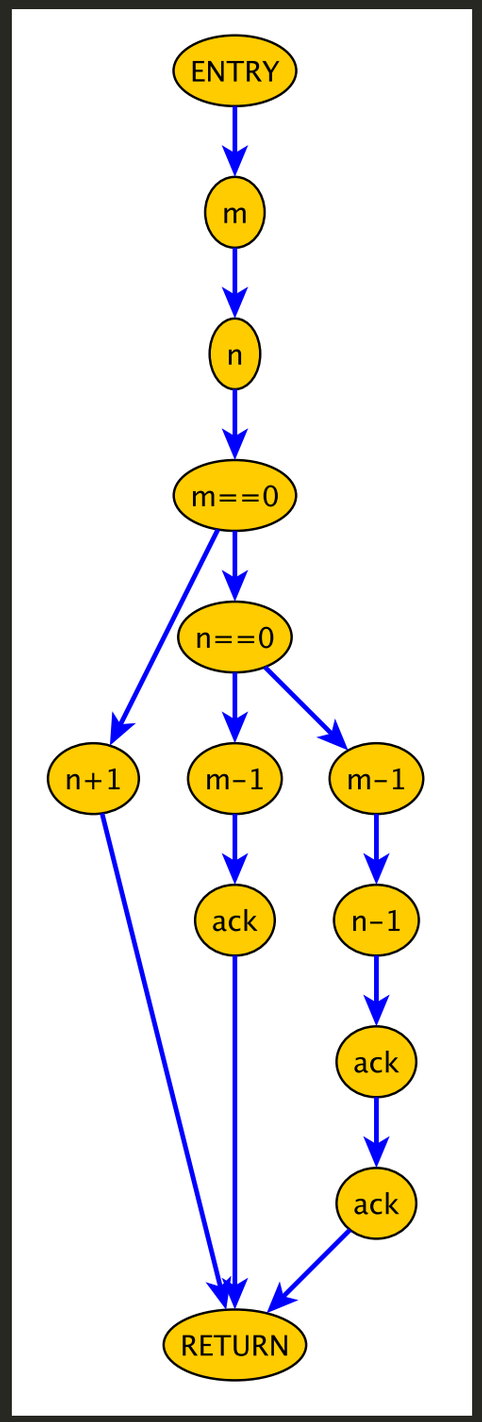
http://github.io/grammarware/pdg

Using Dependence Graphs for Slicing Functional Programs

Dr. Vadim Zaytsev aka @grammarware

IFL 2015

Control Dependence

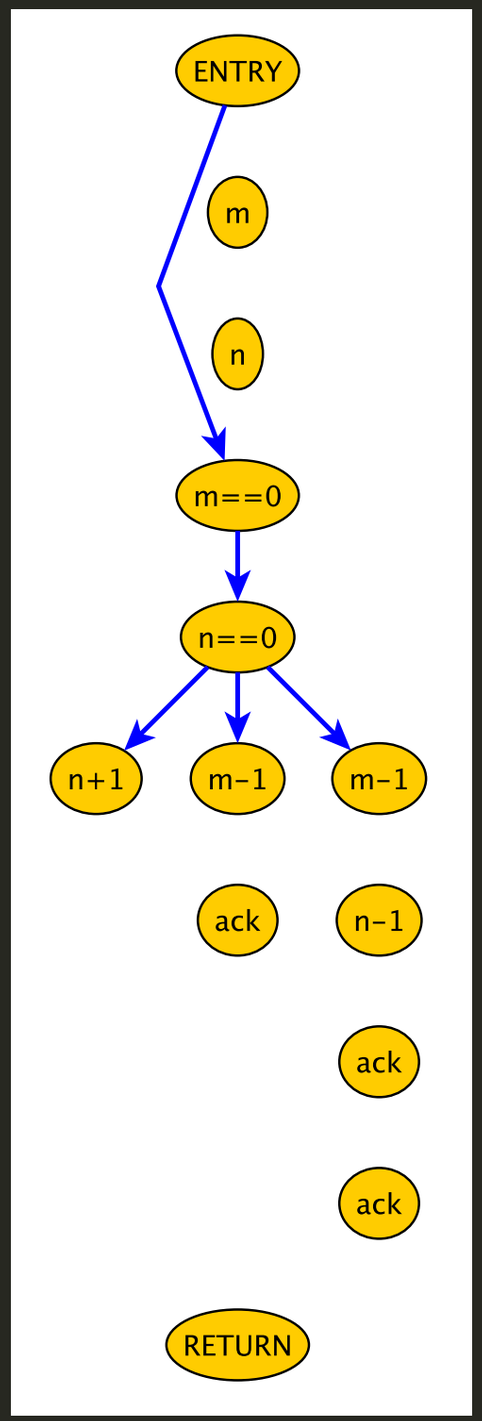



```
ack :: Int -> Int -> Int
ack 0 n = n + 1
ack m 0 = ack (m-1) 1
ack m n = ack (m-1) (ack m (n-1))
```

- ✓ Control flow
  - ✓ execution path
- ✓ Exclude domination
  - ✓ inevitable

J. de Bakker, E. de Vink, Control Flow Semantics, MIT Press, 1996.
O. Shivers. Control-Flow Analysis of Higher-Order Languages. PhD, 1991.

Control Dependence

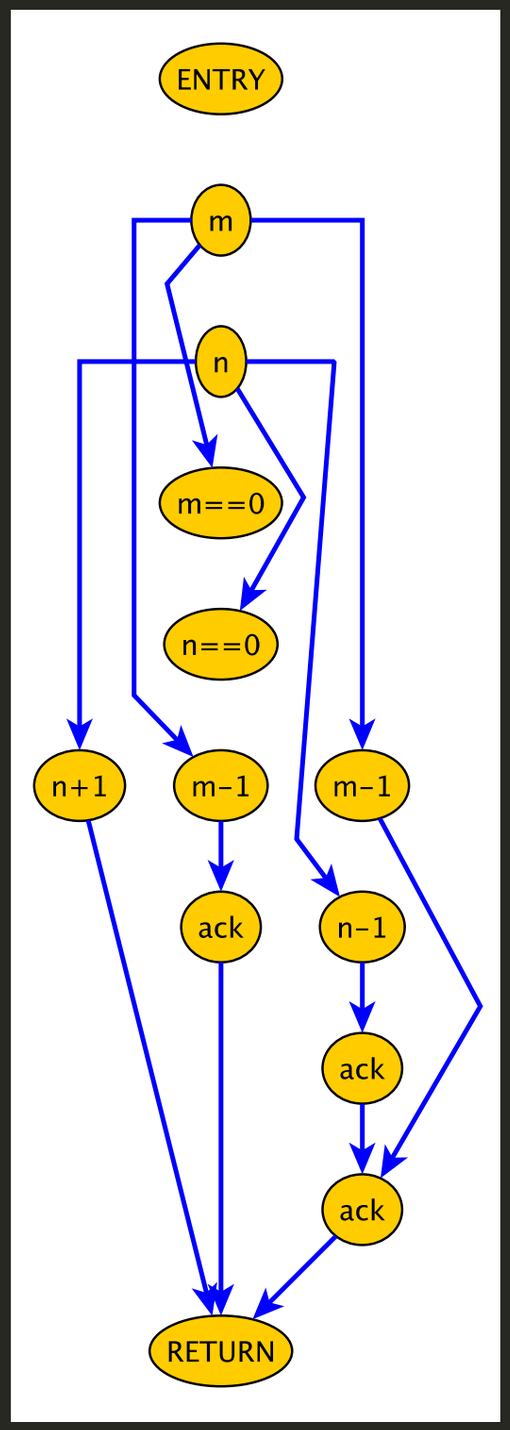



```
ack :: Int -> Int -> Int
ack 0 n = n + 1
ack m 0 = ack (m-1) 1
ack m n = ack (m-1) (ack m (n-1))
```

- ✓ Control flow
  - ✓ execution path
- ✓ Exclude domination
  - ✓ inevitable

J. de Bakker, E. de Vink, Control Flow Semantics, MIT Press, 1996.
O. Shivers. Control-Flow Analysis of Higher-Order Languages. PhD, 1991.

Data Dependence

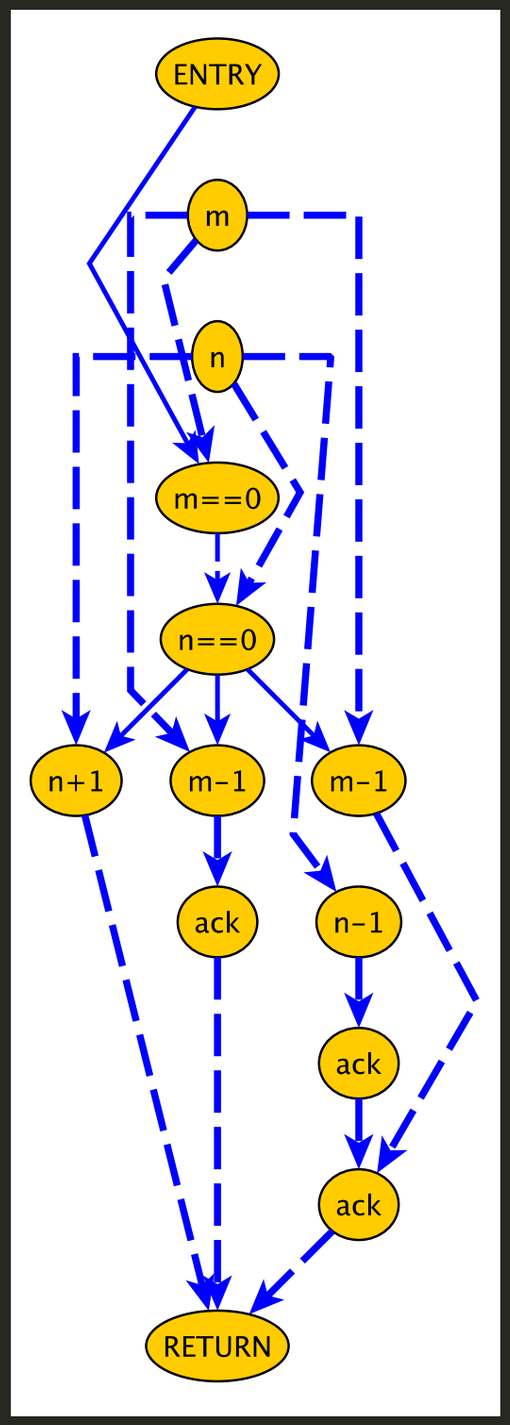



```
ack :: Int -> Int -> Int
ack 0 n = n + 1
ack m 0 = ack (m-1) 1
ack m n = ack (m-1) (ack m (n-1))
```

- ✓ “Define” locations
  - ✓ change
- ✓ “Use” locations
  - ✓ access

L. Lövei. Automated module interface upgrade. Erlang 2009.

Program Dependence

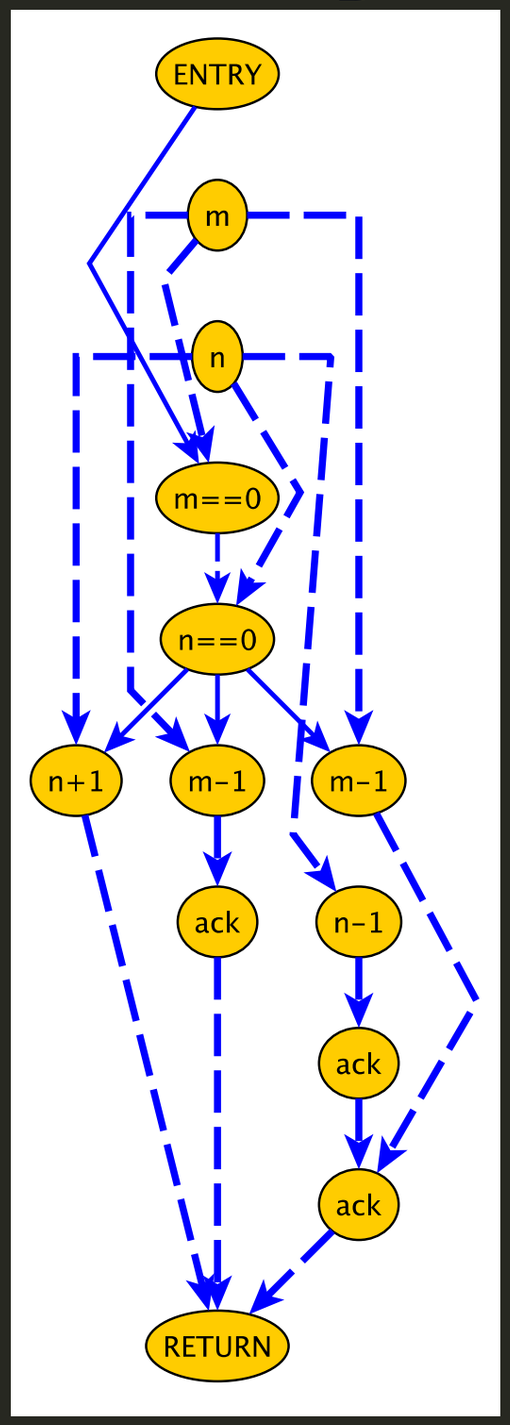



```
ack :: Int -> Int -> Int
ack 0 n = n + 1
ack m 0 = ack (m-1) 1
ack m n = ack (m-1) (ack m (n-1))
```

- ✓ Merge
  - ✓ CDG
  - ✓ DDG
- ✓ into one multigraph

K.J. Ottenstein, L.M. Ottenstein. The Program Dependence Graph in a Software Development Environment. PSDE 1984.

System Dependence




```
ack :: Int -> Int -> Int
ack 0 n = n + 1
ack m 0 = ack (m-1) 1
ack m n = ack (m-1) (ack m (n-1))
```

- ✓ Same as PDG
- ✓ interprocedural support

S. Horwitz, T.W. Reps, D. Binkley. Interprocedural Slicing Using Dependence Graphs. ToPLaS, 1990.

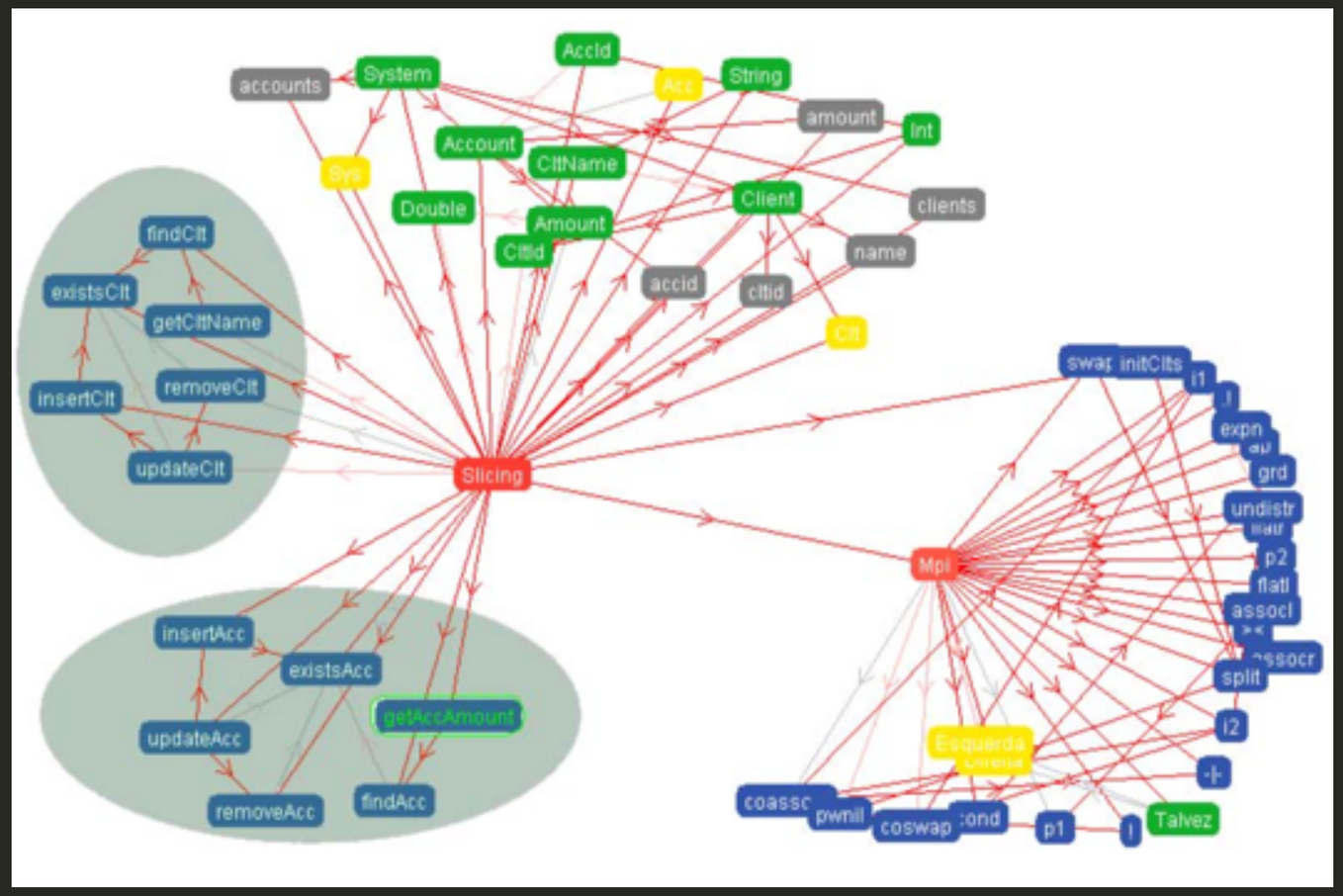
Variations

- ✓ Dynamic + Unified
  - ✓ runtime info
- ✓ Probabilistic + Weighted
  - ✓ trace coverage
- ✓ Values
  - ✓ lazy eval
- ✓ OO
  - ✓ methods, fields, classes

H. Agrawal, J. R. Horgan. Dynamic Program Slicing. PLDI 1990.
I. Forgács, Á. Hajnal, É. Takács. Regression Slicing and Its Use in Regression Testing. COMPSAC 1998.
D. Weise, R.F. Crew, M.D. Ernst, B. Steensgaard. Value Dependence Graphs: Representation without Taxation. POPL 1994.
J. Zhao. Applying Program Dependence Analysis To Java Software. SEDS 1998.
D. Liang, M. J. Harrold. Slicing Objects Using System Dependence Graphs. ICSM 1998.
N. Walkinshaw, M. Roper, M. Wood. The Java System Dependence Graph. SCAM 2003.

F-Statements Dep.

- ✓ m, f, dt, c, d
- ✓ “Functional Statements”
- ✓ Very high level
  - ✓ architectural



N.F. Rodrigues, L.S. Barbosa. Component Identification through Program Slicing. FACS 2005.

Online Haskell Slicer - Beta 2.2 - 2007/02/01

Slicer

Slice Type : Forward Slicing

Node Type : Module

Node Name :

Slice

Utilities

Find Node :

Ok

Back

Functional Dependency Graph

Error. Click for details

http://labdotnet.di.uminho.pt/HaSlicer/HaSlicer.aspx

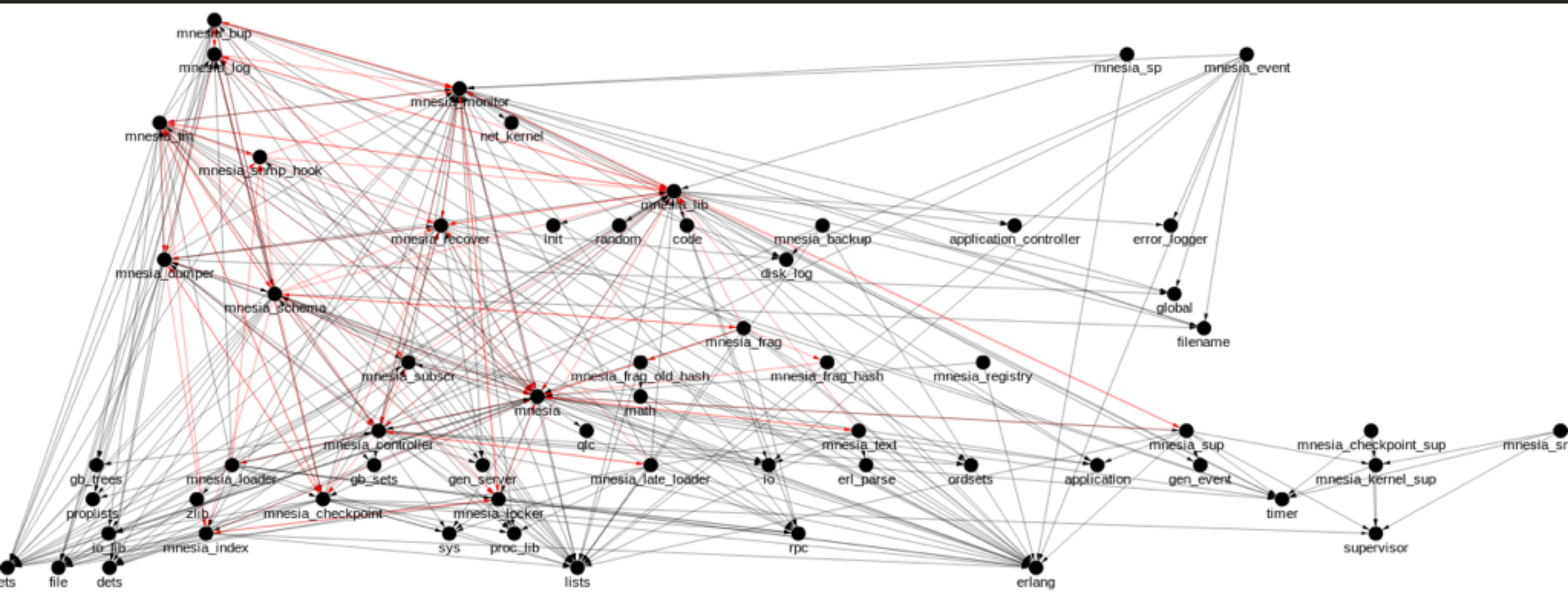
Show Entire Graph

Behaviour Dependence

- ✓ Edges
  - ✓ data, control, behaviour
- ✓ Can handle
  - ✓ pattern-driven dispatch
  - ✓ expression decomposition
- ✓ But
  - ✓ extremely Erlang-specific

M. Tóth, I. Bozó, Z. Horváth, L. Lövei, M. Tejfel, T. Kozsik.
Impact Analysis of Erlang Programs Using Behaviour Dependency Graphs. CEFP 2009.

RefactorErl



http://plc.inf.elte.hu/erlang/

Term Dependence

- ✓ “Program positions”
- ✓ Left and right hand sides
- ✓ S-edges inside terms
  - ✓ structural
- ✓ C-edges link uses & defs
  - ✓ control
- ✓ No higher order

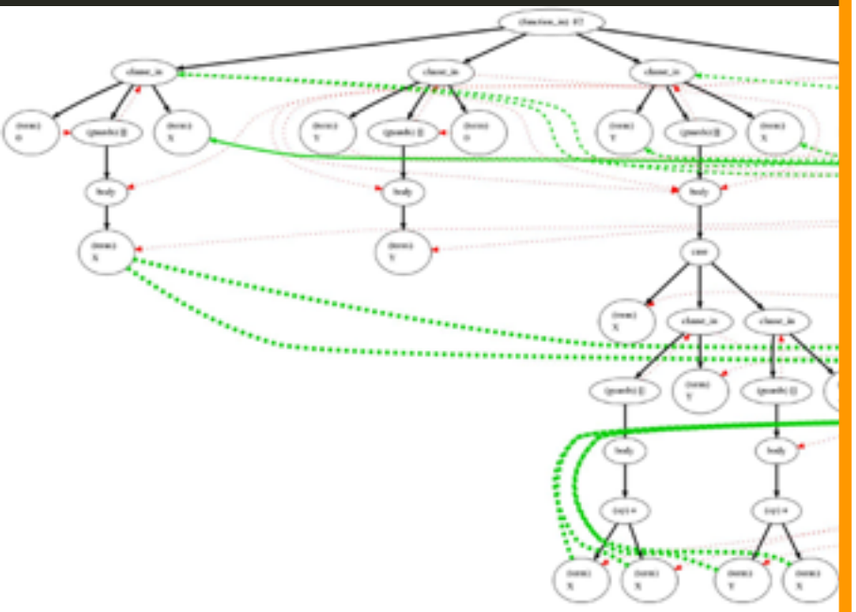
D. Cheda, J. Silva, G. Vidal. Static Slicing of Rewrite Systems. WFLP 2006.

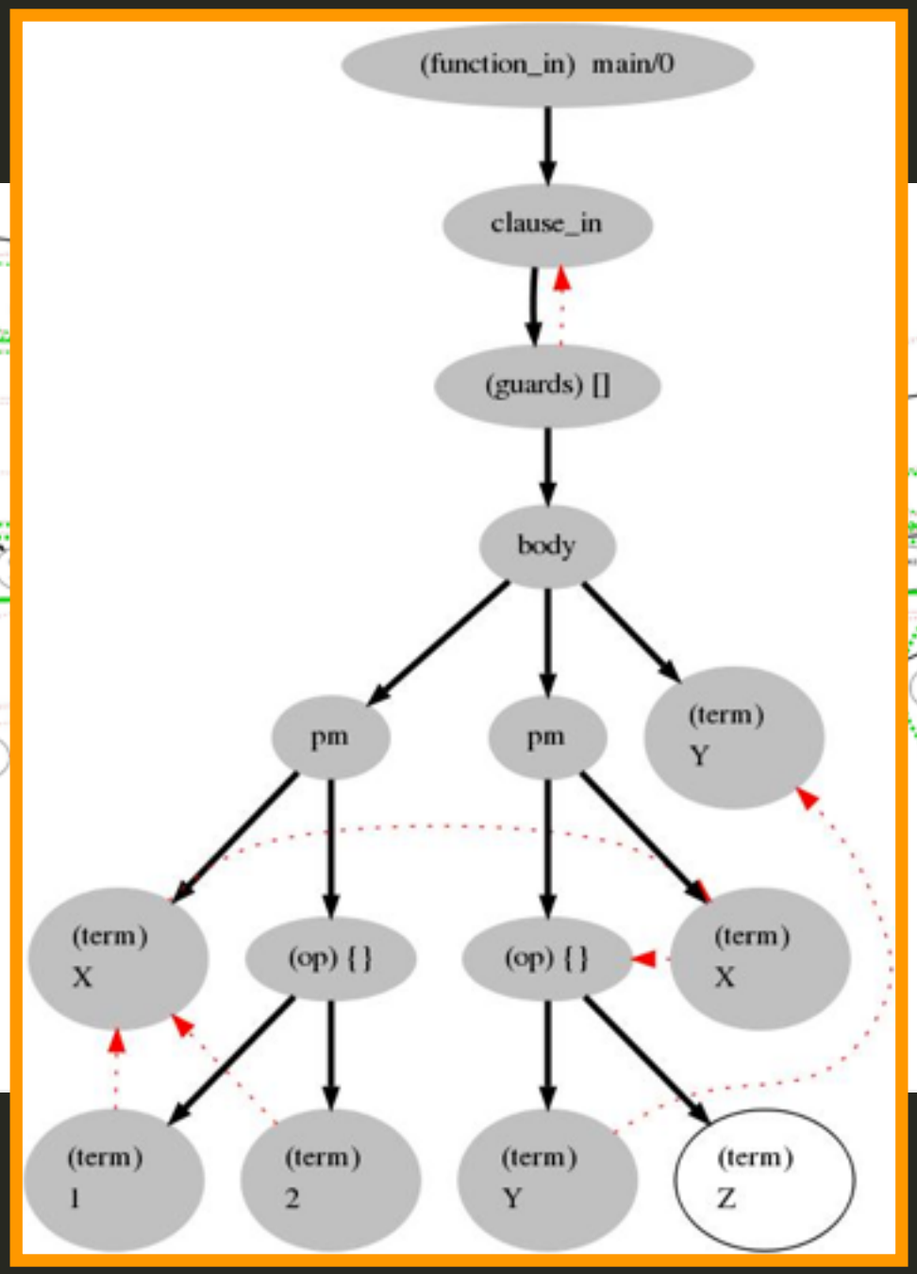
“Erlang Dependence”

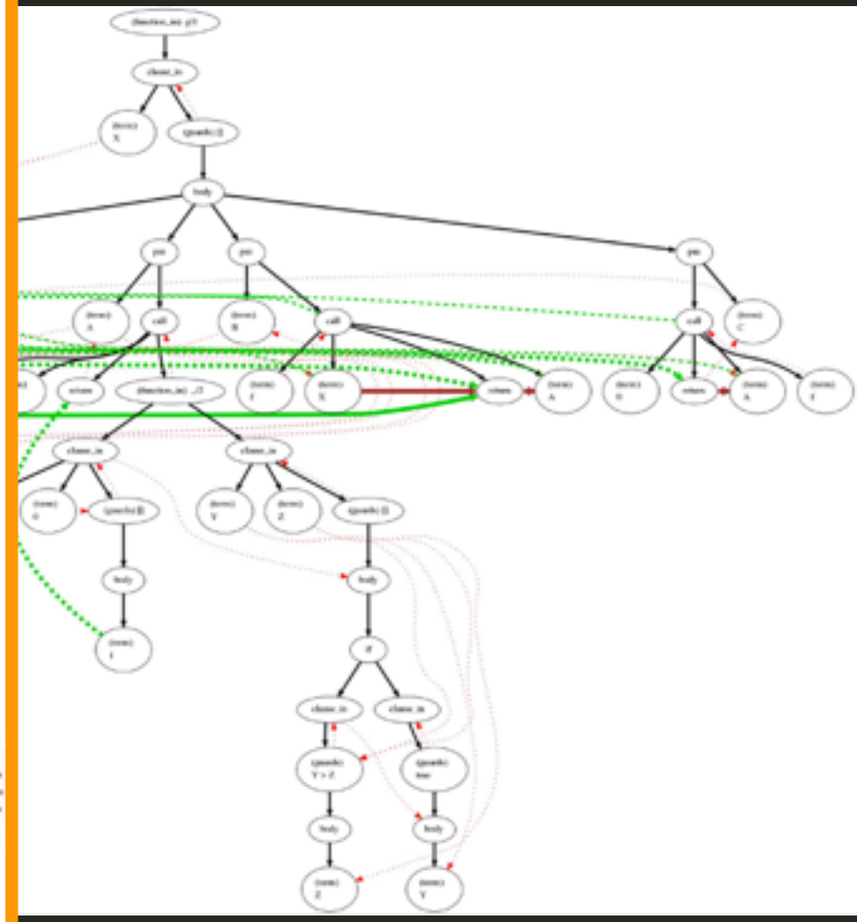
- ✓ (almost) all of the above
- ✓ Edges
  - ✓ control, input, output, data, summary
- ✓ no deep decomposition
- ✓ no concurrency

J. Silva, S. Tamarit, C. Tomás. System Dependence Graphs in Sequential Erlang. FASE 2012.

SlicErl







http://kaz.dsic.upv.es/slicErlang.html

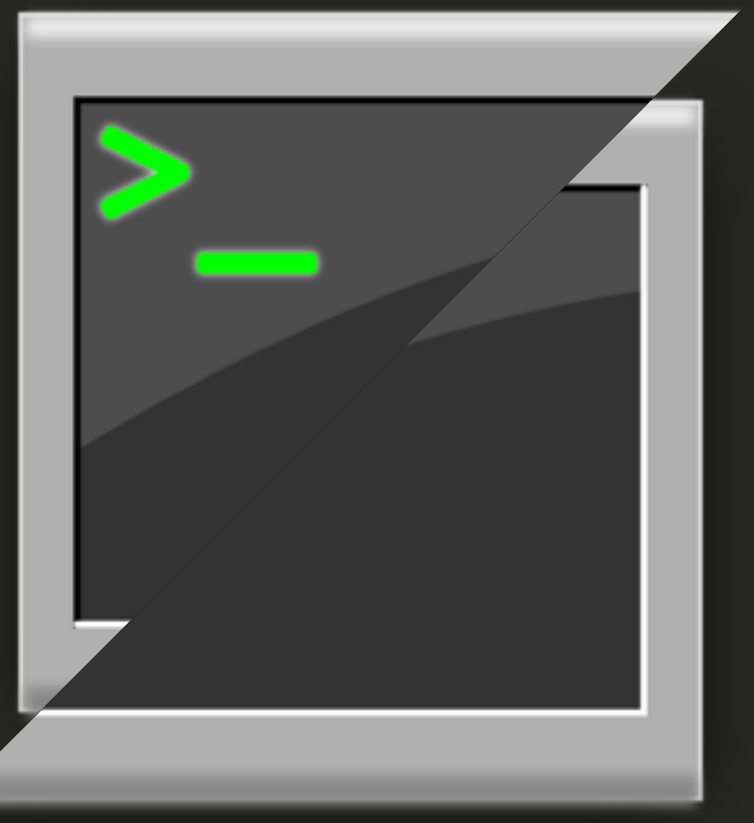
W.I.P.

- ✓ CDG/DDG work
  - ✓ not a serious challenge
- ✓ PDG works

  Lulu Zhang. Implementing a PDG Library in Rascal. April–September, UvA, 2014.
- ✓ SDG works

  René Bulsing. Detecting Refactored Clones with Rascal. April–August, UvA, 2015.
- ✓ next?

Questions/Advice?



- ✓ @grammarware
- ✓ http://grammarware.net
- ✓ http://grammarware.github.io
- ✓ http://twitter.com/grammarware
- ✓ …

Feedback

- ✓ slice modules
  - ✓ comprehension: see where to start
  - ✓ optimisation before deployment
- ✓ slicing across modules
  - ✓ and versions of modules
- ✓ Clemens: general program trafo ± symbolic computation ± slicing
- ✓ Rinus: slicing modules for doc
- ✓ чувак в очках — оптимизация
- ✓ Philip: version slicing
- ✓ чувак из Брюсселя: слайс яваскрипта в VUB

SLEBoK — BibSLEIGH

bibtex.github.io/person/Vadim_Zaytsev.html

@BIBSLEIGH

corpus
tags
bundles
people
edit!

Person: Vadim Zaytsev

DBLP: Zaytsev:Vadim

Facilitated 7 volumes:

PubCh SciCo PrCo PrCo PubCh PubCh WebCh

Contributed to:

2014 2014 2013 2013 2012 2012 2011 2010 2009 2009 2011 2009

Wrote 13 papers:

CSMR-WCRE-2014-BaggeZ
International workshop on open and original problems in software language engineering (AHB, VZ), p. 478.

CSMR-WCRE-2014-Zaytsev T T
Formal foundations for semi-parsing (VZ), pp. 313–317.

MODELS-2014-ZaytsevB T T
Parsing in a Broad Sense (VZ, AHB), pp. 50–67.

SLE-2013-Zaytsev
Micropatterns in Grammars (VZ), pp. 117–136.

WCRE-2013-BaggeZ
Workshop on open and original problems in software language engineering (AHB, VZ), pp. 493–494.

LDTA-2012-Zaytsev T
Notation-parametric grammar recovery (VZ), p. 9.

SAC-2012-Zaytsev T
BNF was here: what have we done about the unnecessary diversity of notation for syntactic definitions (VZ), pp. 1910–1915.

SLE-2011-FischerLZ T T T
Comparison of Context-Free Grammars Based on Parsing Generated Test Data (BF, RL, VZ), pp. 324–343.

SLE-2010-ZaytsevL T
A Unified Format for Language Documents (VZ, RL), pp. 206–225.

GTTSE-2009-Zaytsev T
Language Convergence Infrastructure (VZ), pp. 481–497.

IFM-2009-LammelZ
An Introduction to Grammar Convergence (RL, VZ), pp. 246–260.

SCAM-2009-J-LammelZ11 T T
Recovering grammar relationships for the Java Language Specification (RL, VZ), pp. 333–378.

Travelled to:

1 ⁄ Belgium
1 ⁄ Canada
1 ⁄ Estonia
2 ⁄ Germany
1 ⁄ Italy
2 ⁄ Portugal
1 ⁄ Spain
1 ⁄ The Netherlands
1 ⁄ USA

Collaborated with:

Ø R. Lämmel
A.H. Bagge
B. Fischer