Proving the Correctness of Algorithmic Debugging for Functional Programs

Yong Luo

University of Kent, UK

4th October 2006
Aims and Outline

Aims

- Model the Haskell tracer Hat
- Provide theoretical foundation
- Guide implementation

Outline

- Augmented Redex Trail (ART). What? Why?
- Evaluation Dependency Tree (EDT).
- Replacing unevaluated parts. How?
- Correctness of algorithmic debugging
- Proofs
- Discussion
An example

The program:

\[ \text{doubleneg } x = \text{id} \ (\text{not } x) \]

The starting term:

\[ \text{main} = \text{doubleneg} \ (\text{not } \text{True}) \]
An Augmented Redex Trail (ART) is a graph

- Starts from “main”
- a general function to add new graphs
- Sharing
Formalising ART (2)

- Independence from evaluation order
- Node naming scheme
  - not distinguish isomorphic graphs
  - given parent node implicitly
Algorithmic Debugging

An Evaluation Dependency Tree (EDT) is generated from an ART.

Example

![Diagram of an EDT](image)

- doubleneg
- Truenot
- False
- id
- not
- True
- t
- tl
- tt ttt
- tr
- trl trr
- trt
- ttl ttr ttrt
- ttrl

- doubleneg False = True
- main = True
- not True = False
- yes
- faulty node
- doubleneg False = True
- no
- id True = True
- yes
Condition 1: The head of the node must be a function.
Condition 2: No computation at the node.
Condition 1: The head of the node must be a function.
Condition 2: No computation at the node.
Condition 3: Must not be the LHS of an application.
The original EDT:

The new EDT:
Meaning of an equation

**ART and EDT:**

If the user says

▶ \((g \_ = c2 \_)\) is intended semantics, s/he means

\[ \forall x \exists y. (g \ x = c2 \ y) \]

▶ \((g \_ = c2 \_)\) is NOT intended semantics, s/he means

\[ \exists x \forall y. (g \ x \neq c2 \ y) \]
Correctness of Algorithmic Debugging

Faulty nodes

\[ f \ldots = R \]
\[ g_1 \ldots = R_1 \]
\[ g_2 \ldots = R_2 \]
\[ \ldots \ldots \]
\[ g_n \ldots = R_n \]

Yes

Yes

Yes

No

Correctness

- If the equation of a faulty node is \( f \ldots = R \), then the definition of the function \( f \) in the program is faulty
The difficulties

- suitable reduction principle
- more general induction hypothesis
- Dealing with $\forall$ quantifier.

What have been proved:

- $fa_1...a_n \rightarrow_1 N$. i.e. $fa_1...a_n$ computes to $N$ in a single step.
- But $N$ is not the intended semantics of $fa_1...a_n$. 

Proofs

No details here.
Discussion

- Add local rewriting rules