Weiser's Algorithm Computes Minimal Path-Faithful Slices of Function-linear, Free Program Schemas

Mike Laurence

Department of Computing, Goldsmiths College, London, UK

July 11, 2008

Weiser's Algorithm Computes Minimal Path-Faithful Slices of Function-linear, Free Program Schemas

Mike Laurence

Department of Computing, Goldsmiths College, London, UK
m.laurence@gold.ac.uk

A *schema* is like a program except that real functions and real predicates are replaced by symbols referring to functions and predicates.

Introduction to program schemas

A *schema* is like a program except that real functions and real predicates are replaced by symbols referring to functions and predicates.

A schema thus represents an entire class of programs, depending on how the function and predicate symbols are interpreted.

$$x := a();$$
$$y := b();$$
$$\mathit{while}\ p(y)$$
$$\{$$
$$y := f(y);$$
$$x := g(x, y);$$
$$\}$$

Here is a schema.

$$x := 0;$$
$$y := 0;$$
$$\textit{while}\,(y < 100)$$
$$\{$$
$$y := y + 1;$$
$$x := x + y;$$
$$\}$$

Here is one of the programs that it represents.

Applications of schemas

- Program Transformation

Applications of schemas

- Program Transformation
- Program Comprehension

Applications of schemas

- Program Transformation
- Program Comprehension
- Program Slicing

Slices of schemas

Definition A schema T is a *slice* of a schema S if T is obtained from S by deleting statements from S;

Definition A schema T is a *slice* of a schema S if T is obtained from S by deleting statements from S;

formally, a slice is defined recursively by the following rules;

skip rule;

skip

is a slice of every schema.

Slices of sequences of schemas

- $S_1 skip$ and $skip S_2$ are slices of $S_1 S_2$;

Slices of sequences of schemas

- $S_1 skip$ and $skip S_2$ are slices of $S_1 S_2$;
- $S_1 skip S_3$ is a slice of $S_1 S_2 S_3$.

Slices of *while* statements

If S' is a slice of S then

$$\textit{while } p(v) \textit{ do } S'$$

is a slice of

$$\textit{while } p(v) \textit{ do } S.$$

Slices of *if* statements

Suppose

- S_1' is a slice of S_1, and
- S_2' is a slice of S_2,

then

$$\textit{if } q(u) \textit{ then } S_1' \textit{ else } S_2'$$

is a slice of

$$\textit{if } q(u) \textit{ then } S_1 \textit{ else } S_2.$$

Definition Given a schema S, a slice T of S and variable v;

- T is a v-slice of S if T preserves termination and the final value of v.

Two semantic slicing criteria for schemas

Definition Given a schema S, a slice T of S and variable v;

- T is a v-slice of S if T preserves termination and the final value of v.
- T is a v-path-faithful slice of S if T preserves termination and the final value of v *and also* the executed path through S.

Definition Given a schema S, and variable v, Weiser's slice $\mathcal{W}(S, v)$ is the minimal slice of S which is closed under

- backward data dependence, and

Definition Given a schema S, and variable v, Weiser's slice $\mathcal{W}(S, v)$ is the minimal slice of S which is closed under

- backward data dependence, and
- control dependence.

Definition Given a schema S, and variable v, Weiser's slice $\mathcal{W}(S, v)$ is the minimal slice of S which is closed under

- backward data dependence, and
- control dependence.

Theorem[M R Laurence, PLID 2008]
$\mathcal{W}(S, v)$ is a v-path-faithful slice of S.

An example of Weiser's algorithm for a schema S

$$
\begin{array}{llll}
\textit{while } q(u) & & & \\
& \{ & & \\
& u := k(u); & & \\
& \textit{if } p(w) & \textit{then} & \\
& & & \{ \\
& & & v := g(); \\
& & & w := f(w); \\
& & & \} \\
& & \textit{else} & \textit{skip} \\
& \} & & \\
\end{array}
$$

An example of Weiser's algorithm for a schema S

$$
\begin{array}{l}
\textit{while } q(u) \\
\quad \{ \\
\quad u := k(u); \\
\quad \textit{if } p(w) \quad \textit{then} \\
\qquad \{ \\
\qquad v := g(); \\
\qquad w := f(w); \\
\qquad \} \\
\quad \textit{else} \quad \textit{skip} \\
\quad \}
\end{array}
$$

An example of Weiser's algorithm for a schema S

$$
\begin{array}{l}
\textit{while } q(u) \\
\quad \{ \\
\quad u := k(u); \\
\quad \textit{if } p(w) \quad \textit{then} \\
\qquad \{ \\
\qquad v := g(); \\
\qquad w := f(w); \\
\qquad \} \\
\quad \textit{else} \quad \textit{skip} \\
\quad \}
\end{array}
$$

An example of Weiser's algorithm for a schema S

$$
\begin{array}{l}
\textit{while } \color{red}{q(u)} \\
\qquad \{ \\
\qquad u := k(u); \\
\qquad \textit{if } \color{red}{p(w)} \quad \textit{then} \\
\qquad\qquad \{ \\
\qquad\qquad \color{red}{v := g();} \\
\qquad\qquad \color{red}{w := f(w);} \\
\qquad\qquad \} \\
\qquad \quad \textit{else} \quad \textit{skip} \\
\qquad \}
\end{array}
$$

An example of Weiser's algorithm for a schema S

$$
\begin{array}{l}
\mathit{while}\ q(u) \\
\qquad \{ \\
\qquad u := k(u); \\
\qquad \mathit{if}\ p(w) \quad \mathit{then} \\
\qquad\qquad \{ \\
\qquad\qquad v := g(); \\
\qquad\qquad w := f(w); \\
\qquad\qquad \} \\
\qquad\quad \mathit{else} \quad \mathit{skip} \\
\qquad \}
\end{array}
$$

An example of Weiser's algorithm for a schema S

$$
\begin{array}{l}
\textit{while } q(u) \\
\quad \{ \\
\quad u := k(u); \\
\quad \textit{if } p(w) \quad \textit{then} \\
\qquad \{ \\
\qquad v := g(); \\
\qquad w := f(w); \\
\qquad \} \\
\quad \textit{else} \quad \textit{skip} \\
\quad \}
\end{array}
$$

Thus, $\mathcal{W}(S, v) = S$ in this case;

however,

S has a smaller slice T given by deleting the assignment $w := f(w)$; from S.

T is a v-slice

Thus, $\mathcal{W}(S, v) = S$ in this case;

however,

S has a smaller slice T given by deleting the assignment $w := f(w)$; from S.

T is a v-slice

but not a v-path-faithful slice.

Conclusion from schema S

Weiser's algorithm does not always produce *minimal* v-slices, and

Conclusion from schema S

Weiser's algorithm does not always produce *minimal* v-slices, and

v-slices need not be path-faithful.

Another example of Weiser's algorithm for a schema

Let S' be the schema

$$\textit{while } p(u) \;\; v := g().$$

Another example of Weiser's algorithm for a schema

Let S' be the schema

$$\textit{while}\ p(u)\ \ v := g().$$

Here, $\mathcal{W}(S', v) = S'$, but

Another example of Weiser's algorithm for a schema

Let S' be the schema

$$while\ p(u)\ \ v := g().$$

Here, $\mathcal{W}(S', v) = S'$, but

$$skip$$

is a v-path-faithful slice of S'.

Conclusion;

Weiser's algorithm does not always produce *minimal* path-faithful slices.

Definition A schema is

- *linear* if it has no repeated function *or* predicate symbols;

Definition A schema is

- *linear* if it has no repeated function *or* predicate symbols;
- *function-linear* if it has no repeated function symbols.

Definition A schema S is

- *free* if every path through S is executable for some interpretation of its symbols and some initial state;

Definition A schema S is

- *free* if every path through S is executable for some interpretation of its symbols and some initial state;
- *liberal* if for every executable path ρ through S, there is an interpretation of its symbols and an initial state such that the same expression is not generated more than once along ρ.

Freeness $\not\Rightarrow$ liberality

The schema

$$\begin{aligned} \textit{while}\ p(w)\ \ \{ & \\ & w := f(w); \\ & v := g(); \\ & \} \end{aligned}$$

is free and linear but not liberal.

Liberality $\not\Rightarrow$ freeness

The schema

$$while\ p(w) skip$$

is liberal and linear but not free.

Theorem[M R Laurence, *JLAP* 2005]

Let S be a free, liberal, function-linear schema and let v be a variable.

Then the slice $\mathcal{W}(S, v)$ of S is the minimal v-slice of S.

Theorem[M R Laurence, *PLID* 2008]

Let S, be a free, function-linear schema and let v be a variable.

Then the slice $\mathcal{W}(S, v)$ of S is the minimal path-faithful v-slice of S.