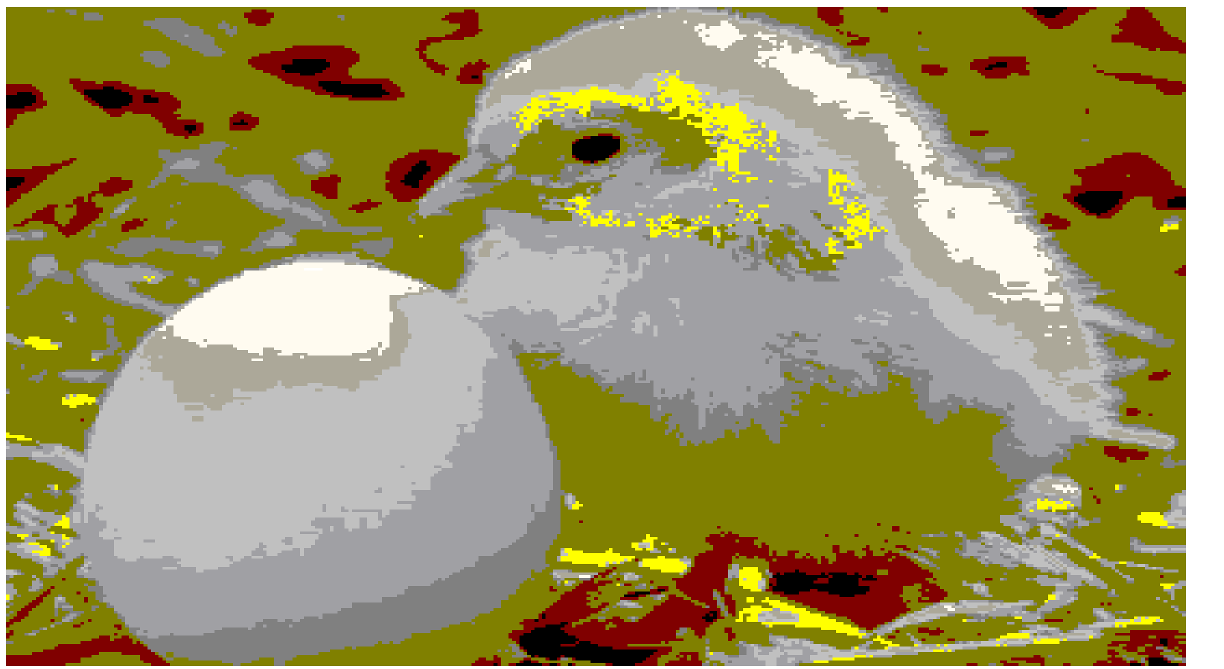
Properties of Slicing Definitions

Martin Ward
martin@gkc.org.uk

Software Technology Research Lab
De Montfort University

Is *S* a slice of *P*



A Simplistic Definition

- Program **S** is a slice of Program **P** taken with respect to Variable **V** if when run on initial state **σ**

1. **P** terminates
2. $\mathcal{M}\,\mathbf{P}\,\boldsymbol{\sigma}\,|_V = \mathcal{M}\,\mathbf{S}\,\boldsymbol{\sigma}\,|_V$

   (**P** and **S** compute the same final value for **V**)

An Example

x := 1

is a slice of

x := 1

y := 2

And when **P** fails to terminate?

is

x := 1

a slice of

x := 1
while true do skip od

or

while true do skip od
x := 1

Goal: Characterize Slicing using *Properties*

Two kinds

- Floor
- Ceiling

Floor Requirement

- A “floor” requirement describes

  “must includes”

- For example

  a program must be a valid slice of itself

Ceiling Requirement

- A “ceiling” requirement describes

  “must avoids”

- For example

  a non-terminating program cannot be

  a slice of a terminating program

More formally

- Define a slicing relation as a set of triples <**S,P,X**>
  - **S** is a slice of **P** taken with respect to **X**

- Slicing relation **R** satisfies

  floor requirement **F** iff $\mathbf{F} \subseteq \mathbf{R}$

  ceiling requirement **C** iff $\mathbf{R} \subseteq U - \mathbf{C}$

A Floor Example

Weaken Criterion

If $\langle \mathbf{S},\mathbf{P},\mathbf{X} \rangle \in \mathbf{R}$ and $\mathbf{X'} \subset \mathbf{X}$

then $\langle \mathbf{S},\mathbf{P},\mathbf{X'} \rangle \in \mathbf{R}$

Example

"if **S** preserves the final value of **a** and **b** then it preserves the final value of **a**"

$\langle \mathbf{S},\mathbf{P},\{\mathbf{a},\mathbf{b}\} \rangle \in \mathbf{R}$ requiries $\langle \mathbf{S},\mathbf{P},\{\mathbf{a}\} \rangle \in \mathbf{R}$

Another Floor Example

Truncation

< *S1, S1;S2,* {**X**} **>** must be included

if *S2* is ***X****-preserving*

Statement *S* is ***X****-preserving* if it does not change the value of **X**

(bit of a simplification)

Final Floor Example

Ditchability

The slicing relation should allow deletion of any code that does not affect the value of any variable of interest

Ceiling Examples

Termination Preserving

Must avoid **<S, P, X>** when **P** terminates and **S** does not

iow, all slices of **P** must terminate when **P** terminates

Ceiling Examples

Non-termination Preserving

Must avoid **<S, P, X>** when **P** diverges and **S** does not

iow, all slices of **P** must diverge when **P** diverges

Ceiling Examples

Behaviour Preserving

<**S, P, X>** requires that when **P** terminates

S also terminates

$\forall \mathbf{x} \in \mathbf{X}$ **x**’s final value is the same for **S** and **P**

Properties allow Characterizing Slicing Approaches

- **Weiser's Definition**
  - Slice **S** of Program **P** is an executable program obtained from **P** by removing statements, such that **S** replicates part of the behaviour of **P**

- Definition is Termination Preserving
- Definition is **not** Non-termination Preserving

Slicing Defined using a Lazy Semantics

- A lazy or demand semantics can be used to define the semantic of a slice

```
x = 1
y = 2
z = x * 3   -- demands x but not y
```

- Does not satisfy *Behaviour Preservation* nor *Truncation*

Slicing Defined using Semi-refinement

For Programs *S1* and *S2*

S2 is a *semi-refinement* of *S1* provided

1. If *S1* terminates on state σ, then *S2* also terminates on σ with the same set of possible final states
2. If *S1* does not terminate, then *S2* can do *anything at all*

Slicing Defined using Semi-refinement

Using semi-refinement satisfies

- Weaken
- Identity
- Termination Preservation
- Ditchability
- Truncation
- but **not** Non-termination Preserving

Properties bound possible slicing relations

For example, the two properties

B*ehaviour Preservation*

and

Truncation

are sufficient to restrict possible slicing relations to just one: semi-refinement

Property Relations

Non-termination preservation is incompatible with ditchability.

Controversial?

Let **P** be

a = 42

a = 4

while true do skip

Slice **P** at the end with respect to *a*

S1:

a = 42

is a slice of **P**

as is **S2:**

a = 46

(a poll of attendees showed little controversy as none thought that **S1** *nor* **S2** *was a slice of* **P***)*