Parametric Shape Analysis via 3-valued Logic
Seminar about Static Program Analysis

Florian Haupenthal
Florian.Haupenthal@googlemail.com

Universität des Saarlandes – Fakultät 6.2 Informatik

12th June 2010
Motivation

What we already saw:
- predicate update formulae
- applicable for both concrete and abstract structures

What deranges us:
- application on abstract structures yields more imprecise results than on concrete ones
Motivation – Example

In the concrete world:

\[
y' = y > n
\]

\[
y'(v) = \exists v_1 : y(v_1) \land n(v_1, v)
\]
Motivation – Example

In the concrete world:

This result would abstract to:
Motivation – Example

From the concrete...

<table>
<thead>
<tr>
<th>indv.</th>
<th>x</th>
<th>y</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₃</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>u₁</th>
<th>u₂</th>
<th>u₃</th>
<th>u₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₂</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>u₃</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>u₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ x \longrightarrow u₁ \longrightarrow u₂ \longrightarrow u₃ \longrightarrow u₄ \]
Motivation – Example

From the concrete...

<table>
<thead>
<tr>
<th>indiv.</th>
<th>x</th>
<th>y</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₂</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₃</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>u₁</th>
<th>u₂</th>
<th>u₃</th>
<th>u₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₂</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>u₃</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>u₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Motivation – Example

...to the abstract world:

<table>
<thead>
<tr>
<th>indiv.</th>
<th>x</th>
<th>y</th>
<th>t</th>
<th>e</th>
<th>sm</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₂₃₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>u₁</th>
<th>u₂₃₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>0</td>
<td>1/2</td>
</tr>
<tr>
<td>u₂₃₄</td>
<td>0</td>
<td>1/2</td>
</tr>
</tbody>
</table>
Motivation – Example

In the abstract world:

<table>
<thead>
<tr>
<th>indiv.</th>
<th>x</th>
<th>y</th>
<th>t</th>
<th>e</th>
<th>sm</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₂₃₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>u₁</th>
<th>u₂₃₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>0</td>
<td>1/2</td>
</tr>
<tr>
<td>u₂₃₄</td>
<td>0</td>
<td>1/2</td>
</tr>
</tbody>
</table>

\[ y = y - n \]

\[ \text{In the abstract world:} \]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{indv.} & x & y & t \\
\hline
u₁ & 1 & 0 & 0 \\
u₂₃₄ & 0 & 1/2 & 0 \\
\hline
\end{array}
\]
Motivation – Example

In the abstract world:

<table>
<thead>
<tr>
<th>indiv.</th>
<th>x</th>
<th>y</th>
<th>t</th>
<th>e</th>
<th>sm</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u₂₃₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>u₁</th>
<th>u₂₃₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>0</td>
<td>1/2</td>
</tr>
<tr>
<td>u₂₃₄</td>
<td>0</td>
<td>1/2</td>
</tr>
</tbody>
</table>

\( y = y - n \)

Two differently precise results:
Motivation – Example

What we would like to have:
- precision from the first solution
- abstraction like in the second solution

How we try to get this:
- taking a closer look at our first simple approach
- extending abstract semantics to yield more precise results
Reinterpretation Principle

- Evaluation of the predicate-update formulae for a statement $st$ in 2-valued logic captures the transfer function for $st$ of the concrete semantics.
- Evaluation of the same formulae in 3-valued logic captures the transfer function for $st$ of the abstract semantics.
Tight Embedding

Minimal loss of information

Idempotence of a tight embedding:

We can apply our embedding function as often as we would like to, the result stays the same:

\[ t_{\text{embed}}_c(t_{\text{embed}}_c(S)) = t_{\text{embed}}_c(S) \]
Collecting Semantics

Similar to the concrete case.

Slightly different concerning potentially true information.
Collecting Semantics

\[
\text{If } v = \text{start} \\
\quad \text{StructSet}[v] = \{ (\emptyset, \emptyset) \} \\
\text{Otherwise} \\
\quad \text{StructSet}[v] = \\
\quad \quad \bigcup_{w \xrightarrow{v} \in \text{E}(G), \ w \in \text{As}(G)} \{ t_{\text{embed}}_c([st(w)]_3(S)) | S \in \text{StructSet}[w] \} \\
\quad \quad \bigcup_{w \xrightarrow{v} \in \text{E}(G), \ w \in \text{Id}(G)} \{ S | S \in \text{StructSet}[w] \} \\
\quad \quad \bigcup_{w \xrightarrow{v} \in \text{Tb}(G)} \{ S | S \in \text{StructSet}[w] \text{ and } S \models \text{cond}(w) \} \\
\quad \quad \bigcup_{w \xrightarrow{v} \in \text{Fb}(G)} \{ S | S \in \text{StructSet}[w] \text{ and } S \models \neg \text{cond}(w) \}
\]
Differences to concrete collecting semantics:

- Predicate-update formulae evaluated in 3-valued logic
- Bounded structures and application of $t_{\ embed_c}$ after every statement
- When $cond(w)$ evaluates to 1/2, information gets propagated along both the true and the false branches
Simple Shape-Analysis Algorithm

- Fixed point iteration over these equations
- In the beginning, for all program points $v$: $StructSet[v] = \emptyset$
- Terminates because of the bounded structures
Local Safety Theorem

- If $S^h$ embeds into $S$, a condition evaluated at a certain program point in $S^h$ is more precise than its evaluation in $S$.
- Updated concrete structures always embed into updated corresponding abstract ones.
Global Safety Theorem

The concrete structures for a certain program point form a subset of the concretisations of all abstract structures for this program point, always concerning the least-fixed-point solutions of the collecting semantics equations.
Overview

\[
\{ S_a \} \xrightarrow{\text{focus}} \{ S_{a,f,0}, S_{a,f,1}, S_{a,f,2} \}
\]

\[
\{ S_b \} \xleftarrow{\text{coerce}} \{ S_{b,1}, S_{b,2} \} \xrightarrow{\text{coerce}} \{ S_{a,o,0}, S_{a,o,1}, S_{a,o,2} \}
\]

\[
\{ S_{b,1}, S_{b,2} \} \xrightarrow{\text{coerce}} \{ S_{a,o,0}, S_{a,o,1}, S_{a,o,2} \}
\]

\[
\{ S_{a,f,0}, S_{a,f,1}, S_{a,f,2} \} \xrightarrow{\text{coerce}} \{ S_{a,o,0}, S_{a,o,1}, S_{a,o,2} \}
\]
Focus

Bringing Formulae into Focus:

*focus* generates a set of structures in which a given set of formulae has definite values for all assignments.

Danger of yielding an infinite set of structures!
Transform

- Reuse of our transfer functions
- Application on all structures, focus gives us
Coerce

Coercing into more precise structures:

*coerce* removes inconsistencies, possibly arised at focusing
Example

Input Structure:
Focused Structures (focused on $\varphi_0(v) \overset{\text{def}}{=} \exists v_1 : y(v_1) \land n(v_1, v)$):

\[
\begin{align*}
&\xrightarrow{n} \quad \xrightarrow{n} \quad \xrightarrow{n} \\
&u_1 \quad u \quad u_1 \quad u_1 \quad u.0 \\
&x, y, \quad x, y, \quad x, y, \quad x, y, \\
r_x, n, r_y, n \quad r_x, n, r_y, n \quad r_x, n, r_y, n \quad r_x, n, r_y, n
\end{align*}
\]
Focus

- Focusing on more formulae can yield more precision
- Focusing on more formulae can yield more arising structures, more expensive analysis

We need:
⇒ to choose wisely where to focus on.
⇒ an algorithm, taking a formula, returning a finite set of focused structures.
Focus – Choosing wisely

For statements like \( lhs = rhs \), we only focus on formulae, defining the heap cells for the left-hand side L-values and the right-hand side R-values. This ensures that the application of the abstract transformers does not set to 1/2 the entries of core predicates and fields that are updated by the statement.
Focus – Choosing wisely – Examples

\[ x = \text{NULL} \] nothing to focus on

\[ y = x \] \( y \) gets set, focusing on \( x \)

\[ y \rightarrow n = x \] \( n \) gets set, focusing on \( y \) and \( x \)

\[ x == y \] focusing on \( x \) and \( y \)
Focus – Algorithm

- Looking for structures, on which the given formula evaluates for a $u$ to $1/2$
- Splitting these structures, if $u$ is a summary node
- Repeat this until all structures have definite values for the given formula
No changes here!
Update

No changes here!

Simply reusing our predicate update formulae
Applying our well-known transformers:
Example

*Coerce* removes the inconsistent structure...

\[
x, r_x, n \quad y, r_x, n, r_y, n
\]

\[
x, r_x, n \quad y, r_x, n, r_y, n
\]
Example

...and removes some “repairable” inconsistencies:
... and removes some “repairable” inconsistencies:
The Sharpening Principle

In a structure $S$, the stored value for a predicate should be at least as precise as the predicate’s defining formula.

If the stored value is definite and the formula evaluates to an incomparable definite value, $S$ doesn’t represent any concrete store.
Compatibility Constraints

Definition:

A *Compatibility Constraint* is a term:

\[ \varphi_1 \triangleright \varphi_2 \]

\( \varphi_1 \) is an arbitrary 3-valued formula

\( \varphi_2 \) is either an atomic formula or the negation of one over distinct logical variables.

A *Compatibility Constraint* is similar to implication. If for an assignment \( \varphi_1 \) evaluates to 1, \( \varphi_2 \) has to do so, too.
Compatibility Constraints

Advantage:

Formulae are monotonic, Compatibility Constraints are not, in the right-hand sides.

Example:

$$1 \triangleright p$$

satisfied in a structure $S$ where $p$ is assigned to 1 but not in $S'$ where $p$ is assigned to $1/2$ but $S \sqsubseteq S'$. 
Coerce – Algorithm

- Search for assignments which violate the constraints
- If the violation is repairable: repair it
- If it is not: remove the structure
What did we do?

- Why did we start with these considerations?
- What did we do to improve our analysis?
What did we do?

- Why did we start with these considerations?
  - first approach with only simple predicate update formulae not as precise as we wanted it to be.
- What did we do to improve our analysis?
What did we do?

- Why did we start with these considerations?
  - first approach with only simple predicate update formulae not as precise as we wanted it to be.
- What did we do to improve our analysis?
  - We introduced an analysis with three steps: focusing, updating and coercing.
So again, what did we do?

- focus
- update
- coerce
So again, what did we do?

- **focus**
  - Increase precision by splitting structures where the given formulae evaluate to $1/2$ for a summary node $u$

- **update**

- **coerce**
So again, what did we do?

- **focus**
  - Increase precision by splitting structures where the given formulae evaluate to $1/2$ for a summary node $u$

- **update**
  - Applying our predicate update formulae on the focused structures

- **coerce**
So again, what did we do?

- **focus**
  - Increase precision by splitting structures where the given formulae evaluate to $1/2$ for a summary node $u$

- **update**
  - Applying our predicate update formulae on the focused structures

- **coerce**
  - Increase precision by removing some inconsistencies
The quintessence

Problem:
- most precise results on concrete structures but...
- ...possibly infinite number of concrete structures for one abstract structure

Solution:
- concretise only relevant parts for the analysis
Now here’s some space for your questions!
Thank you!

Thank you for your attention!