Class 16 – PE since Mix

- Review of specialization
- BTA improvements
- Binding-time optimizations
- User-annotated PE –MetaML and MetaOCaml
Review of specialization in Mix

Assume program:
\[
\begin{align*}
  f_0(x,y) &= \ldots \\
  f_1(x,y) &= \ldots \\
  \vdots
\end{align*}
\]

etc., with \( x \) known and \( y \) unknown.

Then assume it is annotated (i.e. operators are “e” or “r”)

Finally, assume given known input \( a \) for \( f_0 \)
worklist = \{f_{0,a}\}
residfns = \{
repeat until worklist empty {
  remove some function \(f_a\) from worklist
  residfns \supseteq \{f_a(y) = \text{simplify(body}(f), a)\}\}
}
simplify(e,a) = \text{case } e \text{ of }
  x: a \quad \text{y: "y"}
  k: k
  e_1 \oplus e_2: \text{simplify(e1,a) "\oplus" simplify(e2,a)}
  e_1 \oplus_e e_2: \text{simplify(e1,a) \oplus simplify(e2,a)}
  \text{callr } f (e_1,e_2): \text{let } v = \text{simplify(e1, a) in}
    \text{in worklist } \supseteq \{f_v\} – \text{residfns}
    e = \text{simplify(e2,a)}
    \text{return "call } f_v(e)"
  \text{call } f (e_1,e_2): \text{let } v_i = \text{simplify(ei,a) in}
    \text{simplify(body}(f)[\text{v2}/y], v1)
Advances in binding-time analysis

- Partially-static data structures
- Languages – e.g. higher-order functions, imperative features
- Monovariance vs. polyvariance
BTA viewed abstractly

BTA, like any static analysis, is basically a computation on a finite ordered set.

• Define finite ordered set $A$ representing possible outcomes of the BTA

• Assign to the program a monotonic function from $A$ to $A$.

• Starting at minimal value in $A$, iterate function until value doesn’t change.

More complex analyses involve larger sets $A$, but cost of finding fixedpoint increases.
Example: MIX analysis

A = FunctionName -> D*, where

- $D = \{K, U\}$
- For $\alpha, \alpha'$ in $A$, $\alpha \sqsubseteq \alpha'$ if, for all $f$, $\alpha(f) \sqsubseteq \alpha'(f)$ in $D^*$
  - $w \sqsubseteq w'$ in $D^*$ if for each $i$, $w_i \sqsubseteq w'_i$ in $D$
  - $d \sqsubseteq d'$ if $d=K$ or $d'=U$
- Minimal element is $\{f_0 -> KK.., f_1 -> KK.., \ldots\}$
- $|A| = 2^n$, where $n$ is total number of all arguments to all functions.

Function in $A -> A$ defined by program is given by $P$. 
Other analyses

MIX uses about the simplest possible analysis. Other analyses involve larger domains.

- In general, the larger the domain, the more expensive it is to find a fixedpoint.
- Some domains are naturally infinite, so something needs to be done to restrict them. This necessarily involves some loss of precision.
Partially-static data structures

- Pairs

- Lists
BTA for other languages

- Higher-order functions

- Imperative features
Monovariance vs. polyvariance

- Monovariance: Annotate functions
- Polyvariance: Annotate call sites
Binding-time optimizations

For given language and BTA, programs can often be modified to improve PE behavior.
E.g. split a-list into parallel lists

Other examples:
Binding-time optimizations – “The Trick”
User-annotated PE

Alternative approach: Have users supply BTA by annotation.

Simplest annotation is quote/anti-quote.

MetaML/MetaOCaml: ML-based languages with quote-antiquote

(Quote is $<$>, antiquote is $\sim$, lifting is “lift”)

MetaML example

fun member v L =
  if (null L)
  then <false>
  else <if ~v=~(lift hd L)
         then true
         else ~(member v (tl L))>
;

<fn x => ~(member <x> [1,2,3])>
produces
<fn x => if x=1 then true
         else if x=2 then true
         else if x=3 then true else false
MetaML type system

- Correct language types enforced by type system
  – staging has no impact
- New kind of possible type error: Using late value at an early stage.
- Example:
  ```
  fn a => <fn b => ~(a+b)>
  ```
Final words on MetaML

- Quoted fragments look similar to fragments in Jumbo and other systems. However,
- Open code fragments not allowed. Fragments not “first class”
- Still problems of controlling specialization
- Not well suited to *run-time* program generation
- Above all, process of writing program generators very different from explicit construction.