More
Data Flow Analyses

Reading: NNH 2.1

17-654/17-765
Analysis of Software Artifacts
Jonathan Aldrich

Available Expressions

For each program point, which expressions *must* have already been computed, and not later modified, on all paths to the program point.

- Applications
  - Avoid re-computing expressions

Available Expressions Example

[x := a+b]^1;

[y := a*b]^2;

while [y > a+b]^3 do

  [a := a+1]^4;

  [x := a+b]^5

$AE_{exit}(1) = \{a+b\}$

$AE_{exit}(2) = \{a+b, a*b\}$

$AE_{exit}(3) = \{a+b\}$

$AE_{exit}(4) = \emptyset$

$AE_{exit}(5) = \{a+b\}$

Available Expressions Equations

$[x := a+b]^1;$
$[y := a*b]^2;$
while $[y > a+b]^3$ do
    $[a := a+1]^4;$
    $[x := a+b]^5$

$AE_{entry}(1) = \emptyset$
$AE_{entry}(2) = AE_{exit}(1)$
$AE_{entry}(3) = AE_{exit}(2) \cap AE_{exit}(5)$
$AE_{entry}(4) = AE_{exit}(3)$
$AE_{entry}(5) = AE_{exit}(4)$

$AE_{exit}(1) = AE_{entry}(1) \cup \{a+b\}$
$AE_{exit}(2) = AE_{entry}(2) \cup \{a*b\}$
$AE_{exit}(3) = AE_{entry}(3) \cup \{a+b\}$
$AE_{exit}(4) = AE_{entry}(4) \setminus \{a+b, a*b, a+1\}$
$AE_{exit}(5) = AE_{entry}(5) \cup \{a+b\}$

Another Example

<u>Program</u>

[x := a+b]^1;

while [true]^2 do

  [skip]^3;

<u>Dataflow Eqns</u>

$AE_{entry}(1) = \emptyset$

$AE_{entry}(2) = AE_{exit}(1) \cap AE_{exit}(3)$

$AE_{entry}(3) = AE_{exit}(2)$

$AE_{exit}(1) = AE_{entry}(1) \cup \{a+b\}$

$AE_{exit}(2) = AE_{entry}(2)$

$AE_{exit}(3) = AE_{entry}(3)$

<u>Solutions</u>

$\emptyset$

$\emptyset$ or $\{a+b\}$

$\emptyset$ or $\{a+b\}$

$\{a+b\}$

$\emptyset$ or $\{a+b\}$

$\emptyset$ or $\{a+b\}$

What's the fixed point of these equations?

– Which is most informative?

Available Exp. vs. Reaching Defs.

- Available Exp. ***Must*** analysis
  - Initial dataflow values: *empty* sets
  - *Intersection* at control flow merge
  - Precision: want *greatest* fixed point
  - Safety: err on the side of *smaller* sets
- Reaching Defs. ***May*** analysis
  - Initial dataflow values: *universal* sets
  - *Union* at control flow merge
  - Precision: want *least* fixed point
  - Safety: err on the side of *larger* sets

Kill and Gen Sets

- Useful for defining transfer functions
  - Kill: dataflow facts the statement removes
  - Gen: dataflow facts the statement adds
  - Kill always comes first

Kill and Gen for Available Expressions

$\text{kill}_{AE}([x := a]^{\ell}) = \{a' \in \mathbf{AExp}_{*} \mid x \in FV(a')\}$

$\text{kill}_{AE}([\text{skip}]^{\ell}) = \emptyset$

$\text{kill}_{AE}([b]^{\ell}) = \emptyset$

$\text{gen}_{AE}([x := a]^{\ell}) = \{a' \in \mathbf{AExp}(a) \mid x \notin FV(a')\}$

$\text{gen}_{AE}([\text{skip}]^{\ell}) = \emptyset$

$\text{gen}_{AE}([b]^{\ell}) = \mathbf{AExp}(b)$

We should define this!

Definition of **AExp**(a), **AExp**(b)

AExp(x) = ∅

AExp(n) = ∅

AExp(a_1 *op*$_a$ a_2) = {a_1 *op*$_a$ a_2} ∪ AExp(a_1) ∪ AExp(a_2)

AExp(true) = ∅

AExp(false) = ∅

AExp(not b) = {not b} ∪ AExp(b)

AExp(b_1 *op*$_b$ b_2) = {b_1 *op*$_b$ b_2} ∪ AExp(b_1) ∪ AExp(b_2)

AExp(a_1 *op*$_r$ a_2) = {a_1 *op*$_r$ a_2} ∪ AExp(a_1) ∪ AExp(a_2)

You'll do something similar on the homework

Kill and Gen for Reaching Definitions

$\text{kill}_{AE}([x := a]^{\ell}) = \{(x,?)\} \cup \{(x,\ell) \mid \ell \textit{ is any label}\}$

$\text{kill}_{AE}([\text{skip}]^{\ell}) = \emptyset$

$\text{kill}_{AE}([b]^{\ell}) = \emptyset$

$\text{gen}_{AE}([x := a]^{\ell}) = \{(x,\ell)\}$

$\text{gen}_{AE}([\text{skip}]^{\ell}) = \emptyset$

$\text{gen}_{AE}([b]^{\ell}) = \emptyset$

Some Notation

- This will help us describe analyses in a more precise and general way
  - *init*(S) – the label of the first statement in S
  - *final*(S) – the set of labels of the last statements in S
    - the last statement on each branch of an if
    - the test of a while
  - *blocks*(S) – the set of primitive statements and tests in S
  - *labels*(S) – the set of labels of blocks in S
  - *flow*(S) – {(ℓ,ℓ’) | control may transfer from block ℓ to block ℓ’ }
    - A pair for each edge in the control flow graph

- The text defines these formally

General Data Flow Equations

Available Expressions

$$AE_{entry}(\ell) = \begin{cases} \emptyset & \text{if } (\ell = \mathit{init}(S_*)) \\ \cap \{ AE_{exit}(\ell') \mid (\ell', \ell) \in \mathit{flow}(S_*) \} & \text{otherwise} \end{cases}$$

$$AE_{exit}(\ell) = (AE_{entry}(\ell) \setminus kill_{AE}(B^\ell)) \cup gen_{AE}(B^\ell)$$

Reaching Definitions

$$RD_{entry}(\ell) = \begin{cases} \{(x,?) \mid x \in FV(S_*)\} & \text{if } \ell = \mathit{init}(S_*) \\ \cup \{ RD_{exit}(\ell') \mid (\ell', \ell) \in \mathit{flow}(S_*) \} & \text{otherwise} \end{cases}$$

$$RD_{exit}(\ell) = (RD_{entry}(\ell) \setminus kill_{RD}(B^\ell)) \cup gen_{RD}(B^\ell)$$

Constant Propagation

- For each program point, what value *may* each variable hold?

Constant Propagation

$$CP_{entry}(\ell) = \begin{cases} \{(x,n) \mid x \in FV(S_*),\ n \in N\} & \text{if } (\ell = \mathit{init}(S_*)) \\ \cup \{ CP_{exit}(\ell') \mid (\ell', \ell) \in \mathit{flow}(S_*) \} & \text{otherwise} \end{cases}$$

$$CP_{exit}(\ell) = (CP_{entry}(\ell) \setminus kill_{CP}(B^\ell)) \cup gen_{CP}(B^\ell)$$

$kill_{CP}([x := a]^\ell) = \{ (x,n) \mid n \in N \}$
$kill_{CP}([skip]^\ell) = \emptyset$
$kill_{CP}([b]^\ell) = \emptyset$

$gen_{CP}([x := a]^\ell) = \{ (x, n) \mid n \in \mathbf{CP}^\ell(a)) \}$
$gen_{CP}([skip]^\ell) = \emptyset$
$gen_{CP}([b]^\ell) = \emptyset$

Constant Propagation

$\mathbf{CP}^{\ell}(x) = \{ CP_{entry}(\ell)(x) \}$

$\mathbf{CP}^{\ell}(n) = \{ n \}$

$\mathbf{CP}^{\ell}(a_1\ op_a\ a_2) = \mathbf{CP}^{\ell}(a_1)\ \widehat{op}_a\ \mathbf{CP}^{\ell}(a_2)$

$set_1\ \widehat{op}_a\ set_2 = \{ n_1\ op_a\ n_2 \mid n_1 \in set_1,\ n_2 \in set_2 \}$

Constant Propagation Example

[y := 5]^1;
[z := 1+y]^2;
if [x = 5]^3
    then [w := x+1]^4;
    else [w := y+1]^5
[skip]^6

	x	y	z	w
$AE_{exit}(1)$ =	?	5	?	?
$AE_{exit}(2)$ =	?	5	6	?
$AE_{exit}(3)$ =	?	5	6	?
$AE_{exit}(4)$ =	?	5	6	?
$AE_{exit}(5)$ =	?	5	6	6
$AE_{exit}(6)$ =	?	5	6	?

Here ? is a shorthand for the set of all integers

Constant Propagation Example

[y := 5]^1;
[z := 1+y]^2;
if [x = 5]^3
    then [w := x+1]^4;
    else [w := y+1]^5
[skip]^6

	x	y	z	w
$AE_{exit}(1)$ =	?	5	?	?
$AE_{exit}(2)$ =	?	5	6	?
$AE_{exit}(3)$ =	?	5	6	?
$AE_{exit}(4)$ =	?	5	6	?
$AE_{exit}(5)$ =	?	5	6	6
$AE_{exit}(6)$ =	?	5	6	?

But we know that x=5 at statement 4. Can we do better?

Constant Propagation, Take 2

$CP_{entry}(\ell)$ = $\{(x,n) \mid x \in FV(S_*), n \in N\}$ if ($\ell = init(S_*)$)
= $\cup\{ CP_{exit}(\ell') \mid (\ell', \ell) \in flow(S_*) \}$ otherwise

$CP_T(\ell)$ = $(CP_{entry}(\ell) \setminus kill_{CP,T}(B^\ell)) \cup gen_{CP,T}(B^\ell)$
$CP_F(\ell)$ = $(CP_{entry}(\ell) \setminus kill_{CP,F}(B^\ell)) \cup gen_{CP,F}(B^\ell)$

$kill_{CP,T}([x := a]^\ell) = \{ (x, n) \mid n \in N \}$
$kill_{CP,T}([skip]^\ell) = \emptyset$
$kill_{CP,T}([b]^\ell) = \{ (x, n) \mid n \in N \text{ and } n \neq m\}$ when $b = (x=a)$ and $\mathbf{CP}^\ell(a)=\{m\}$
$= \emptyset$ otherwise
$kill_{CP,F}([b]^\ell) = \{ (x, m) \}$ when $b = (x=a)$ and $\mathbf{CP}^\ell(a)=\{m\}$
$= \emptyset$ otherwise

$gen_{CP,T}([x := a]^\ell) = \{ (x, n) \mid n \in \mathbf{CP}^\ell(a)) \}$
$gen_{CP,T}([skip]^\ell) = \emptyset$
$gen_{CP,T}([b]^\ell) = \emptyset$
$gen_{CP,F}([b]^\ell) = \emptyset$

Constant Propagation Example

[y := 5]¹;
[z := 1+y]²;
if [x = 5]³
    then [w := x+1]⁴;
    else [w := y+1]⁵
[skip]⁶

	x	y	z	w
$AE_{exit}(1)$ =	?	5	?	?
$AE_{exit}(2)$ =	?	5	6	?
$AE_{T}(3)$ =	5	5	6	?
$AE_{F}(3)$ =	?\5	5	6	?
$AE_{exit}(4)$ =	5	5	6	6
$AE_{exit}(5)$ =	?\5	5	6	6
$AE_{exit}(6)$ =	?	5	6	6

Keeping track of data flow values separately on each branch supports a more precise final result.