COS320 Dataflow analysis & ~~Register allocation~~ Optimization II.

recall constant propagation.

A const. env. is a symbol table that maps a variable to an int n, ⊤ (more than one value), ⊥ (less than one value)

IN, OUT are conservative if

Constant System:

$\forall$ bb $\in N$,

$OUT[bb] \sqsupseteq post(bb, IN[bb])$ (Const env at end of bb)

$\forall$ edge $(src, dst) \in E$:

$IN[dst] \sqsupseteq OUT[src]$

Worklist Algorithm (work set) ← Choose broken constraint, fix it until no broken constraints.

Input: Control-flow graph (N, E, s) with variables $x_1 \dots x_n$.

Output: least conservative assignment of constant environments.

$IN[s] = \{x_1 \mapsto \top, \dots, x_n \mapsto \top\}$

$OUT[s] = \{x_1 \mapsto \bot, \dots, x_n \mapsto \bot\}$

$IN[n] = OUT[n] = \{x_1 \mapsto \bot, \dots, x_n \mapsto \bot\}$ $\forall$ other nodes n;

work ← N;

↓

while work $\neq \emptyset$ do

Pick some n from work;

work $\leftarrow$ work $\setminus \{n\}$;

old $\leftarrow$ OUT[n];

$IN[n] \leftarrow \bigsqcup_{p \to n \in E} OUT[p]$;

$OUT[n] \leftarrow Post(n, IN[n])$;

if old $\neq$ OUT(n) then

  work $\leftarrow$ work $\cup$ succ(n);

(succ: successor)

return IN, OUT

Q: What is the runtime of this algorithm?

Common Subexpression Elimination

- Search for expressions that
  - Appear at multiple points in the cfg
  - replace the cost of recalculation by storing the value.

Available expressions:

An expression e is available at a bb if ∀ path from s to n in G

① the expression e is evaluated along the path

② after the last evaluation of e along the path, no variables in e are overwritten

Propagating available expressions

given a set of expressions E and an instruction $x = e$.

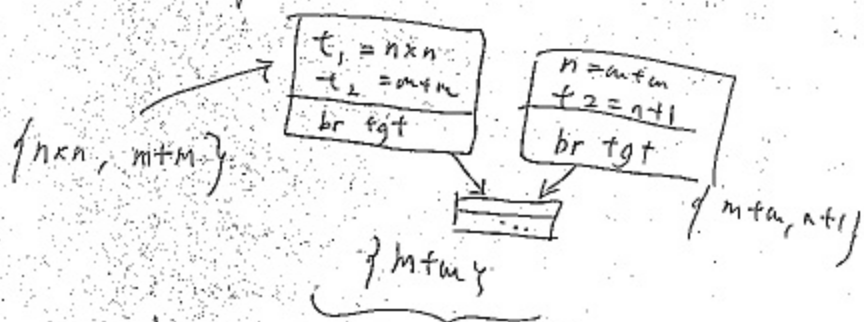
Assuming the set of expressions E available before the instruction, what expressions are available after the instruction?

→ Add in e, remove everything involving x.

$$Post_{AE}(x=e, E) \doteq \{e' \in E \cup \{e\} : x \text{ not in } e'\}$$

Q: how to propagate instrs across a bb?

- Break bb up into list of instrs... term.
- Take $post_{AE}(bb, \sqsubseteq) = post_{AE}(term, post_{AE}...)$ Composition
- How to combine info from multiple predecessors?



Answer: take intersection.

Now we can formulate this as a constraint system...

Let $G = (N, E, s)$ be a ctg.

$\forall bb \in N$, associate $IN[bb]$ $OUT[bb]$

$IN[bb]$ = set of exprs available at entry of bb.

$OUT[bb]$ = set of exprs available at exit of bb.

Assignments IN, OUT. are conservative if

$IN(s) = \emptyset$ ...

$\forall\ bb \in N$

$OUT \subseteq post_{AE}(bb, IN[bb])$

$\forall$ edge $(s, t) \in E$

$IN[t] \subseteq OUT(s)$

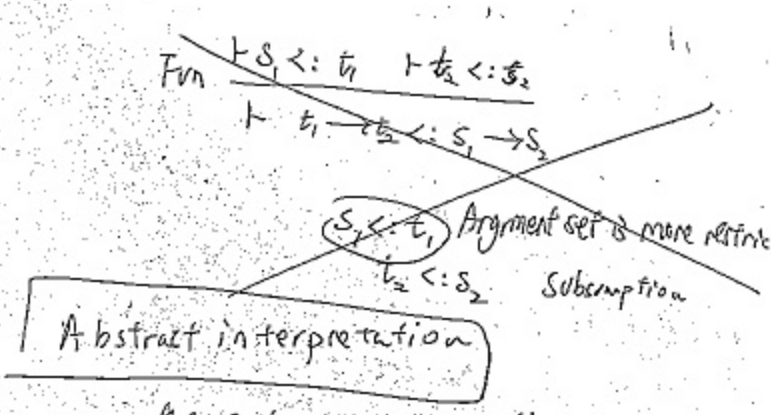
- Augment Worklist algorithm to compute this constraint system.

(output: least conservative assignment of available expressions)

COMMONALITY between const prop
common subexp elimination

→ Optimal solutions to system of local constraints.

- "local": In terms of edges, contrast w/ "global", which depends on structure of whole graph.



Abstract interpretation

general theory relating program analysis to program semantics

- dataflow analyses are all about solving a local constraint system

- What does it mean for a constraint system to be correct?

- How do we prove it?

data flow analysis and proving analysis correct

(And more on solving constraint systems)

forward dataflow analysis:

- Abstract domain L
  defines space of program properties we're interested
- Abstract transformer $Post_L$
  determines how each BB transforms properties
  e.g. if property p holds before n,
  then $post_L(n, p)$ is a property that holds after n...

think about this like an interface...

parametrizing a worklist algorithm by these two arguments.

Abstract domain: set $\mathcal{L}$ w/ a partial order $\sqsubseteq$

$x \sqsubseteq y \Rightarrow x$ rep's more precise information about the program than y.

(the opposite direction also works... but we'll use this here).

② A least upper bound (join) operator $\sqcup$

a. $x \sqsubseteq (x \sqcup y)$

b. $y \sqsubseteq (x \sqcup y)$

c. $x \sqcup y \sqsubseteq z$ $\forall z$ satisfying 1,2

③ A least elem $\bot$ (identity element)

a. $\bot \sqsubseteq x \quad \forall x$

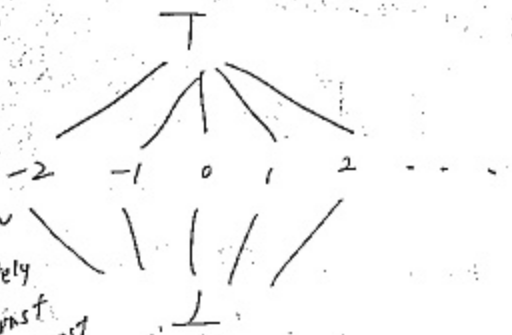
b. $\top \sqcup x = x \sqcup \top = \top \quad \forall x.$

④ A greatest element $\top$

- $x \sqsubseteq \top \quad \forall x$
- $\top \sqcup x = x \sqcup \top = \top \quad \forall x.$

It is often convenient posets as Hasse diagram
- small things on bottom big things on top
- Arc from x to y if $x \sqsubseteq y$ and
$\neg \exists z \quad x \sqsubseteq z \sqsubseteq y$
(y covers x)
- $x \sqsubseteq y$ iff upwards path from x to y

Hasse diagram:
where all integers
are incompamble



this represents
the constant propagation
domain. (more accurately
the range of const
env for const
prop).

Function spaces

Constant envs are maps $\{\text{Variables}\} \to \mathbb{Z} \cup \{\bot, \top\}$

Environments inherit pointwise ordering $\sqsubseteq^*$
from $\sqsubseteq$ on $\mathbb{Z} \cup \{\bot, \top\}$

$f \sqsubseteq^* g \iff f(x) \sqsubseteq g(x) \quad \forall x \in Vars$

- least/greatest environment

$$\bot^* = (\text{fun } x \to \bot)$$

$$\top^* = (\text{fun } x \to \top)$$

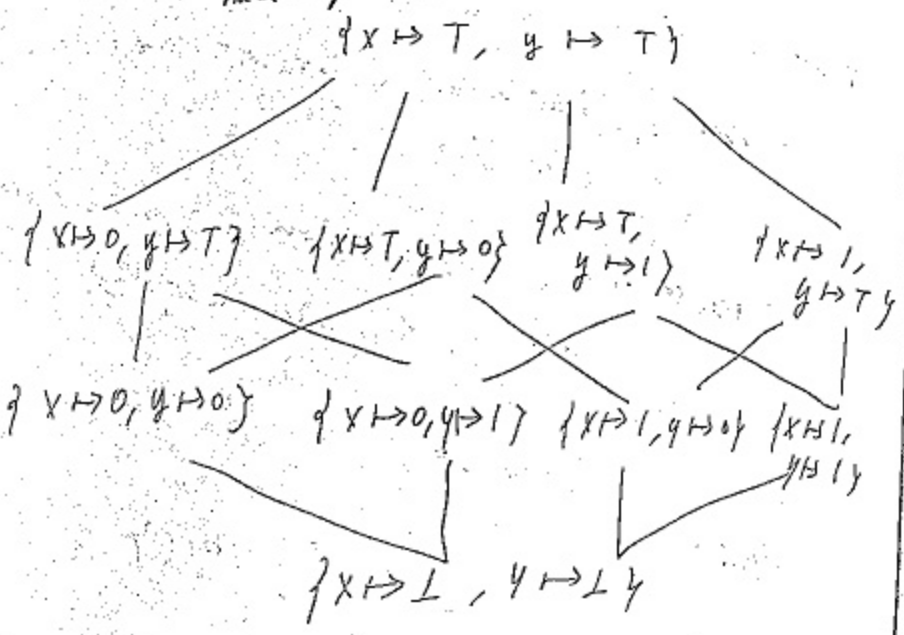
environments also have least upper bounds

$$f \sqcup^* g = (\text{fun } (x) \to f(x) \sqcup g(x))$$

this holds more generally.

If L is abstract domain and X is any set, the set of functions $X \to L$ is an abstract domain under the pointwise ordering.

Example: ~~Const. Env.~~ for two variables set
hasse diagram



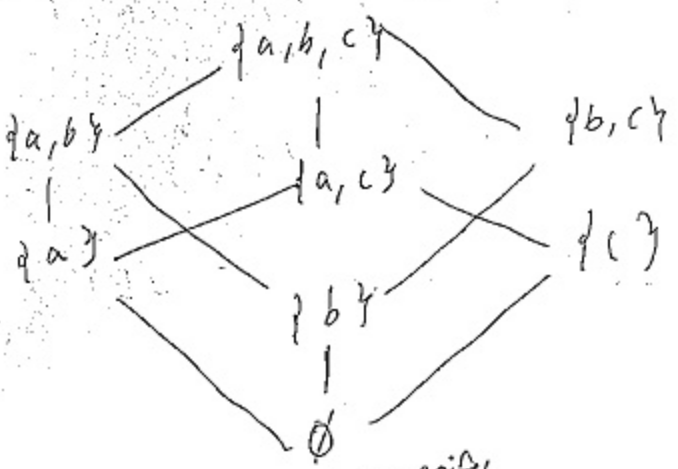
here $\$ \sqsubseteq \top$ for $\forall s \in S$ and $\forall t \in T$ $\bot \sqsubseteq t$

Powersets

$\forall$ set X, 2^X is an abstract domain:

- Order $\subseteq$, least element $\emptyset$, greatest element X, join $\cup$

- reverse order $\supseteq$, least elem X, greatest elem $\emptyset$, join $\cap$ ← (Available exprs).



☆ Caveat: Can use this to generify a worklist algorithm. When we have a reverse order, just flip it.

Transfer Functions

transfer function.

$Post_{\mathcal{L}} : BB \times \mathcal{L} \to \mathcal{L}$

maps each BB and "pre-state" value to a post-state value.

We require that $Post_{\mathcal{L}}$ is monotone:

$$x \sqsubseteq y \implies Post_{\mathcal{L}}(n, x) \sqsubseteq Post_{\mathcal{L}}(n, y).$$

"more info in → more info out".

have to prove this on the abstract domain.

☆: Monotonicity $\not\Rightarrow$ $x \sqsubseteq f(x)$ $\forall x$.

this is the extensive property.

Generic fwd dataflow analysis algorithm

Given Abstract domain $L, \sqsubseteq, \sqcup, \bot, \top$

Transfer function $Post_L : bb \times L \to L$

Cfg $G = (N, E, s)$

Compute least annotation IN, OUT such that

- $IN(s) = \top$
- $\forall n \in N,\ post_c(n, IN[n]) \sqsubseteq OUT[n]$
- $\forall p \to n \in E,\ OUT[p] \sqsubseteq IN(n)$.

Generic Algorithm:

$IN[s] = \top$, $OUT[s] = \bot$;

$IN[n] = OUT[n] = \bot$;

for all other nodes n;

Work $\leftarrow N$;

While work $\neq \emptyset$ do

  Pick some n from work;

  work $\leftarrow$ work $\setminus \{n\}$;

  old $\leftarrow OUT[n]$;

  $IN[n] = \bigsqcup_{p \to n \in E} OUT[p]$;

  if old $\neq OUT[n]$ then work $\leftarrow$ work $\cup succ(n)$;

return IN, OUT.