Fixpoint equations

Goal

- Many problems in programming languages can be formulated as a set of mutually recursive equations of the following form:
  - Example: fixing first/follow sets in LL(k) parsing
  - PL semantics
  - Dataflow analysis
  - Type inference

- General questions
  - What assumptions on D, f, and g are sufficient to ensure that such a system of equations has a solution?
  - If a system has multiple solutions, which solution is the one we really want?
  - How do we compute that solution?

- Keywords:
  - Assumptions on D: partially ordered set, semi-lattice, lattice, complete lattice, ...
  - Assumptions on functions f, g: monotonic, continuous, extensive, ...
  - Solutions: fixpoint, least fixpoint, greatest fixpoint, ...

Game plan

- Finite partially-ordered set with least element: D
- Function f: D→D
- Monotonic function f: D→D
- Fixpoints of monotonic function f:D→D
  - Least fixpoint
- Solving equation x = f(x)
  - Least solution is least fixpoint of f
- Generalization to case when D has a greatest element T
  - Least and greatest solutions to equation x = f(x)
- Generalization of systems of equations
- Semi-lattices and lattices

Partially-ordered set

- Set D with a binary relation ≤ that is
  - reflexive: x ≤ x
  - anti-symmetric: x ≤ y and y ≤ x ⇒ x = y
  - transitive: x ≤ y and y ≤ z ⇒ x ≤ z
- Example: set of integers ordered by standard < relation
  - Graphical representation of poset:
    - Graph in which nodes are elements of D and relation ≤ is shown by arrows
    - Usually we omit transitive arrows to simplify picture

Another example of poset

- Powerset of any set ordered by set containment is a poset
- In example shown to the left, poset elements are (), {a}, {a,b},{a,b,c}, etc.
  - x ≤ y if x is a subset of y

Finite poset with least element

- Poset in which
  - set is finite
    - there is a least element that is below all other elements in poset
- Examples:
  - Set of primes ordered by natural ordering is a poset but is not finite
  - Factors of 12 ordered by natural ordering on integers is a finite poset with least element
  - Powerset example from previous slide is a finite poset with least element ({}
Domain

- Since “finite partially-ordered set with a least element” is a mouthful, we will just abbreviate it to “domain”.
- Later, we will generalize the term “domain” to include other posets of interest to us in the context of dataflow analysis.

Functions on domains

- If D is a domain, f:D→D
  - a function maps each element of D to some element of D itself
- Examples: for D = powerset of {a,b,c}
  - f(x) = x U {a}
    - so f maps {} to {a}, {b} to {a,b} etc.
  - g(x) = x \{a\}
  - h(x) = {a} - x

Monotonic functions

- Function f: D→D where D is a domain is monotonic if
  - x ≤ y ⇒ f(x) ≤ f(y)
- Common confusion: people think f is monotonic if x < f(x). This is a different property called extensivity.
- Intuition:
  - think of f as an electrical circuit mapping input to output
  - f is monotonic if increasing the input voltage causes the output voltage to increase or stay the same
  - f is extensive if the output voltage is greater than or equal to the input voltage

Examples

- Domain D is powerset of {a,b,c}
- Monotonic functions: (x in D)
  - x → {} (why?)
  - x → x U {a}
  - x → x \{a\}
- Not monotonic:
  - x → {a} - x
- Why? Because {} is mapped to {a} and {a} is mapped to {}.
- Extensivity
  - x → x U {a} is extensive and monotonic
  - x → x \{a\} is not extensive but monotonic
- Exercise: define a function on D that is extensive but not monotonic

Fixpoint of f:D→D

- Suppose f: D→D. A value x is a fixpoint of f if f(x) = x. That is, f maps x to itself.
- Examples: D is powerset of {a,b,c}
  - Identity function: x → x
    - Every point in domain is a fixpoint of this function
  - x → x U {a}
    - {}, {a}, {a,b}, {a,c}, {a,b,c} are all fixpoints
  - x → {a} - x
    - no fixpoints

Fixpoint theorem(I)

- If D is a domain, ⊥ is its least element, and f:D→D is monotonic, then f has a least fixpoint that is the largest element in the sequence (chain)
  - ⊥, f(⊥), f(f(⊥)), f(f(f(⊥))), ….
- Examples: for D = power-set of {a,b,c}, so ⊥ is {}
  - Identity function: sequence is {}, {}, {}, … so least fixpoint is {}, which is correct.
  - x → x U {a}; sequence is {}, {a}, {a}, {a}, … so least fixpoint is {a} which is correct.
Proof of fixpoint theorem

- Largest element of chain is a fixpoint:
  - ⊥ ≤ f(⊥) (by definition of ⊥)
  - f(⊥) ≤ f(f(⊥)) (from previous fact and monotonicity of f)
  - f(f(⊥)) ≤ f(f(f(⊥))) (same argument)
  - since the set D is finite, this chain cannot grow arbitrarily, so it has some largest element that f maps to itself. Therefore, we have constructed a fixpoint of f.
- This is the least fixpoint
  - let p be any other fixpoint of f
  - ⊥ < p (from definition of ⊥)
  - So f(⊥) ≤ f(p) (monotonicity of f)
  - similarly f(f(⊥)) ≤ p etc.
  - therefore all elements of chain are ≤ p, so largest element of chain must be ≤ p
  - therefore largest element of chain is the least fixpoint of f.

Solving equations

- If D is a domain and f:D→D is monotonic, then the equation x = f(x) has a least solution given by the largest element in the sequence ⊥, f(⊥), f(f(⊥)), f(f(f(⊥))), ... 
- Proof: follows trivially from fixpoint theorem

Easy generalization

- Proof goes through even if D is not a finite set but only has finite height
  - no infinite chains

Another result

- If D is a domain with a greatest element T and f:D→D is monotonic, then the equation x = f(x) has a greatest solution given by the smallest element in the descending sequence T, f(T), f(f(T)), f(f(f(T))), ...
- Proof: left to reader

Functions with multiple arguments

- If D is a domain, a function f(x,y):DxD→D that takes two arguments is said to be monotonic if it is monotonic in each argument when the other argument is held constant.
- Intuition:
  - electrical circuit has two inputs
  - if you increase voltage on any one input keeping voltage on other input fixed, the output voltage stays the same or increases

Fixpoint theorem(II)

- If D is a domain and f,g:DxD→D are monotonic, the following system of simultaneous equations has a least solution computed in the obvious way.
  \[ x = f(x,y) \]
  \[ y = g(x,y) \]
- You can easily generalize this to more than two equations and to the case when D has a greatest element T.
Upper and lower bounds

- If \( D, \leq \) is a poset and \( S \subseteq D \), \( l \in D \) is a lower bound of \( S \) if
  \[ \forall x \in S. \quad l \leq x \]
- Example: lower bounds of \( \{c,d\} \) are \( d \) and \( f \)
- In general, a given \( S \) may have many lower bounds.
- Greatest lower bound (glb) of \( S \): greatest element of \( D \) that is a lower bound of \( S \)
  - Caveat: glb may not always exist
  - Example: lower bounds of \( \{b,c\} \) are \( d, e, f \) but there is no glb
- If for every pair of elements \( x, y \in D \), \( \text{glb}(\{x,y\}) \) exists, we can define a function called meet \( \wedge : D \times D \to D \)
  - Idempotent: \( x \wedge x = x \)
  - Commutative: \( x \wedge y = y \wedge x \)
  - Associative: \( x \wedge (y \wedge z) = (x \wedge y) \wedge z \)
- Analogous notions: upper bounds, least upper bounds, join \( \vee \)
- Meet semilattice:
  - partially ordered set in which every pair of elements has a glb
- Join semilattice
  - analogous notion
- Lattice:
  - both a meet and join semilattice

Fixpoint equations in lattices

- If \( (D, \leq, \wedge, \vee) \) is a finite lattice, it has a least and greatest element.
- Meet and join functions are monotonic
- Therefore, if \( (D, \leq, \wedge, \vee) \) is a finite lattice, fixpoint theorem (II) applies even if some of the functions \( f, g \) etc. are \( \wedge \) or \( \vee \)

Computing the least solution for a system of equations

- Consider
  \[
  \begin{align*}
  x &= f(x,y,z) \\
  y &= g(x,y,z) \\
  z &= h(x,y,z)
  \end{align*}
  \]
- Obvious iterative strategy: evaluate all equations at every step (Jacobi iteration)
  \[
  \begin{bmatrix}
  f(\bot, \bot, \bot) \\
  g(\bot, \bot, \bot) \\
  h(\bot, \bot, \bot)
  \end{bmatrix}
  \]
  ...

Work-list based algorithm

- Obvious point: it is not necessary to reevaluate a function if its inputs have not changed
- Worklist based algorithm:
  - initialize worklist with all equations
  - initialize solution vector \( S \) to all \( \bot \)
  - while worklist not empty do
    - get equation from worklist
    - evaluate rhs of equation with current solution vector values and update entry corresponding to the variable in solution vector
    - put all equations that use this variable in their RHS on worklist
- You can show that this algorithm will compute the least solution to the system of equations