FIXED POINT ALGORITHMS

Cameron Moy

March 13, 2018

There is a class of algorithms called fixed point (hence abbreviated fixpoint) algorithms. Such algorithms often have elegant and short implementations.

Definition. A fixpoint of a function $f : X \to Y$ is an element $a \in X$ such that $f(a) = a$.

Example. Let $f(x) = x^2 - 3x + 4$. Since $f(2) = 2$, then 2 is a fixpoint of f.

Example (Newton's method)**.** Given function f, a fixpoint of $g(x) = x - \frac{f(x)}{f'(x)}$ is a root of f.

This is all well and good, but how does one actually compute a fixpoint? In Newton's method (under certain conditions) we can start with a guess x_0, repeatedly apply g, and eventually get close to a fixpoint. In other words, $g(g(\dots g(x_0)\dots)) \approx a$ where $f(a) = 0$. For some functions, we are guaranteed this iteration will terminate and reach a fixpoint.

Example (Reachability)**.** Let $G = (V, E)$ be a graph. The set of vertices reachable from $u \in V$ is a fixpoint of $g : \mathcal{P}(V) \to \mathcal{P}(V)$ where $g(S) = S \cup \{v \mid u \in S, (u, v) \in E\}$. Specifically, the fixpoint of g where $x_0 = \{u\}$.

This is very formal, let's see some pseudo-code. First, we define a function `fix` that computes the fixpoint of g given initial value x_0. We repeatedly apply g until a fixpoint is reached.

```
fix g x0 =
  let x1 = g x0 in
  if x0 = x1 then x0
  else fix g x1
```

Assume we have two functions: `union` that takes the union of a family of sets, and `adj` that takes a graph G, a vertex v, and returns the set of vertices adjacent to v. Now, let's write our function g for graph reachability.

```
g G S = union {S, map (adj G) S}
```

So a function `reachable` that takes graph G, vertex v, and returns the set of vertices reachable from v may be implemented as follows.

```
reachable G v = fix (g G) {v}
```

Example. ε-closure is similar to `reachable` and is most easily computed by fixpoint algorithm.

Example. The powerset construction may also be computed by fixpoint algorithm. Assume our NFA is $(Q, \Sigma, \delta, q_0, F)$. First, we define an analogue of `adj` from above. Let $h : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$ where $h(q, t) = \varepsilon\text{-close}(\text{move}(q, t))$. In other words, given subset q, h computes the subset associated with transitioning on input t.

Now, our desired DFA is a fixpoint of $g : \mathcal{M} \to \mathcal{M}$ where $g(\bar{Q}, \Sigma, \bar{\delta}, \bar{q_0}, \bar{F}) = (\bar{Q} \cup \bar{Q}', \Sigma, \bar{\delta} \cup \bar{\delta}', \bar{q_0}, \bar{F} \cup \bar{F}')$ and:

- $\bar{Q}' = \{h(q, t) \mid q \in \bar{Q}, t \in \Sigma\}$,
- $\bar{\delta}' = \{(q, t, h(q, t)) \mid q \in \bar{Q}, t \in \Sigma\}$,
- $\bar{F}' = \{q \mid q \in \bar{Q}', q \cap F \neq \emptyset\}$.

Specifically, if $\bar{q_0} = \varepsilon\text{-close}(q_0)$ then our DFA is the fixpoint of g where $x_0 = (\{\bar{q_0}\}, \Sigma, \emptyset, \bar{q_0}, \{q \mid q = \bar{q_0}, q \cap F \neq \emptyset\})$.

Observation. Notice the similarities between reachability's g and the powerset construction's g.

When implementing the `nfa_to_dfa` function, we recommend the following division of labor:

- `nfa_to_dfa` calls `fix` to find the appropriate fixpoint of `step_dfa`.
- `step_dfa` (analogous to g above) computes one step of the powerset construction algorithm (i.e. applies `step_state` over all states in the DFA).
- `step_state` computes one step of the powerset construction on a given state (i.e. applies `step_symbol` over all symbols in Σ).
- `step_symbol` adds new subset, transition, and possibly final state, given a state and input symbol (i.e. applies `step` and unions into DFA).
- `step` is equivalent to the h described above.

Remark. We provide a correct implementation of the set functions from P2A and some additional utility functions. You should use them.