1 Set Operators

Last time we considered a set of *rule instances* of the form

$$\frac{x_1\ x_2\ \ldots\ x_n}{x}, \tag{1}$$

where x and the x_i are members of some set S. From this we defined a set operator

$$R(B) \triangleq \{x \mid \{x_1, \ldots, x_n\} \subseteq B \text{ and } \frac{x_1\ x_2\ \ldots\ x_n}{x} \text{ is a rule instance}\}. \tag{2}$$

An important property of set operators defined by rules of this form is *monotonicity*. A set operator R is *monotone* if $B \subseteq C$ implies $R(B) \subseteq R(C)$. We listed two desirable properties of the set $A \subseteq S$ defined by R, namely:

- A should be R-*closed*: $R(A) \subseteq A$. We would like this to hold because we would like every element that the rules say should be in A to actually be in A.
- A should be R-*consistent*: $A \subseteq R(A)$. We would like this to hold because we would like every element of A to be included in A *only* as a result of applying a rule.

These two properties together say that $A = R(A)$, or in other words, A should be a *fixpoint* of R. We then asked two natural questions:

- Does R actually have a fixpoint?
- Is the fixpoint unique? If not, which one should we take?

2 Least Fixpoints

In fact, any monotone set operator R always has at least one fixpoint. It may have many, but among all its fixpoints, it has a unique minimal one with respect to set inclusion $\subseteq$; that is, a fixpoint that is a subset of all other fixpoints of R. We call this the *least fixpoint* of R. It also has a *greatest fixpoint*, which is a fixpoint that is a superset of all other fixpoints of R.

The least fixpoint of R can be defined in two different ways, "from below" and "from above". For the rule-based operators we have been studying, these constructions take the following form:

$$A_* \triangleq \bigcup_{n \geq 0} R^n(\varnothing) \ = \ R(\varnothing) \cup R(R(\varnothing)) \cup R(R(R(\varnothing))) \cup \cdots \tag{3}$$

$$A^* \triangleq \bigcap \{B \subseteq S \mid R(B) \subseteq B\}. \tag{4}$$

The set A_* is the union of all sets of the form $R^n(\varnothing)$, the sets obtained by applying R some finite number of times to the empty set. The set A^* is the intersection of all the R-closed subsets of S. We will show that for our rule-based systems, $A_* = A^*$ and that this set is the least fixpoint of R.

3 The Knaster–Tarski Theorem

The fact that $A_* = A^*$ is a special case of a more general theorem called the *Knaster–Tarski theorem*. It states that any monotone set operator R has a unique least fixpoint, and that this fixpoint can be obtained either "from below" by iteratively applying R to the empty set, or "from above" by taking the intersection of all R-closed sets.

For general monotone operators R, iterating through the finite ordinals in the "from below" construction may not be enough; it may require iteration through transfinite ordinals. However, the operators R defined from rule systems as described above are *chain-continuous* (definition below). This is a stronger property than monotonicity. It guarantees that the "from below" construction converges to a fixpoint after only ω steps, where ω is the first transfinite ordinal.

3.1 Monotone, Continuous, and Finitary Operators

If $\mathcal{C}$ is a set of subsets of S, denote by $\bigcup \mathcal{C}$ the union of all elements of $\mathcal{C}$; that is, $x \in \bigcup \mathcal{C}$ iff there exists $A \in \mathcal{C}$ such that $x \in A$. The set $\bigcup \mathcal{C}$ is the smallest subset of S containing all $A \in \mathcal{C}$ as subsets. A *chain* is a set $\mathcal{C}$ of subsets of S that is linearly ordered by the set inclusion relation $\subseteq$; that is, for all $B, C \in \mathcal{C}$, either $B \subseteq C$ or $C \subseteq B$.

A set operator $R : 2^S \to 2^S$ is said to be

- *monotone* if $B \subseteq C$ implies $R(B) \subseteq R(C)$;
- (*chain-*)*continuous* if for any chain of sets $\mathcal{C}$,
$$R(\bigcup \mathcal{C}) = \bigcup \{R(B) \mid B \in \mathcal{C}\};$$
- *finitary* if for any set C, the value of $R(C)$ is determined by the values of $R(B)$ for finite subsets $B \subseteq C$ in the following sense:
$$R(C) = \bigcup \{R(B) \mid B \subseteq C,\ B \text{ finite}\}.$$

Lemma 7.1.

(i) *Every rule-based operator of the form* (2) *is finitary.*

(ii) *Every finitary operator is chain-continuous.*

(iii) *Every chain-continuous operator is monotone.*

The proof of Lemma 7.1 is fairly straightforward and we will leave it as an exercise. In fact, the converse of (ii) holds as well—every chain-continuous operator is finitary—but the proof requires transfinite induction and is more difficult.

3.2 Proof of the Knaster–Tarski Theorem for Chain-Continuous Operators

Let us prove the Knaster–Tarski theorem in the special case of chain-continuous operators, which will allow us to avoid introducing transfinite ordinals (not that they are not worth introducing!), and that is all we need to handle rule-based inductive definitions.

Theorem 7.2 (Knaster–Tarski theorem)**.** *Let $R : 2^S \to 2^S$ be a chain-continuous set operator. Let A_* and A^* be defined as in (3) and (4), respectively. Then $A_* = A^*$, and this is the $\subseteq$-least fixpoint of R. That is,*

- *it is a fixpoint of R: $R(A_*) = A_*$; and*
- *it is the $\subseteq$-least fixpoint: if $R(B) = B$, then $A_* \subseteq B$.*

Proof. The theorem will follow from two observations:

(i) For every n and every R-closed set B, $R^n(\varnothing) \subseteq B$. This can be proved by induction on n. It follows that $A_* \subseteq A^*$.

(ii) A_* is R-closed. Since A^* is contained in all R-closed sets, $A^* \subseteq A_*$.

For (i), let B be an R-closed set. We proceed by induction on n. The basis for $n = 0$ is $\varnothing \subseteq B$, which is trivially true. Now suppose $R^n(\varnothing) \subseteq B$. We have

$$\begin{aligned} R^{n+1}(\varnothing) &= R(R^n(\varnothing)) \\ &\subseteq R(B) && \text{by the induction hypothesis and monotonicity} \\ &\subseteq B && \text{since } B \text{ is } R\text{-closed.} \end{aligned}$$

We conclude that for all n and all R-closed sets B, $R^n(\varnothing) \subseteq B$, therefore

$$A_* \;=\; \bigcup \{R^n(\varnothing) \mid n \geq 0\} \;\subseteq\; \bigcap \{B \subseteq S \mid R(B) \subseteq B\} \;=\; A^*.$$

For (ii), we want to show that $R(A_*) \subseteq A_*$. It can be proved by induction on n that the sets $R^n(\varnothing)$ form a chain:

$$\varnothing \;\subseteq\; R(\varnothing) \;\subseteq\; R^2(\varnothing) \;\subseteq\; R^3(\varnothing) \;\subseteq\; \cdots$$

We have $\varnothing \subseteq R(\varnothing)$ trivially, and by monotonicity, if $R^n(\varnothing) \subseteq R^{n+1}(\varnothing)$, then

$$R^{n+1}(\varnothing) \;=\; R(R^n(\varnothing)) \;\subseteq\; R(R^{n+1}(\varnothing)) \;=\; R^{n+2}(\varnothing).$$

Now by chain-continuity,

$$R(A_*) \;=\; R(\bigcup_{n\geq 0} R^n(\varnothing)) \;=\; \bigcup_{n\geq 0} R(R^n(\varnothing)) \;=\; \bigcup_{n\geq 0} R^{n+1}(\varnothing) \;=\; A_*. \qquad \square$$

4 Rule Induction

Let us use our newfound wisdom on well-founded induction and least fixpoints of monotone maps to prove some properties of the reduction rules.

4.1 Example: CBV Preserves Closedness

Theorem 7.3. *If $e \to e'$ under the CBV reduction rules, then $FV(e') \subseteq FV(e)$. In other words, CBV reductions cannot introduce any new free variables.*

Proof. By induction on the CBV derivation of $e \to e'$. There is one case for each CBV rule, corresponding to each way $e \to e'$ could be derived.

Case 1 $\dfrac{e_1 \to e_1'}{e_1\, e_2 \to e_1'\, e_2}$.

We assume that the desired property is true of the premise—this is the induction hypothesis—and we wish to prove under this assumption that it is true for the conclusion. Thus we are assuming that $FV(e_1') \subseteq FV(e_1)$ and wish to prove that $FV(e_1'\, e_2) \subseteq FV(e_1\, e_2)$.

$$\begin{aligned} FV(e_1'\, e_2) &= FV(e_1') \cup FV(e_2) && \text{by the definition of } FV \\ &\subseteq FV(e_1) \cup FV(e_2) && \text{by the induction hypothesis} \\ &= FV(e_1\, e_2) && \text{again by the definition of } FV. \end{aligned}$$

Case 2 $\dfrac{e_2 \to e_2'}{v\, e_2 \to v\, e_2'}$.

This case is similar to Case 1, where now $e_2 \to e_2'$ is used in the induction hypothesis.

Case 3 $\dfrac{}{(\lambda x.\, e)\, v \to e\{v/x\}}$.

There is no induction hypothesis for this case, since there is no premise in the rule; thus this case constitutes the basis of our induction. We wish to show, independently of any inductive assumption, that $FV(e\{v/x\}) \subseteq FV((\lambda x.\, e)\, v)$.

This case requires a lemma, stated below, to show that $FV(e\{v/x\}) \subseteq (FV(e) - \{x\}) \cup FV(v)$. Once that is shown, we have

$$\begin{aligned} FV(e\{v/x\}) &\subseteq (FV(e) - \{x\}) \cup FV(v) && \text{by the lemma to be proved} \\ &= FV(\lambda x.\, e) \cup FV(v) && \text{definition of } FV \\ &= FV((\lambda x.\, e)\, v) && \text{definition of } FV. \end{aligned}$$

We have now considered all three rules of derivation for the CBV λ-calculus, so the theorem is proved. □

The following lemma is used in Case 3 of Theorem 7.3 above.

Lemma 7.4. *$FV(e\{v/x\}) \subseteq (FV(e) - \{x\}) \cup FV(v)$.*

Proof. By structural induction on e. There is one case for each clause in the definition of the substitution operator. We have assumed previously that values are closed terms, so $FV(v) = \varnothing$ for any value v; but actually we do not need this for the proof, and we do not assume it.

Case 1 $e = x$.

$$\begin{aligned} FV(x\{v/x\}) &= FV(v) && \text{definition of substitution} \\ &= (\{x\} - \{x\}) \cup FV(v) \\ &= (FV(x) - \{x\}) \cup FV(v) && \text{definition of } FV. \end{aligned}$$

Case 2 $e = y$, $y \neq x$.

$$\begin{aligned} FV(y\{v/x\}) &= FV(y) && \text{definition of substitution} \\ &= \{y\} && \text{definition of } FV \\ &\subseteq (\{y\} - \{x\}) \cup FV(v) \\ &= (FV(y) - \{x\}) \cup FV(v) && \text{definition of } FV. \end{aligned}$$

Case 3 $e = e_1\, e_2$.

$$\begin{aligned} FV((e_1\, e_2)\{v/x\}) &= FV(e_1\{v/x\}\, e_2\{v/x\}) && \text{definition of substitution} \\ &\subseteq (FV(e_1) - \{x\}) \cup FV(v) \cup (FV(e_2) - \{x\}) \cup FV(v) && \text{induction hypothesis} \\ &= ((FV(e_1) \cup FV(e_2)) - \{x\}) \cup FV(v) \\ &= (FV(e_1\, e_2) - \{x\}) \cup FV(v) && \text{definition of } FV. \end{aligned}$$

Case 4 $e = \lambda x.\, e'$.

$$\begin{aligned} FV((\lambda x.\, e')\{v/x\}) &= FV(\lambda x.\, e') && \text{definition of substitution} \\ &= FV(\lambda x.\, e') - \{x\} && \text{because } x \notin FV(\lambda x.\, e') \\ &\subseteq (FV(\lambda x.\, e') - \{x\}) \cup FV(v). \end{aligned}$$

Case 5 $e = \lambda y.\, e'$, $y \neq x$. This is the most interesting case, because it involves a change of bound variable. Using the fact $FV(v) = \varnothing$ for values v would give a slightly simpler proof. Let v be a value and z a variable such that $z \neq x$, $z \notin FV(e')$, and $z \notin FV(v)$.

$$\begin{aligned} FV((\lambda y.\, e')\{v/x\}) &= FV(\lambda z.\, e'\{z/y\}\{v/x\}) && \text{definition of substitution} \\ &= FV(e'\{z/y\}\{v/x\}) - \{z\} && \text{definition of } FV \\ &= ((((FV(e') - \{y\}) \cup FV(z)) - \{x\}) \cup FV(v)) - \{z\} && \text{induction hypothesis (twice)} \\ &= (((FV(\lambda y.\, e') \cup \{z\}) - \{x\}) \cup FV(v)) - \{z\} && \text{definition of } FV \\ &= ((FV(\lambda y.\, e') - \{x\}) \cup FV(v) \cup \{z\}) - \{z\} \\ &= (FV(\lambda y.\, e') - \{x\}) \cup FV(v). \end{aligned}$$

□

There is a subtle point that arises in Case 5. We said at the beginning of the proof that we would be doing structural induction on e; that is, induction on the well-founded subterm relation $<$. This was a lie. Because of the change of bound variable necessary in Case 5, we are actually doing induction on the relation of subterm modulo α-equivalence:

$$e <_\alpha e' \triangleq \exists e''\ e'' < e' \wedge e =_\alpha e''.$$

But a moment's thought reveals that this relation is still well-founded, since α-reduction does not change the size or shape of the term, so we are ok.

5 Remark

These proofs may seem rather tedious, and one may wonder why we are doing them in such detail. But of course, this is exactly the point. Formal reasoning about the semantics of the λ-calculus, including such seemingly complicated notions as reductions and substitutions, can be reduced to the mindless application of a few simple rules. There is no hand-waving, no magic, no hidden behavior. To the extent that we can do this for real programming languages, we will be better able to understand and predict their behavior and to automate parts of the reasoning process.