1 Set operators

A set of inference rules generate *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. The *premises* are $X_1, X_2, \ldots, X_n$ and the *conclusion* is X. We use such rule instances to specify an inductively defined subset $A \subseteq S$; intuitively, the rule (1) says that if you have $X_1, \ldots, X_n \in A$, then you must also take $X \in A$.

Note that since an inductively defined set is defined by the rule instances, we are free to introduce metanotation in rules as long as it is clear what rule instances are generated.

Formally, given a set of rules and a subset $B \subseteq S$, define a rule operator R that captures the inferences that can be made from the rule instances:

$$R(B) \triangleq \{X \mid \{X_1, X_2, \ldots, X_n\} \subseteq B \text{ and } \frac{X_1\ X_2\ \ldots\ X_n}{X} \text{ is a rule instance}\}. \tag{2}$$

Then R is a function mapping subsets of S to subsets of S; that is, $R : 2^S \to 2^S$, where 2^S denotes the *powerset* (set of all subsets) of S. An important property of R is that it is clearly *monotone*: if $B \subseteq C$, then $R(B) \subseteq R(C)$.

What set $A \subseteq S$ is defined by the rules? At the very least, we would like A to satisfy the following two properties:

- A is *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.
- A is *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.

These two properties together say that $A = R(A)$, or in other words, A should be a fixed point of R. There are two natural questions to ask:

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

2 Least fixed points

In fact, any monotone set operator R always has at least one fixed point. It may have many. But among all its fixed points, it has a unique minimal one with respect to set inclusion $\subseteq$; that is, a fixed point that is a subset of all other fixed points of R. We call this the *least fixed point* of R, and we take A to be this set.

The least fixed point of R can be defined in two different ways, "from below" and "from above":

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

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

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. It consists of all elements of S that are in $R^n(\varnothing)$ for some $n \geq 0$.

The set A^* is the intersection of all the R-closed subsets of S. It consists of all elements of S that are in every R-closed set.

We will show that $A_* = A^*$ and that this set is the least fixed point of R, so we will take $A \triangleq A_* = A^*$.

3 Proof

Obviously, $\varnothing \subseteq R(\varnothing)$. Since R is monotone, we can apply it to both sides, obtaining $R(\varnothing) \subseteq R^2(\varnothing)$. Clearly $R^n(\varnothing) \subseteq R^{n+1}(\varnothing)$ for all n. So we can see that the successive sets $R^n(\varnothing)$ in the union defining A_* are in a chain with respect to $\subseteq$:

$$\varnothing \subseteq R(\varnothing) \subseteq R^2(\varnothing) \subseteq R^3(\varnothing) \subseteq R^4(\varnothing) \subseteq \ldots$$

3.1 Closure

First, we show that A_* is R-closed: i.e., $R(A_*) \subseteq A_*$. Consider an arbitrary $x \in R(A_*)$. It must be there is a rule $\frac{x_1,\ldots x_n}{x}$ where all of the x_i are in A_*. Each of x_i must be found first in one of the sets $R^m(\varnothing)$. Since there are a finite number of premises (n), there must be a set $R^m(\varnothing)$ in the chain that includes all of the x_i. But then $R^{m+1}(\varnothing)$ must include x, so therefore so does A_*. Since this is true for arbitrary $x \in R(A_*)$, we have $R(A_*) \subseteq A_*$.

3.2 Consistency

We need to show that $A_* \subseteq R(A_*)$. Consider an arbitrary $x \in A_*$. We must have $x \in R^m(\varnothing)$ for some m. In this case, we know that all its premises x_i are found in $R^{m-1}(\varnothing)$. Since $R^{m-1}(\varnothing) \subseteq A_*$, we can apply R to both sides to obtain $R^m(\varnothing) \subseteq R(A_*)$. Therefore x must be in $R(A_*)$, and so A_* is R-consistent.

3.3 Least fixed point

Suppose we have another fixed point of R. Call it B. Given $B = R(B)$, we want to show that $A_* \subseteq B$. Clearly we have $\varnothing \subseteq B$. Applying R to both sides, we obtain $R(\varnothing) \subseteq R(B) = B$ We can repeat as necessary to obtain $R^n(\varnothing) \subseteq B$ (that is, this holds by induction on n). Therefore the union of all $R^n(\varnothing)$ cannot contain any elements not in B, so $A_* \subseteq B$. In fact, by the same argument, not only is A_* the least fixed point, it is the least R-consistent set.

4 Other fixed points

In general the rule operator will have other fixed points. These correspond to allowing some proof trees with infinite derivations. For example, consider these three rules over $\mathbb{Z}$:

$$\frac{0}{0} \quad \frac{n+1}{n} \quad \frac{}{1}$$

The least fixed point will consist of $\{n \mid n \geq 1\}$. The *greatest* fixed point will be $\mathbb{N}$, including 0, because $\{0, 1, \ldots\}$ is a fixed point of R, even though there is no finite proof tree deriving 0. The greatest fixed point is the union of all R-consistent sets.

However, we use the least fixed point because it ensures that all the proof trees are finite. Hence, the subderivation relation on proof trees is well-founded. This would not be the case with other fixed points.

5 Finiteness of premises

One importance feature of inference rules is that the number of premises is finite. We saw that was needed in the argument that A_* is R-closed. If inference rules have an infinite number of premises, A_* need not be closed! Consider these rules over $\mathbb{Z}$:

$$\frac{}{1} \quad \frac{n}{n+1} \quad \frac{m \quad {\scriptstyle (\forall m > n)}}{n}$$

In this case, the A_* construction gives us the set $\{1, 2, \ldots\}$, but $0 \in R(A_*)$.

6 Rule Induction

Let us use our newfound wisdom on well-founded induction and least fixed points of monotone maps to prove some properties of the semantics we have seen so far.

6.1 Example 1: evaluation preserves closedness

Theorem 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 $FV(e_1'\ e_2) \subseteq FV(e_1\ e_2)$.

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

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

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

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

Lemma $FV(e\{v/x\}) \subseteq (FV(e) - \{x\}) \cup FV(v)$ (this lemma is used by case 3 in the above theorem).

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: $e = x$.

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

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

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

Case 3: $e = e_1\ e_2$.

$$\begin{aligned}
FV(e\{v/x\}) &= FV((e_1\ e_2)\{v/x\}) \\
&= FV(e_1\{v/x\}\ e_2\{v/x\}) \quad \text{by the definition of the substitution operator} \\
&\subseteq (FV(e_1) - \{x\}) \cup FV(v) \cup (FV(e_2) - \{x\}) \cup FV(v) \quad \text{by the induction hypothesis} \\
&= ((FV(e_1) \cup FV(e_2)) - \{x\}) \cup FV(v) \\
&= (FV(e_1\ e_2) - \{x\}) \cup FV(v) \quad \text{again by the definition of } FV \\
&= (FV(e) - \{x\}) \cup FV(v).
\end{aligned}$$

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

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

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 $z \notin FV(v)$.

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

□

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:

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

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.

6.2 Example 2: agreement of big-step and small-step semantics

As we saw earlier, we can express the idea that the two semantics should agree on terminating executions by connecting the $\longrightarrow^*$ and $\Downarrow$ relations:

$$\langle c, \sigma\rangle \longrightarrow^* \langle \mathbf{skip}, \sigma'\rangle \iff \langle c, \sigma\rangle \Downarrow \sigma'$$

This can be proved using induction. To prove the $\Rightarrow$ direction, we can use structural induction on c. The $\Leftarrow$ direction requires induction on the derivation of the big-step evaluation. We are given $\langle c, \sigma\rangle \Downarrow \sigma'$, so we know that there is a derivation. The form of the derivation depends on the form of c. Here we show just a few of the cases for c.

- Case **skip**. In this case we know $\sigma = \sigma'$, and trivially, $\langle \mathbf{skip}, \sigma\rangle \longrightarrow^* \langle \mathbf{skip}, \sigma\rangle$.
- Case $x := a$. In this case we know from the premises that $\langle a, \sigma\rangle \Downarrow n$ for some n, and that $\sigma' = \sigma[x \mapsto n]$.

  We will need a lemma that $\langle a, \sigma\rangle \Downarrow n \Rightarrow \langle a, \sigma\rangle \longrightarrow^* n$. This can be proved using the same technique being used on commands. We will also need a lemma showing that $\langle a, \sigma\rangle \longrightarrow^* n$ implies $\langle x := a, \sigma\rangle \longrightarrow^* \langle x := n, \sigma\rangle$. This obvious result can be proved easily using an induction on the number of steps taken.

  Given these lemmas, we have $\langle x := a, \sigma\rangle \longrightarrow^* \langle x := n, \sigma\rangle$ and $\langle x := n, \sigma\rangle \longrightarrow \langle \mathbf{skip}, \sigma[x \mapsto n]\rangle$, so $\langle x := a, \sigma\rangle \longrightarrow^* \langle \mathbf{skip}, \sigma[x \mapsto n]\rangle$.
- Case **while** b **do** c, where b evaluates to *false*. In this case we have $\langle b, \sigma\rangle \Downarrow \mathit{false}$ and $\sigma = \sigma'$. Consider what happens in the small-steps semantics. Given two more lemmas, that $\langle b, \sigma\rangle \Downarrow t \Rightarrow \langle b, \sigma\rangle \longrightarrow^* t$, and that $\langle b, \sigma\rangle \longrightarrow^* t \Rightarrow \langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma\rangle \longrightarrow^* \langle \mathbf{while}\ t\ \mathbf{do}\ c, \sigma\rangle$, we have $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma\rangle \longrightarrow^* \langle \mathbf{skip}, \sigma\rangle$, as desired.
- Case **while** b **do** c, where b evaluates to *true*. This is the most interesting case in the whole proof. We need one more obvious lemma for stitching together small-step executions:

$$(\langle c_1, \sigma\rangle \longrightarrow^* \langle \mathbf{skip}, \sigma'\rangle \wedge \langle c_2, \sigma'\rangle \longrightarrow^* \langle \mathbf{skip}, \sigma''\rangle) \implies \langle c_1; c_2, \sigma\rangle \longrightarrow^* \langle \mathbf{skip}, \sigma''\rangle \tag{5}$$

  This can be proved by induction on the number of steps.

  Now, because $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma\rangle \Downarrow \langle \mathbf{skip}, \sigma'\rangle$, we know that $\langle c, \sigma\rangle \Downarrow \sigma''$ and $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma''\rangle \Downarrow \sigma'$. Further, because the derivations of these two assertions are subderivations of that for $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma\rangle$, the induction hypothesis gives us that $\langle c, \sigma\rangle \longrightarrow^* \langle \mathbf{skip}, \sigma''\rangle$ and that $\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma''\rangle \longrightarrow^* \sigma'$. Using Lemma 5, we have $\langle c; \mathbf{while}\ b\ \mathbf{do}\ c, \sigma\rangle \longrightarrow^* \langle \mathbf{skip}, \sigma'\rangle$.

  Now consider the small-step side. We have an initial step

$$\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma\rangle \longrightarrow \langle \mathbf{if}\ b\ \mathbf{then}\ (c; \mathbf{while}\ b\ \mathbf{do}\ c)\ \mathbf{else}\ \mathbf{skip}, \sigma\rangle$$

  From prior lemmas, we know this will step to $\langle c; \mathbf{while}\ b\ \mathbf{do}\ c, \sigma\rangle$, which we just showed will step to $\langle \mathbf{skip}, \sigma'\rangle$ as desired.

Notice that we could not have used structural induction for this proof, because the induction step involved relating an evaluation of the command **while** b **do** c to a different evaluation of the same command rather than to an evaluation of a subexpression.

7 Remark

The value of the reasoning framework we have set up is that formal reasoning about the semantics of programming languages, 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 or magic involved. There is nothing hidden, it is all right there in front of you. To the extent that we can do this for real programming languages, we will be better able to understand what is going on.

8 The Knaster–Tarski Theorem

This material is optional.

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 fixed point, and that this fixed point 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, the "from below" construction 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 fixed point after only ω steps, where ω is the first transfinite ordinal.

8.1 Monotone, continuous, and finitary operators

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 $\mathcal{C}$,

  $$R(\bigcup \mathcal{C}) \quad = \quad \bigcup \{R(B) \mid B \in \mathcal{C}\}$$

  (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 $C \subseteq B$);
- *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:

  $$R(C) \quad = \quad \bigcup \{R(B) \mid B \subseteq C,\ B \text{ finite}\}.$$

One can show

1. Every rule-based operator of the form (2) is finitary;
2. Every finitary operator is chain-continuous (in fact, the converse holds as well);
3. Every chain-continuous operator is monotone.

The proofs of 1, 2, and 3 are fairly straightforward and we will leave them as exercises. The converse of 2 requires transfinite induction and is more difficult.

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

Proof. The theorem follows 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 a fixed point of R, thus 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

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

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 $R(A_*) = A_*$. 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,

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

□