Notes on Inductive Definitions

Joonhyup Lee

1 What is an inductive definition?

Throughout this article, we always assume S to be a set, and Φ to be a subset of $\mathcal{P}(S) \times S$.

Definition 1.1 (Φ-closed). A set $A \subseteq S$ is Φ-closed iff:

$$\forall (X, x) \in \Phi, X \subseteq A \Rightarrow x \in A$$

Notation. We call $(X, x) \in \Phi$ *rules*, and we write:

$$\Phi : X \to x \triangleq (X, x) \in \Phi$$

Definition 1.2 (Inductive definition). The set $I(\Phi)$ that is *inductively defined* by Φ is the least Φ-closed set. That is:

$$I(\Phi) \triangleq \bigcap \{A | A \text{ is } \Phi\text{-closed}\}$$

Remark. This definition makes sense, since:

1. Φ-closed sets exist: $\bigcup_X \{x | (X, x) \in \Phi\}$ is obviously Φ-closed.
2. For a collection of Φ-closed sets $\{A_i\}_{i \in I}$, $\bigcap_{i \in I} A_i$ is Φ-closed.

Example. A classical example of inductive definitions is the *syntax* of programming languages. For instance, the BNF grammar for the *untyped λ-calculus* is:

$$e \to x \mid \lambda x.e \mid e\ e$$

This is actually a definition for the rule set in some base set S, where S is a set large enough to contain all λ-expressions. For example, when we assume a set of variables $\mathbb{V}$, we can define the set of symbols $\mathbb{S}$ and the base set S as:

$$\mathbb{S} \triangleq \mathbb{V} \cup \{\lambda,\ .,\ \sqcup\} \quad S \triangleq \{\text{words generated by symbols in } \mathbb{S}\}$$

The rule set that is expressed by the BNF grammar is:

$$\Phi \triangleq \{\emptyset \to x | x \in \mathbb{V}\} \cup \{\{e\} \to \lambda x.e | x \in \mathbb{V}\} \cup \{\{e_1, e_2\} \to e_1\ e_2 | \mathsf{True}\}$$

and the set Expr of λ-expressions is:

$$\text{Expr} \triangleq I(\Phi)$$

Example. Let us look at another example: the big-step evaluation relation for λ-expressions.

First, we need to define our set S.

$$\text{Env} \triangleq \mathbb{V} \xrightarrow{\text{fin}} \text{Val} \quad \text{Val} \triangleq \text{Expr} \times \text{Env} \quad S \triangleq \text{Val} \times \text{Val}$$

Env and Val are mutually recursive and should be defined inductively, but we ignore this for now.

Next, we need to define the set $\Phi \subseteq \mathcal{P}(S) \times S$ that inductively defines our relation.

$$\frac{\sigma(x) = v}{((x, \sigma), v)}\,\textsc{Var} \qquad \frac{}{((\lambda x.e, \sigma), (\lambda x.e, \sigma))}\,\textsc{Lam} \qquad \frac{((e_1, \sigma), (\lambda x.e, \sigma_1)) \quad ((e_2, \sigma), v_2) \quad ((e, \sigma_1[x \mapsto v_2]), v)}{((e_1\, e_2, \sigma), v)}\,\textsc{App}$$

The traditional way to present Φ is through the form of *inference rules*, as displayed above. This is a way of expressing the set of rules Φ defined as:

$$\begin{aligned}\Phi \triangleq\ & \{\emptyset \to ((x, \sigma), v) | \sigma(x) = v\} \\ \cup\ & \{\emptyset \to ((\lambda x.e, \sigma), (\lambda x.e, \sigma)) | \mathsf{True}\} \\ \cup\ & \{\{((e_1, \sigma), (\lambda x.e, \sigma_1)), ((e_2, \sigma), v_2), ((e, \sigma_1[x \mapsto v_2]), v)\} \to ((e_1\, e_2, \sigma), v) | \mathsf{True}\}\end{aligned}$$

One thing to note is that the rule set may contain elements that are nonsensical:

$$\{((x, []), (\lambda x.x, [])), ((x, []), (x, [])), ((x, [x \mapsto (x, [])]), (x, []))\} \to ((x\, x, []), (x, []))$$

is contained in Φ, according to the definition above.

However, since the big-step evaluation relation is defined as the *least* set that is closed under Φ, the defined relation makes sense.

2 What is a proof tree?

In this section, we aim to connect the notion of *proof trees* with inductive definitions.

Definition 2.1 (Well-founded tree). A *well-founded tree* T is a set of finite sequences of positive length such that:

1. There exists a unique sequence of length 1. We call it the *root* (a_T) of the tree.
2. T is prefix-closed. That is, if $(a_1, \dots, a_{n+1}) \in T$, then $(a_1, \dots, a_n) \in T$.
3. The relation $<_T$ defined by:

   $$(a_1, \dots, a_n) <_T (b_1, \dots, b_m) \triangleq n = m + 1 \text{ and } a_i = b_i \text{ for } i = 1, \dots, m$$

   is well-founded (there is no infinitely decreasing sequence).

   That is, there is no infinite sequence $(b_i)_{i \in \mathbb{N}}$ such that its prefix $(b_i)_{i \leq n}$ is in T for each $n > 0$.

Definition 2.2 (Tree Φ-proof). A well-founded tree T is a *tree Φ-proof* of $a \in S$ when:

1. (a) is the root of the tree. That is, $a = a_T$.
2. For each $(a_1, \dots, a_n) \in T$, $\Phi : T_{(a_1, \dots, a_n)} \to a_n$, where

   $$T_{(a_1, \dots, a_n)} \triangleq \{a | (a_1, \dots, a_n, a) \in T\}$$

Naturally, we expect the following to hold:

Claim 2.1 (Proof trees, infinite version).

$$I(\Phi) = \{a_T | T \text{ is a tree } \Phi\text{-proof}\}$$

This claim holds, however the proof requires the axiom of dependent choice and transfinite induction. If we restrict Φ to be *finitary*, we can prove a weakened, yet still useful version of the above claim.

Definition 2.3 (Finitary rule set). Φ is called *finitary* when for each $(X, x) \in \Phi$, X is finite.

Definition 2.4 (Finite tree). A *finite tree* T is a set of finite sequences of positive length such that:

1. There exists a unique sequence of length 1. We call it the *root* (a_T) of the tree.
2. T is prefix-closed. That is, if $(a_1, \dots, a_{n+1}) \in T$, then $(a_1, \dots, a_n) \in T$.
3. T is finite.

Naturally, finite trees are also well-founded, so the following definition makes sense:

Definition 2.5 (Finite tree Φ-proof). A finite tree T is a *finite tree Φ-proof* of $a \in S$ when it is a tree Φ-proof of a.

For finite trees, we may define the *level* of the tree.

Definition 2.6 (Level of a finite tree). We define the *level* of a finite tree T to be the length of the longest sequence in T:

$$\text{level}(T) \triangleq \max\{n | (a_1, \dots, a_n) \in T\}$$

Remark (Proof methods). Before going into the proof, we must review methods of proof.

1. (Φ-induction) The set $I(\Phi)$ is the least Φ-closed set.

   Therefore, to prove that some set A contains $I(\Phi)$, we can prove that A is Φ-closed.

2. (Induction on the number of nodes in T) To prove a formula of the form

   $$\forall T \in \{\text{finite trees}\}, P(T)$$

   we can perform induction on $|T|$, which is the *number of nodes in* T.

   $|T|$ is the number of nodes, since each path from the root to a node uniquely determines the node. That is, we utilize:

   $$[\forall T \in \{\text{finite trees}\}, P(T)] \Leftrightarrow [\forall n \in \mathbb{N}, T \in \{\text{finite trees}\}, |T| \leq n \Rightarrow P(T)]$$

3. (Induction on the level of T) Likewise, we may also perform induction on the level of T:

   $$[\forall T \in \{\text{finite trees}\}, P(T)] \Leftrightarrow [\forall n \in \mathbb{N}, T \in \{\text{finite trees}\}, \text{level}(T) \leq n \Rightarrow P(T)]$$

Now we can prove the following:

Theorem 2.1 (Proof tree, finite version). For finitary Φ,

$$I(\Phi) = \{a_T | T \text{ is a finite tree } \Phi\text{-proof}\}$$

Proof. We prove both sides of the inclusion.
($\subseteq$) We show that the right hand side is Φ-closed. That is:

$$\forall (X, x) \in \Phi, X \subseteq \{a_T | T \text{ is a finite tree } \Phi\text{-proof}\} \Rightarrow x \in \{a_T | T \text{ is a finite tree } \Phi\text{-proof}\}$$

Fix a $(X, x) \in \Phi$. Since Φ is finitary, we may enumerate the elements of X as $x_1, \dots, x_n$.
By assumption, we can pick finite tree Φ-proofs $T_1, \dots, T_n$ such that the root of T_i is (x_i).
Now, we let:

$$T \triangleq \{(x)\} \cup \bigcup_{1 \leq i \leq n} \{(x, a_1, \dots, a_m) | (a_1, \dots, a_m) \in T_i\}$$

T is a finite tree, since it has a unique root (x), is prefix-closed, and is finite.
It is a finite tree Φ-proof, since for each $(b_1, \dots, b_k) \in T$, $\Phi : T_{(b_1, \dots, b_k)} \to b_k$.
Thus, since $x = a_T$, we have shown that $x \in \{a_T | T \text{ is a finite tree } \Phi\text{-proof}\}$.

($\supseteq$) We want to prove that:

$$\forall T \in \{\text{finite trees}\}, T \text{ is a finite tree } \Phi\text{-proof} \Rightarrow a_T \in I(\Phi)$$

We perform induction on the level of T.
For level 1, for (a_T) to be a finite tree Φ-proof, $\Phi : \emptyset \to a_T$ must hold.
Thus, for any Φ-closed set X, $a_T \in X$. Thus $a_T \in I(\Phi)$.
For the inductive step, assume that the statement holds for level $\leq k$, and prove for level $= k+1$.
Fix a finite tree Φ-proof T with level $k+1$, and let (r) be the root of T.
Since Φ is finitary, $T_{(r)}$ can be enumerated as $x_1, \dots, x_n$.
Now define:

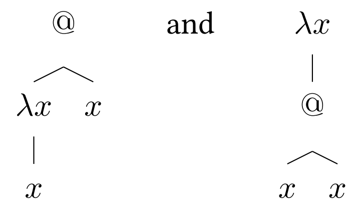
$$T_i \triangleq \{(a_1, \dots, a_m) | a_1 = x_i \text{ and } (r, a_1, \dots, a_m) \in T\}$$

We can show that T_i is a finite tree Φ-proof of x_i, and that level$(T_i) \leq k$.
Thus, $x_i \in I(\Phi)$ for each i by the inductive hypothesis.
Since $\Phi : \{x_1, \dots, x_n\} \to r$ and $I(\Phi)$ is Φ-closed, $r \in I(\Phi)$. $\square$

Example. The *abstract syntax* for lambda expressions is the *set* of tree Φ-proofs, more commonly known as *derivation trees* or *proof trees*. For example, the expression

$$\lambda x. x\ x$$

has two possible tree Φ-proofs as to why it is contained in Expr, namely:



Each of these trees are a distinct *abstract phrase*, which is an element of the abstract syntax. Actually, what Expr really denotes is not the set of concrete strings, but the abstract syntax.

3 Fixed point computation of proof trees

In this section, we aim to show how the set of proof trees that satisfy certain conditions may be computed as the least fixed point of some monotonic function.

Of special interest is the case when the base set S is of the form $A \times B$ for sets A and B. Given a finitary rule set Φ, we want to compute, for each $a \in A$, the set

$$[\![a]\!] \triangleq \{b \in B | (a, b) \in I(\Phi)\}$$

As can be deduced from the notation, characterizing $[\![a]\!]$ as a least fixed point is useful for the semantics of programming languages. For example, if we let $A = B = \mathsf{Val}$ in the example from section 1, $[\![(e, \sigma)]\!]$ becomes the set of values that the expression e can evaluate to under σ.

Thus, in this section we assume $S = A \times B$ for some A, B.

Definition 3.1 (Propagation function). $f_a \in \bigcup_b \{Y | \Phi : Y \to (a, b)\} \times \mathcal{P}(A \cup A \times B) \to \mathcal{P}(A)$ is a *propagation function* at $a \in A$ when:

1. f_a is monotonic in its second argument:

$$X \subseteq X' \Rightarrow f_a(Y, X) \subseteq f_a(Y, X')$$

2. $f_a(Y, X)$ must propagate when there is some premise $(a', b') \in Y$ that is not in X:

$$Y - X \neq \emptyset \Rightarrow f_a(Y, X) \cap (Y - X).1 \neq \emptyset$$

   where $Z.1 \triangleq \{a | (a, b) \in Z \text{ for some } b\}$.

The propagation function at a propagates all premisses a' such that $(a', \_)$ *might be needed* for the computation of $(a, \_)$.

Definition 3.2 (Incremental proof tree computation). $F \in \mathcal{P}(A \cup A \times B) \to \mathcal{P}(A \cup A \times B)$ is an *incremental proof tree computation* function when there is a family of propagation functions $(f_a)_{a \in A}$ satisfying:

$$F(X) = \bigcup_{a \in X} \left(\underbrace{\{(a, b) | \Phi : Y \to (a, b) \text{ for some } Y \subseteq X\}}_{\text{evaluate}} \cup \overbrace{\bigcup_{Y} f_a(Y, X)}^{\text{propagate}} \right)$$

for each $X \subseteq A \cup A \times B$.

Note that F is monotonic, since f_a is monotonic in its second argument.

We want to prove that, when F is an incremental proof tree computation function and $\mathsf{Init} \subseteq A$,

$$\mathrm{lfp}(\lambda X.F(X) \cup \mathsf{Init})$$

is a *sound and complete* representation of all proof trees for $\mathsf{Init} \times B$.

Theorem 3.1 (Soundness). Let F be an incremental proof tree computation function. For each finite tree Φ-proof T, $X \subseteq A \cup A \times B$, and $a \in X$, if:

1. X is a post-fixed point of F, and
2. T is a finite tree Φ-proof of (a, b) for some $b \in B$,

we have:

$$\forall p \in T, p \subseteq X$$

Proof. We use induction on the number of nodes in T.

When $|T| = 1$, $\Phi : \emptyset \to (a, b)$ for some b, so

$$(a, b) \in \bigcup_{a \in X} \{(a, b) | \Phi : Y \to (a, b) \text{ for some } Y \subseteq X\} \subseteq F(X) \subseteq X$$

For the inductive step, assume the statement for $|T| \leq k$, and prove for $|T| = k + 1$.

Fix F, X, (a, b) such that $a \in X$, and a finite tree Φ-proof T of (a, b).

Note that $f_a(Y, X) \subseteq F(X)$ for any $Y \in \bigcup_b \{Y | \Phi : Y \to (a, b)\}$, since $a \in X$.

Step 1. We can prove that $T_{((a,b))} \subseteq X$ by assuming $T_{((a,b))} - X \neq \emptyset$ and drawing a contradiction.

If $T_{((a,b))} - X \neq \emptyset$, since $T_{((a,b))} \in \bigcup_b \{Y | \Phi : Y \to (a, b)\}$, we have:

$$\text{There exists an } a' \in f_a(T_{((a,b))}, X) \subseteq F(X) \subseteq X \text{ such that } a' \in (T_{((a,b))} - X).1$$

by condition 2 of definition 3.1.

Now, pick a b' such that $(a', b') \in T_{((a,b))} - X$, and let

$$T' \triangleq \{(x_1, \dots, x_m) | x_1 = (a', b') \text{ and } ((a, b), x_1, \dots, x_m) \in T\}$$

T' is a finite tree Φ-proof of (a', b'), $a' \in X$, and $|T'| \leq k$, so by the inductive hypothesis:

$$(a', b') \in X$$

This is a contradiction, since $(a', b') \in T_{((a,b))} - X$.

Step 2. We can prove $T_{((a,b))}.1 \subseteq X$. Fix a $(a', b') \in T_{((a,b))}$. Then:

$$f_a(T_{((a,b))}, X - \{(a', b')\}) \cap \{(a', b')\}.1 \neq \emptyset$$

since $T_{((a,b))} \subseteq X$ and by condition 2 of definition 3.1.

Now, by monotonicity of f_a in its second argument (condition 1 of definition 3.1), we have:

$$a' \in f_a(T_{((a,b))}, X - \{(a', b')\}) \subseteq f_a(T_{((a,b))}, X) \subseteq F(X) \subseteq X$$

Step 3. Now we can show that for any $p \in T$, $p \subseteq X$.

Note that $(a, b) \in (\text{evaluation part of } F(X)) \subseteq X$, since $T_{((a,b))} \subseteq X$ and $a \in X$.

If $p = ((a, b))$, $p = ((a, b)) \subseteq X$, since $(a, b) \in X$.

If $p = ((a, b), y_1, \dots, y_m)$, $y_1 = (a', b')$ for some $a' \in T_{((a,b))}.1 \subseteq X$.

Letting T' as above and using the inductive hypothesis, we have that $(y_i) \subseteq X$. □

Theorem 3.2 (Completeness). F is an incremental proof tree computation function. For $\mathsf{Init} \subseteq A$,

$$\text{lfp}(\lambda X.F(X) \cup \mathsf{Init}) \subseteq I(\Phi) \cup A$$

Proof. We prove that $I(\Phi) \cup A$ is a post-fixed point of $\lambda X.F(X) \cup \mathsf{Init}$.

Note that $F(I(\Phi) \cup A)$ can be divided into the evaluation and the propagation part.

Since the propagation part is trivially included in A, we only have to prove:

$$\bigcup_{a \in A} \{(a, b) | \Phi : Y \to (a, b) \text{ for some } Y \subseteq I(\Phi)\} \subseteq I(\Phi)$$

This is true, since $I(\Phi)$ is Φ-closed. $\square$

Corollary 3.1 (Sound and complete incremental computation). F is an incremental proof tree computation function. For $\mathsf{Init} \subseteq A$,

$$\forall a \in \mathsf{Init}, b \in B, (a, b) \in \text{lfp}(\lambda X.F(X) \cup \mathsf{Init}) \Leftrightarrow (a, b) \in I(\Phi)$$

Proof. From completeness, we have $(\Rightarrow)$. From soundness, we have $(\Leftarrow)$. $\square$

Example. For the λ-calculus example,

$$\begin{aligned}
f_{(x,\sigma)}(\emptyset, X) &\triangleq \emptyset \\
f_{(\lambda x.e,\sigma)}(\emptyset, X) &\triangleq \emptyset \\
f_{(e_1\ e_2,\sigma)}(\{((e_1, \sigma), (\lambda x.e, \sigma_1)), ((e_2, \sigma), v_2), ((e, \sigma_1[x \mapsto v_2]), v)\}, X)& \\
\triangleq \{(e_1, \sigma), (e_2, \sigma)\} \cup \bigcup_{((e_1,\sigma),(\lambda x.e,\sigma_1)) \in X} \bigcup_{((e_2,\sigma),v_2) \in X} &\{(e, \sigma_1[x \mapsto v_2])\}
\end{aligned}$$

is a propagation function. Thus,

$$\begin{aligned}
F(X) \triangleq & \bigcup_{(x,\sigma) \in X} \{((x, \sigma), v) | \sigma(x) = v\} \cup \bigcup_{(\lambda x.e,\sigma) \in X} \{((\lambda x.e, \sigma), (\lambda x.e, \sigma))\} \\
\cup & \bigcup_{(e_1\ e_2,\sigma) \in X} \left(\{(e_1, \sigma), (e_2, \sigma)\} \cup \bigcup_{((e_1,\sigma),(\lambda x.e,\sigma_1)) \in X} \bigcup_{((e_2,\sigma),v_2) \in X} \{(e, \sigma_1[x \mapsto v_2])\} \right) \\
\cup & \bigcup_{(e_1\ e_2,\sigma) \in X} \bigcup_{((e_1,\sigma),(\lambda x.e,\sigma_1)) \in X} \bigcup_{((e_2,\sigma),v_2) \in X} \bigcup_{((e,\sigma_1[x \mapsto v_2]),v) \in X} \{((e_1\ e_2, \sigma), v)\}
\end{aligned}$$

is an incremental proof tree computation function. We have that:

$$[\![(e, \sigma)]\!] = \{v | ((e, \sigma), v) \in \text{lfp}(\lambda X.F(X) \cup \{(e, \sigma)\})\}$$

References

[1] Aczel, P. An introduction to inductive definitions. In *HANDBOOK OF MATHEMATICAL LOGIC*, J. Barwise, Ed., vol. 90 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1977, pp. 739–782.