Chapter 21: Metatheory of Recursive Types

Induction and Coinduction
Finite and Infinite Types/Subtyping
Membership Checking

21.1 Induction and Coinduction
Universal Set $U$

- **Type:** a subset of $U$

Inductive/Coinductive Definition

$U$: everything in the world

Generating Function

- **Definition:** A function $F \in P(U) \rightarrow P(U)$ is **monotone** if $X \subseteq Y$ implies $F(X) \subseteq F(Y)$.

- **Definition:** Let $F$ be monotone, and $X$ be a subset of $U$.
  - $X$ is **F-closed** if $F(X) \subseteq X$.
  - $X$ is **F-consistent** if $X \subseteq F(X)$.
  - $X$ is a **fixed point** of $F$ if $F(X) = X$.
Exercise: Consider the following generating function on the three-element universe $U=\{a, b, c\}$:

- $E_1(\emptyset) = \{c\}$
- $E_1(\{a\}) = \{c\}$
- $E_1(\{b\}) = \{c\}$
- $E_1(\{c\}) = \{b, c\}$
- $E_1(\{a, b\}) = \{c\}$
- $E_1(\{a, c\}) = \{b, c\}$
- $E_1(\{b, c\}) = \{a, b, c\}$
- $E_1(\{a, b, c\}) = \{a, b, c\}$

Q: Which subset is $E_1$-closed, $E_1$-consistent?

---

Knaster-Tarski Theorem (1955)

**Theorem**

- The intersection of all $F$-closed sets is the least fixed point of $F$.
- The union of all $F$-consistent sets is the greatest fixed point of $F$.

**Definition**: The least fixed point of $F$ is written $\mu F$. The greatest fixed point of $F$ is written $\nu F$. 

---

2019/5/15
Exercise: Consider the following generating function on the three-element universe $U=\{a, b, c\}$:

$E_1(\emptyset) = \{c\}$
$E_1(\{a\}) = \{c\}$
$E_1(\{b\}) = \{c\}$
$E_1(\{c\}) = \{b, c\}$
$E_1(\{a, b\}) = \{c\}$
$E_1(\{a, c\}) = \{b, c\}$
$E_1(\{b, c\}) = \{a, b, c\}$
$E_1(\{a, b, c\}) = \{a, b, c\}$

Q: What are $\mu_{E_1}$ and $\nu_{E_1}$?

Exercise: Suppose a generating function $E_2$ on the universe $\{a, b, c\}$ is defined by the following inference rules:

$$
\begin{array}{ccc}
    & c & b \\
    a & c & b \\
    b & a & c \\
\end{array}
$$

Q: Write out the set of pairs in the relation $E_2$ explicitly, as we did for $E_1$ above. List all the $E_2$-closed and $E_2$-consistent sets. What are $\mu_{E_2}$ and $\nu_{E_2}$?
Principles of Induction/Coinduction

Corollary:

• Principle of induction:
  
  If X is \( F \)-closed, then \( \mu F \subseteq X \).

• Principle of coinduction:
  
  If X is \( F \)-consistent, then \( X \subseteq \nu F \).

The induction principle says that any property whose characteristic set is closed under \( F \) is true of all the elements of the inductively defined set \( \mu F \).

The coinduction principle, gives us a method for establishing that an element \( x \) is in the coinductively defined set \( \nu F \).

21.2 Finite and Infinite Types

To instantiate the general definitions of greatest fixed points and the coinductive proof method with the specifics of subtyping.
Tree Type

Definition: A tree type (or, simply, a tree) is a partial function $T \in \{1,2\}^* \rightarrow \{\rightarrow, \times, Top\}$ satisfying the following constraints:

- $T(*)$ is defined;
- if $T(\pi,\sigma)$ is defined then $T(\pi)$ is defined;
- if $T(\pi) = \rightarrow$ or $T(\pi) = \times$ then $T(\pi,1)$ and $T(\pi,2)$ are defined;
- if $T(\pi) = Top$ then $T(\pi,1)$ and $T(\pi,2)$ are undefined.

Definition: A tree type $T$ is finite if $\text{dom}(T)$ is finite.

The set of all tree types is written $\mathcal{T}$; the subset of all finite tree types is written $\mathcal{T}_f$.

Exercise: Give a universe $U$ and a generating function $F \in P(U) \rightarrow P(U)$ such that the set of finite tree types $\mathcal{T}_f$ is the least fixed point of $F$ and the set of all tree types $\mathcal{T}$ is its greatest fixed point.
21.3 Subtyping

Finite Subtyping

**Definition:** Two finite tree types $S$ and $T$ are in the subtype relation ("$S$ is a subtype of $T$") if $(S,T) \in \mu S_f$, where the monotone function

$$S_f : P(T_f \times T_f) \rightarrow P(T_f \times T_f)$$

is defined by

$$S_f(R) = \{ (T,\text{Top}) \mid T \in T_f \}$$

$$\cup \{ (S1 \times S2, T1 \times T2) \mid (S1,T1), (S2,T2) \in R \}$$

$$\cup \{ (S1 \rightarrow S2, T1 \rightarrow T2) \mid (T1,S1), (S2,T2) \in R \}.$$
Inference Rules

\[
\begin{align*}
T & <: \text{Top} \\
S_1 & <: T_1 \quad S_2 & <: T_2 \\
\frac{}{S_1 \times S_2 & <: T_1 \times T_2} \\
T_1 & <: S_1 \quad S_2 & <: T_2 \\
\frac{}{S_1 \rightarrow S_2 & <: T_1 \rightarrow T_2}
\end{align*}
\]

Infinite Subtyping

**Definition:** Two (finite or infinite) tree types \(S\) and \(T\) are in the subtype relation ("\(S\) is a subtype of \(T\)") if \((S,T) \in \nu_S\), where the monotone function \(S \in P(T \times T) \rightarrow P(T \times T)\) is defined by

\[
S(R) = \{(T,\text{Top}) \mid T \in \mathcal{T}\} \\
\cup \{(S_1 \times S_2, T_1 \times T_2) \mid (S_1, T_1), (S_2, T_2) \in R\} \\
\cup \{(S_1 \rightarrow S_2, T_1 \rightarrow T_2) \mid (T_1, S_1), (S_2, T_2) \in R\}.
\]
Inference Rules

\[ T \ll T_{\text{Top}} \]

\[ S_1 \ll T_1 \quad S_2 \ll T_2 \]

\[ \begin{array}{c}
S_1 \times S_2 \ll T_1 \times T_2 \\
\hline
\end{array} \]

\[ T_1 \ll S_1 \quad S_2 \ll T_2 \]

\[ \begin{array}{c}
S_1 \rightarrow S_2 \ll T_1 \rightarrow T_2 \\
\hline
\end{array} \]

EXERCISE [ ●]: Check that \( \forall S \) is not the whole of \( T \times T \) by exhibiting a pair \((S, T)\) that is not in \( \forall S \).

EXERCISE [ ●]: Is there a pair of types \((S, T)\) that is related by \( \forall S \), but not by \( \mu S \)? What about a pair of types \((S, T)\) that is related by \( \forall S_T \), but not by \( \mu S_T \)?
Transitivity

**Definition:** A relation $R \subseteq U \times U$ is **transitive** if $R$ is closed under the monotone function

$$TR(R) = \{(x,y) \mid \exists z \in U. (x,z), (z,y) \in R\},$$

i.e., if $TR(R) \subseteq R$.

**Lemma:** Let $F \in P(U \times U) \rightarrow P(U \times U)$ be a monotone function. If $TR(F(R)) \subseteq F(TR(R))$ for any $R \subseteq U \times U$, then $\nu F$ is transitive.

**Theorem:** $\nu S$ is transitive.

---

### 21.5 Membership Checking

Given a generating function $F$ on some universe $U$ and an element $x \in U$, check whether or not $x$ falls in $\nu F$. 
Invertible Generating Function

Definition: A generating function $F$ is said to be invertible if, for all $x \in U$, the collection of sets

$$G_x = \{X \subseteq U | x \in F(X)\}$$

either is empty or contains a unique member that is a subset of all the others.

We will consider invertible generating function in the rest of this chapter.

F-Supported/F-Ground

When $F$ is invertible, we define:

$$\text{support}_F(x) = \begin{cases} X & \text{if } X \in G_x \text{ and } \forall X' \in G_x, X \subseteq X' \\ \emptyset & \text{if } G_x = \emptyset \end{cases}$$

$$\text{support}_F(X) = \begin{cases} \bigcup_{x \in X} \text{support}_F(x) & \text{if } \forall x \in X, \text{ support}_F(x) \downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

Definition: An element $x$ is F-supported if $\text{support}_F(x) \downarrow$; otherwise, $x$ is F-unsupported. An F-supported element is called F-ground if $\text{support}_F(x) = \emptyset$.

Exercise: What is $\text{support}_F(x)$?
Support Graph

- An Example of the support graph of \( E \) function on \( \{a,b,c,d,e,f,g,h,i\} \)

\[ \text{x is in the greatest fixed point iff no unsupported element is reachable from x in the support graph.} \]

Greatest Fixed Point

**Definition:** Suppose \( F \) is an invertible generating function. Define the Boolean-valued function \( \text{gfp}_F \) (or just \( \text{gfp} \)) as follows:

\[
\text{gfp}(X) = \begin{cases} 
\text{if support}(X) \uparrow, \text{then false} \\
\text{else if support}(X) \subseteq X, \text{then true} \\
\text{else} \text{gfp}(\text{support}(X) \cup X).
\end{cases}
\]

**Theorem (Sound):**
1. If \( \text{gfp}_F(X) = \text{true} \), then \( X \subseteq \text{vF} \).
2. If \( \text{gfp}_F(X) = \text{false} \), then \( X \subseteq \text{vF} \).

**Theorem (Terminate):** If \( \text{reachable}_F(X) \) is finite, then \( \text{gfp}_F(X) \) is defined. Consequently, if \( F \) is finite state, then \( \text{gfp}_F(X) \) terminates for any finite \( X \subseteq U \).
More Efficient Algorithms

Inefficiency

Recomputation of “support”

\[
gfp\{a\} = gfp\{a, b, c\} = gfp\{a, b, c, e, f, g\} = gfp\{a, b, c, e, f, g, d\} = true
\]

support(a) is recomputed four times!
A More Efficient Algorithm

**Definition:** Suppose \( F \) is an invertible generating function. Define the function \( \text{gfp}^a \) as follows

\[
\text{gfp}^a(A, X) = \begin{cases} 
\text{false} & \text{if } \text{support}(X) = 1, \\
\text{true} & \text{if } X = \emptyset, \\
\text{gfp}^a(A \cup X, \text{support}(X) \setminus (A \cup X)) & \text{else}.
\end{cases}
\]

**Example:**

\[
\begin{align*}
\text{gfp}^a(\emptyset, \{a\}) &= \text{false} \\
&= \text{gfp}^a(\{a\}, \{b, c\}) \\
&= \text{gfp}^a(\{a, b, c\}, \{e, f, g\}) \\
&= \text{gfp}^a(\{a, b, c, e, f, g\}, \{d\}) \\
&= \text{gfp}^a(\{a, b, c, e, f, g, d\}, \emptyset) \\
&= \text{true}.
\end{align*}
\]

**Variation 1**

**Definition:** A small variation on \( \text{gfp}^a \) has the algorithm pick just one element at a time from \( X \) and expand its support. The new algorithm is called \( \text{gfp}^s \)

\[
\text{gfp}^s(A, X) = \begin{cases} 
\text{true} & \text{if } X = \emptyset, \\
\text{false} & \text{if } \text{support}(x) \uparrow \text{ then } \text{false}, \\
\text{gfp}^s(A \cup \{x\}, (X \cup \text{support}(x)) \setminus (A \cup \{x\})) & \text{else}.
\end{cases}
\]

\[\text{Tail-recursion}\]
Variation 2

**Definition:** Given an invertible generating function $F$, define the function $\text{gfp}^t$ as follows:

\[
\text{gfp}^t(A, x) = \begin{cases} 
A & \text{if } x \in A, \\
\text{fail} & \text{else if } \text{support}(x) = 1, \\
\text{else} & \\
\text{let } \{x_1, \ldots, x_n\} = \text{support}(x) \text{ in} \\
\text{let } A_0 = A \cup \{x\} \text{ in} \\
\text{let } A_1 = \text{gfp}^t(A_0, x_1) \text{ in} \\
\text{...} \\
\text{let } A_n = \text{gfp}^t(A_{n-1}, x_n) \text{ in} \\
A_n.
\end{cases}
\]

Regular Trees

If we restrict ourselves to regular types, then the sets of reachable states will be guaranteed to remain finite and the subtype checking algorithm will always terminate.
Regular Trees

**Definition:** A tree type $S$ is a **subtree** of a tree type $T$ if $S = \lambda \sigma . T(\pi, \sigma)$ for some $\pi$.

**Definition:** A tree type $T \in \Sigma$ is **regular** if $\text{subtrees}(T)$ is finite.

**Examples:**
- Every finite tree type is regular.
- $T = \text{Top} \times (\text{Top} \times (\text{Top} \times \ldots))$ is regular.
- $T = B \times (A \times (B \times (A \times (B \times (A \times (A \times (B \times (A \times (A \times (B \times \ldots))$ is irregular.

**Proposition:** The restriction of the generating function $S$ to regular tree types is finite state.

**Proof:** We need to show that for any pair $(S, T)$ of regular tree types, the set $\text{reachable}(S, T)$ is finite. Since $\text{reachable} \ (S, T) \subseteq \text{subtrees}(S) \times \text{subtrees}(T)$; the latter is finite as $S$ and $T$ are regular.
µ-Types

Establishes the correspondence between subtyping on µ-expressions and the subtyping on tree types

µ-Types:

**Definition:** Let X range over a fixed countable set \{X_1, X_2, \ldots\} of type variables. The set of raw µ-types is the set of expressions defined by the following grammar:

\[
T ::= X \\
    \text{Top} \\
    T \times T \\
    T \rightarrow T \\
    \mu X. T
\]

**Definition:** A raw µ-type T is **contractive** (and called µ-types) if, for any subexpression of T of the form \(\mu X. \mu X_1 \ldots \mu X_n.S\), the body S is not X.
Finite Notation for Infinite Tree Types

**Definition:** The function $\text{treeof}$, mapping closed $\mu$-types to tree types, is defined inductively as follows:

- $\text{treeof}(\text{Top})(\bullet) = \text{Top}$
- $\text{treeof}(T_1 \rightarrow T_2)(\bullet) = \rightarrow$
- $\text{treeof}(T_1 \rightarrow T_2)(i,\pi) = \text{treeof}(T_i)(\pi)$
- $\text{treeof}(T_1 \times T_2)(\bullet) = \times$
- $\text{treeof}(T_1 \times T_2)(i,\pi) = \text{treeof}(T_i)(\pi)$
- $\text{treeof}(\mu X . T)(\pi) = \text{treeof}([(X \mapsto \mu X . T)]T)(\pi)$
Subtyping Correspondence: µ-Types and Tree Types

Definition: Two µ-types S and T are said to be in the subtype relation if \((S, T) \in \nu S_m\), where the monotone function \(S_m \in \mathcal{P}(T_m^\times T_m^\times) \rightarrow \mathcal{P}(T_m^\times T_m^\times)\) is defined by:

\[
S_m(R) = \begin{cases} 
\emptyset & \text{if } T = \text{Top} \\
\{(S, T) \mid S \in T_m\} & \text{if } S = S_1 \times S_2, T = T_1 \times T_2 \\
\{(S_1, T_1), (S_2, T_2)\} & \text{if } S_1 = S_1 \rightarrow S_2, T = T_1 \rightarrow T_2 \\
\{(T_1, S_1), (S_2, T_2)\} & \text{if } S = S_1 \rightarrow S_2, T = T_1 \rightarrow T_2 \\
\{(S, \mu X. T) \mid (S, \mu X. T) \in R\} & \text{if } T = \mu X. T \\
\{(\mu X. S, T) \mid (\mu X. S, T) \in R\} & \text{if } T \neq \text{Top, and } S \neq \mu X. T \\
\end{cases}
\]

Theorem: Let \((S, T) \in T_m^\times T_m^\times\). Then \((S, T) \in \nu S_m\) iff \((\text{treeof } S, \text{treesof } T) \in \nu S\).

Exercise: What is the support for \(S_m\)?

\[
support_{S_m}(S, T) = \begin{cases} 
\emptyset & \text{if } T = \text{Top} \\
\{(S_1, T_1), (S_2, T_2)\} & \text{if } S = S_1 \times S_2 \text{ and } T = T_1 \times T_2 \\
\{(T_1, S_1), (S_2, T_2)\} & \text{if } S = S_1 \rightarrow S_2 \text{ and } T = T_1 \rightarrow T_2 \\
\{(S, \mu X. T) \mid (S, \mu X. T) \in R\} & \text{if } T = \mu X. T \\
\{(\mu X. S, T) \mid (\mu X. S, T) \in R\} & \text{if } T \neq \text{Top, and } S \neq \mu X. T \\
\end{cases}
\]
Subtyping Algorithm for \( \mu \)-Types

Instantiating \( gfp^t \) for subtyping relation on \( \mu \)-Types.

\[
\text{subtype}(A, S, T) = \begin{cases} 
A & \text{if } (S, T) \in A, \text{ then} \\
\text{else let } A_0 = A \cup \{(S, T)\} \text{ in} \\
\text{if } T = \text{Top}, \text{ then} \\
A_0 \\
\text{else if } S = S_1 \times S_2 \text{ and } T = T_1 \times T_2, \text{ then} \\
\text{let } A_1 = \text{subtype}(A_0, S_1, T_1) \text{ in} \\
\text{subtype}(A_1, S_2, T_2) \\
\text{else if } S = S_1 \rightarrow S_2 \text{ and } T = T_1 \rightarrow T_2, \text{ then} \\
\text{let } A_1 = \text{subtype}(A_0, T_0, S_1) \text{ in} \\
\text{subtype}(A_1, S_2, T_2) \\
\text{else if } T = \mu X. T_1, \text{ then} \\
\text{subtype}(A_0, S, [X \rightarrow \mu X. T_1]T_1) \\
\text{else if } S = \mu X. S_1, \text{ then} \\
\text{subtype}(A_0, [X \rightarrow \mu X. S_1]S_1, T) \\
\text{else} \\
\text{fail} 
\end{cases}
\]

Terminate?

Summary

- We study the theoretical foundation of type checkers (subtyping) for equi-recursive types.
  - Induction/coinduction & proof principles
  - Finite and Infinite Types/Subtyping
  - Membership checking algorithm
Homework

21.5.2 Exercise [★★]: Verify that $S_t$ and $S$, the generating functions for the subtyping relations from Definitions 21.3.1 and 21.3.2, are invertible, and give their support functions.