

## 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  $F \in \mathcal{P}(U) \rightarrow \mathcal{P}(U)$  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  $U=\{a, b, c\}$ :

$$E1(\emptyset) = \{c\}$$

$$E1(\{a\}) = \{c\}$$

$$E1(\{b\}) = \{c\}$$

$$E1(\{c\}) = \{b, c\}$$

$$E1(\{a,b\}) = \{c\}$$

$$E1(\{a, c\}) = \{b, c\}$$

$$E1(\{b, c\}) = \{a, b, c\}$$

$$E1(\{a, b, c\}) = \{a, b, c\}$$

$$\frac{\quad}{c} \quad \frac{c}{b} \quad \frac{b}{a} \quad \frac{c}{\quad}$$

Q: Which subset is E1-closed, E1-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  $\nu F$ .



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

$$E1(\emptyset) = \{c\}$$

$$E1(\{a\}) = \{c\}$$

$$E1(\{b\}) = \{c\}$$

$$E1(\{c\}) = \{b, c\}$$

$$E1(\{a,b\}) = \{c\}$$

$$E1(\{a, c\}) = \{b, c\}$$

$$E1(\{b, c\}) = \{a, b, c\}$$

$$E1(\{a, b, c\}) = \{a, b, c\}$$

$$\frac{\quad}{c} \quad \frac{c}{b} \quad \frac{b}{a} \quad \frac{c}{\quad}$$

Q: What are  $\mu E1$  and  $\nu E1$ ?



**Exercise:** Suppose a generating function  $E2$  on the universe  $\{a, b, c\}$  is defined by the following inference rules:

$$\frac{}{a} \quad \frac{c}{b} \quad \frac{a \quad b}{c}$$

Q: Write out the set of pairs in the relation  $E2$  explicitly, as we did for  $E1$  above. List all the  $E2$ -closed and  $E2$ -consistent sets. What are  $\mu_{E2}$  and  $\nu_{E2}$ ?





# 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

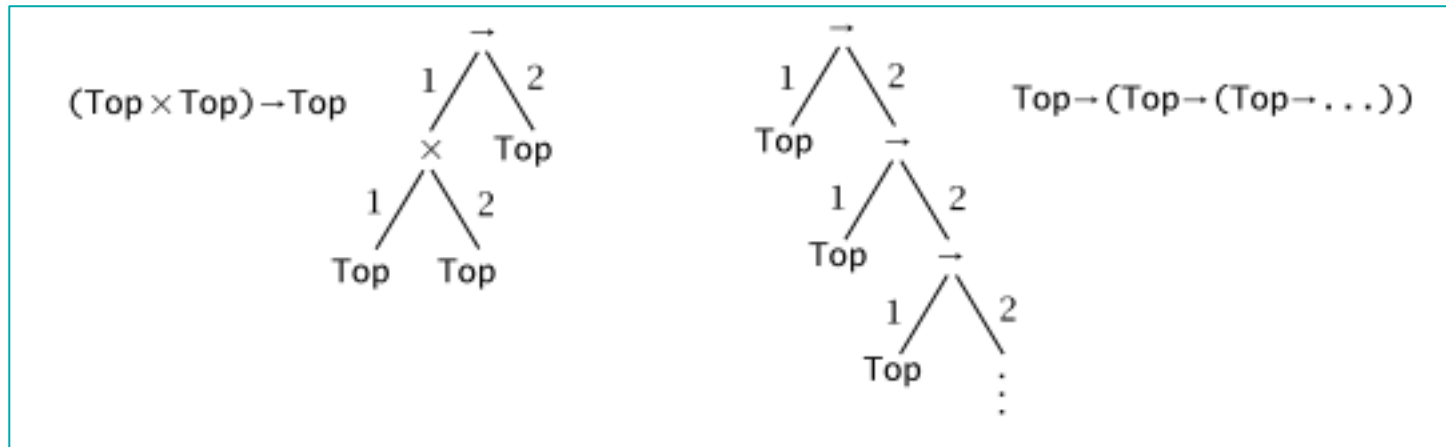
To instantiate the general definitions of greatest fixed points and the coinductive proof method with the specifics of subtyping.



# Tree Type

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

- $T(\bullet)$  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  $T(\pi) = \text{Top}$  then  $T(\pi, 1)$  and  $T(\pi, 2)$  are undefined.



Note:  $T(\cdot) = \text{Top}$



**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  $F \in \mathcal{P}(U) \rightarrow \mathcal{P}(U)$  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

$$S_f \in \mathcal{P}(\mathcal{T}_f \times \mathcal{T}_f) \rightarrow \mathcal{P}(\mathcal{T}_f \times \mathcal{T}_f)$$

is defined by

$$\begin{aligned} S_f(R) = & \{ (T, \text{Top}) \mid T \in \mathcal{T}_f \} \\ & \cup \{ (S_1 \times S_2, T_1 \times T_2) \mid (S_1, T_1), (S_2, T_2) \in R \} \\ & \cup \{ (S_1 \rightarrow S_2, T_1 \rightarrow T_2) \mid (T_1, S_1), (S_2, T_2) \in R \}. \end{aligned}$$



## Inference Rules

$T \Leftarrow \text{Top}$

$S1 \Leftarrow T1 \quad S2 \Leftarrow T2$

-----

$S1 \times S2 \Leftarrow T1 \times T2$

$T1 \Leftarrow S1 \quad S2 \Leftarrow T2$

-----

$S1 \rightarrow S2 \Leftarrow T1 \rightarrow T2$



# 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

$$S \in \mathcal{P}(\mathcal{T} \times \mathcal{T}) \rightarrow \mathcal{P}(\mathcal{T} \times \mathcal{T})$$

is defined by

$$\begin{aligned} S(R) = & \{(T, \text{Top}) \mid T \in \mathcal{T}\} \\ & \cup \{(S_1 \times S_2, T_1 \times T_2) \mid (S_1, T_1), (S_2, T_2) \in R\} \\ & \cup \{(S_1 \rightarrow S_2, T_1 \rightarrow T_2) \mid (T_1, S_1), (S_2, T_2) \in R\}. \end{aligned}$$





# Transitivity

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

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

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

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

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



## A Digression on Transitivity

The possibility of giving a declarative presentation with the rule of transitivity turns out to be a consequence of a “trick” that can be played with inductive, but not coinductive, definitions.

- The union of two sets of rules, when applied inductively, generates the least relation that is closed under both sets of rules separately.
- Adding transitivity to the rules generating a coinductively defined relation always gives us a degenerate relation.



## 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  $\mathcal{V}F$ .



# Invertible Generating Function

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

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

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:

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

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

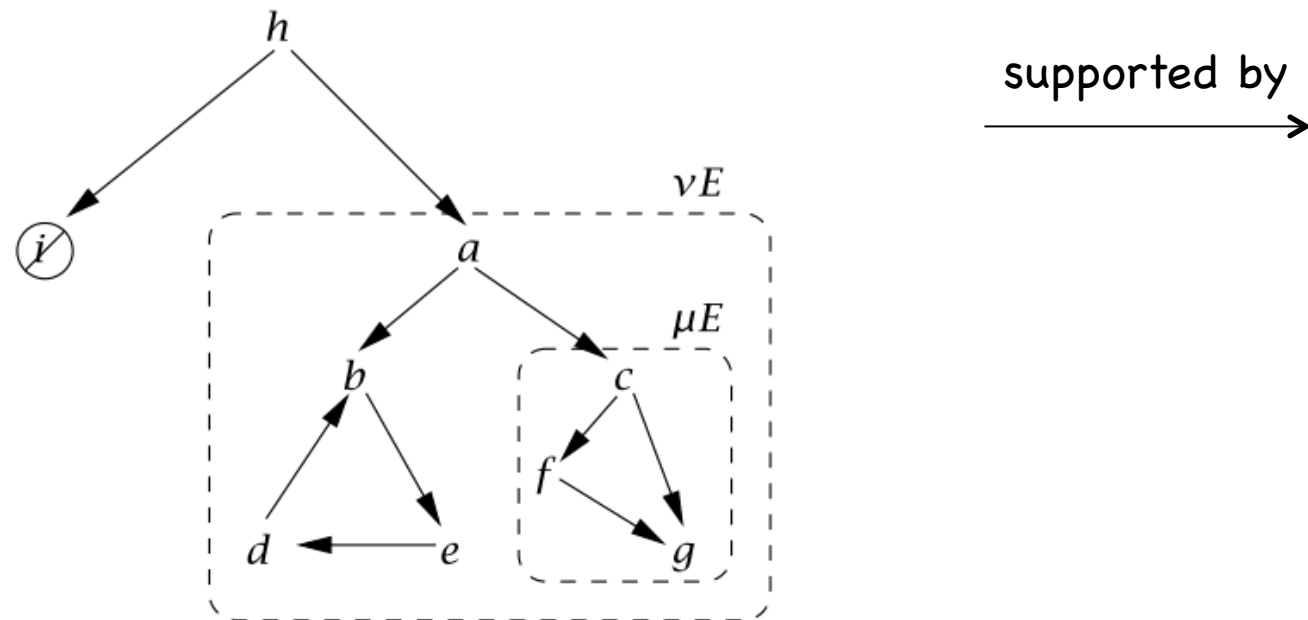
**Definition:** An element  $x$  is **F-supported** if  $\text{support}_F(x) \downarrow$ ; otherwise,  $x$  is F-unsupported. An F-supported element is called **F-ground** if  $\text{support}_F(x) = \emptyset$ .

Exercise: What is  $\text{support}_S(x)$ ?



# Support Graph

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



$x$  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:

$$\begin{aligned} \text{gfp}(X) = & \text{if } \text{support}(X) \uparrow, \text{ then } \text{false} \\ & \text{else if } \text{support}(X) \subseteq X, \text{ then } \text{true} \\ & \text{else } \text{gfp}(\text{support}(X) \cup X). \end{aligned}$$

**Theorem (Sound):**

1. If  $\text{gfp}_F(X) = \text{true}$ , then  $X \subseteq \nu F$ .
2. If  $\text{gfp}_F(X) = \text{false}$ , then  $X \not\subseteq \nu F$ .

**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  $gfp^a$  as follows

$$gfp^a(A, X) = \begin{array}{l} \text{if } support(X) \uparrow, \text{ then } false \\ \text{else if } X = \emptyset, \text{ then } true \\ \text{else } GFP^a(A \cup X, support(X) \setminus (A \cup X)). \end{array}$$

Tail-recursion

Example:

$$\begin{aligned} & GFP^a(\emptyset, \{a\}) \\ &= GFP^a(\{a\}, \{b, c\}) \\ &= GFP^a(\{a, b, c\}, \{e, f, g\}) \\ &= GFP^a(\{a, b, c, e, f, g\}, \{d\}) \\ &= GFP^a(\{a, b, c, e, f, g, d\}, \emptyset) \\ &= true. \end{aligned}$$



## Variation 1

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

$$gfp^s(A, X) = \begin{array}{l} \text{if } X = \emptyset, \text{ then } true \\ \text{else let } x \text{ be some element of } X \text{ in} \\ \quad \text{if } x \in A \text{ then } gfp^s(A, X \setminus \{x\}) \\ \quad \text{else if } support(x) \uparrow \text{ then } false \\ \quad \text{else } gfp^s(A \cup \{x\}, (X \cup support(x)) \setminus (A \cup \{x\})). \end{array}$$


## Variation 2

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

$$\begin{aligned}gfp^{\dagger}(A, x) = & \text{if } x \in A, \text{ then } A \\ & \text{else if } support(x) \uparrow, \text{ then } fail \\ & \text{else} \\ & \quad \text{let } \{x_1, \dots, x_n\} = support(x) \text{ in} \\ & \quad \text{let } A_0 = A \cup \{x\} \text{ in} \\ & \quad \text{let } A_1 = GFP^{\dagger}(A_0, x_1) \text{ in} \\ & \quad \dots \\ & \quad \text{let } A_n = GFP^{\dagger}(A_{n-1}, x_n) \text{ in} \\ & \quad A_n.\end{aligned}$$


## 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  $T \in \mathcal{T}$  is **regular** if  $\text{subtrees}(T)$  is finite.

Examples:

- Every finite tree type is regular.
- $T = \text{Top} \times (\text{Top} \times (\text{Top} \times \dots))$  is regular.
- $T = B \times (A \times (B \times (A \times (A \times (B \times (A \times (A \times (A \times (B \times \dots))$  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.



## $\mu$ -Types

Establishes the correspondence between  
subtyping on  $\mu$ -expressions and the  
subtyping on tree types





## $\mu$ -Types:

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

$$\begin{aligned} T & ::= X \\ & \quad \text{Top} \\ & \quad T \times T \\ & \quad T \rightarrow T \\ & \quad \mu X. T \end{aligned}$$
$$\mathcal{T}_m$$

**Definition:** A raw  $\mu$ -type  $T$  is **contractive** (and called  **$\mu$ -types**) if, for any subexpression of  $T$  of the form  $\mu X. \mu X_1 \dots \mu X_n. S$ , the body  $S$  is not  $X$ .



# Finite Notation for Infinite Tree Types

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

$$\text{treeof}(\text{Top})(\bullet) = \text{Top}$$

$$\text{treeof}(\tau_1 \rightarrow \tau_2)(\bullet) = \rightarrow$$

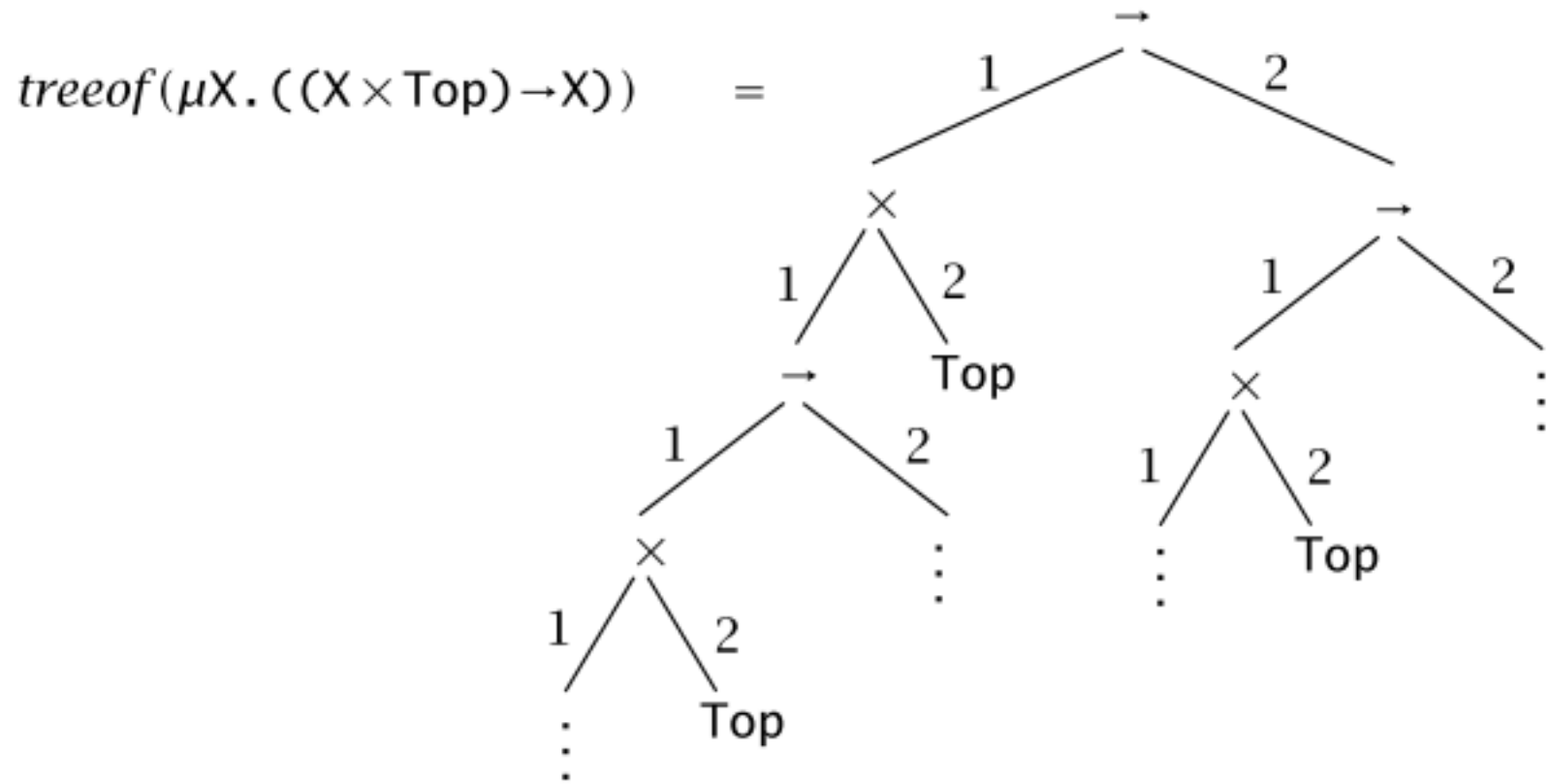
$$\text{treeof}(\tau_1 \rightarrow \tau_2)(i, \pi) = \text{treeof}(\tau_i)(\pi)$$

$$\text{treeof}(\tau_1 \times \tau_2)(\bullet) = \times$$

$$\text{treeof}(\tau_1 \times \tau_2)(i, \pi) = \text{treeof}(\tau_i)(\pi)$$

$$\text{treeof}(\mu X. \tau)(\pi) = \text{treeof}([X \mapsto \mu X. \tau]\tau)(\pi)$$





# Subtyping Correspondence: $\mu$ -Types and Tree Types

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

$$\begin{aligned} S_m(R) = & \{(S, \text{Top}) \mid S \in \mathcal{T}_m\} \\ & \cup \{(S_1 \times S_2, T_1 \times T_2) \mid (S_1, T_1), (S_2, T_2) \in R\} \\ & \cup \{(S_1 \rightarrow S_2, T_1 \rightarrow T_2) \mid (T_1, S_1), (S_2, T_2) \in R\} \\ & \cup \{(S, \mu X. T) \mid (S, [X \mapsto \mu X. T]T) \in R\} \\ & \cup \{(\mu X. S, T) \mid ([X \mapsto \mu X. S]S, T) \in R, T \neq \text{Top}, \text{ and } T \neq \mu Y. T_1\}. \end{aligned}$$

**Theorem:** Let  $(S, T) \in \mathcal{T}_m \times \mathcal{T}_m$ . Then  $(S, T) \in \nu S_m$  iff  $(\text{treeof } S, \text{treesof } T) \in \nu S$ .



**Exercise:** What is the support for  $S_m$ ?

$$\text{support}_{S_m}(S, T) = \begin{cases} \emptyset & \text{if } T = \text{Top} \\ \{(S_1, T_1), (S_2, T_2)\} & \text{if } S = S_1 \times S_2 \text{ and} \\ & T = T_1 \times T_2 \\ \{(T_1, S_1), (S_2, T_2)\} & \text{if } S = S_1 \rightarrow S_2 \text{ and} \\ & T = T_1 \rightarrow T_2 \\ \{(S, [X \mapsto \mu X. T_1]T_1)\} & \text{if } T = \mu X. T_1 \\ \{([X \mapsto \mu X. S_1]S_1, T)\} & \text{if } S = \mu X. S_1 \text{ and} \\ & T \neq \mu X. T_1, T \neq \text{Top} \\ \uparrow & \text{otherwise.} \end{cases}$$



# Subtyping Algorithm for $\mu$ -Types

Instantiating  $\text{gfp}^\dagger$  for subtyping relation on  $\mu$ -Types.

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

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  $S_f$  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. □

