Lesson 9
Recursive Types

2/19, 21
Chapters 20, 21

Recursive type

- Recursive type terms are infinite, but regular
- For finite terms, use induction based on least fixed points
- For infinite terms/trees, we have to use coinduction, based on greatest fixed points.
- For least fixed points, recursively explore all subterms
- For greatest fixed points, recursively explore the support set (and hope that it is finite)
Greatest fixed point algorithm

How to check for membership in the gfp (of an invertible generating function $F$)?

$$
gfp(X) = \begin{cases} 
\text{false} & \text{if support}(X) \uparrow \\
\text{true} & \text{if support}(X) \Downarrow X \\
gfp(\text{support}(X) \Downarrow X) & \text{else}
\end{cases}
$$

**Correctness:**

Thm: 1. If $gfp(X) = \text{true}$ then $X \Downarrow F$
2. If $gfp(X) = \text{false}$ then $X \uparrow F$

Proof: induction on recursive computation of $gfp(X)$

Gfp algorithm: termination

$$
pred(x) = \begin{cases} 
\emptyset & \text{if support}(x) \uparrow \\
\text{support}(X) & \text{if support}(X) \Downarrow \\
\text{pred}(x) & \text{else}
\end{cases}
$$

$$
pred(X) = X \Downarrow \text{pred}(x)
$$

$$
\text{reachable}(X) = \bigcap_{n=0}^{\infty} \text{pred}^n(x)
$$

$$
\text{reachable}(x) = \text{reachable}(\{x\})
$$

$F$ is finite state if $\text{reachable}(x)$ is finite for each $x$

Thm: If $\text{reachable}(X)$ is finite, then $gfp(X)$ is defined.
If $F$ is finite state, $gfp(X)$ terminates for any finite $X$. 
More efficient gfp: \( \text{gfpa} \)

Avoid repeated addition of elements by keeping track of all elements examined before in a new argument \( A \).

\[
\text{gfpa}(A,X) = \begin{cases} 
\text{false} & \text{if support}(X) \uparrow \\
\text{true} & \text{if } X = \emptyset \\
\text{gfpa}(A \uplus X, \text{support}(X) \setminus (A \uplus X)) & \text{else}
\end{cases}
\]

This version considers only new elements added by support function.

\( x \not\in F \iff \text{gfpa}(\emptyset,\{x\}) \)

More efficient gfp: \( \text{gfpt} \)

Threaded version of gfp, adding one element at a time and producing visited set as the result (assume \( \text{support}(x) \) finite):

\[
\text{gfpt}(A,x) = \begin{cases} 
A & \text{if } x \not\subseteq A \\
\text{fail} & \text{if } \text{support}(\{x\}) \uparrow \\
\text{fold } \text{gfpt}(A \uplus \{x\}, \text{support}(x)) & \text{else}
\end{cases}
\]

where \( \text{fold} \) is a function like list fold but operating on sets:

\[
\text{fold } f X \emptyset = X \\
\text{fold } f X \{y_1, y_2, \ldots, y_n\} = f f(X, y_1) \{y_2, \ldots, y_n\}
\]

Correctness: \( x \not\subseteq F \iff \text{gfpt}(\emptyset,\{x\}) \emptyset \)
Regular Trees

**Def:** \( S \subseteq T \) is a **subtree** of \( T \subseteq T' \) if \( S = T(p, \ell) \) for some path \( \ell \subseteq \text{dom}(T) \). \( \text{subtrees}(T) \) is the set of subtrees of \( T \).

**Def:** \( T \subseteq T' \) is **regular** if \( \text{subtrees}(T) \) is finite. \( T_r \) is the set of regular trees.

Subtype relation

The subtyping relation on \( T \) is defined as the greatest fixed point of the relation generator function

\[
S(R) = \{(T, \text{Top}) \mid T \subseteq T\}
\]

\[
\circ\{(S1 \subseteq S2, T1 \subseteq T2) \mid (S1,T1), (S2,T2) \in R\}
\]

\[
\circ\{(S1 \subseteq S2, T1 \subseteq T2) \mid (T1,S1), (S2,T2) \in R\}
\]

\( S_r \) is the restriction of \( S \) to \( T_r \) (regular trees).

**Prop:** \( S_r \) is finite state.
### □-Types

**Def**: $\mathcal{T}_{\text{m-raw}}$, the set of raw □-types, is defined by the grammar:

$$T ::= X \mid \text{Top} \mid T \square T \mid T \bowtie T \mid \square X.T$$

This contains useless terms like □X.X that we should exclude.

**Def**: $T \square \mathcal{T}_{\text{m-raw}}$ is **contractive** if for any subtree of $T$ of the form □X.□X₁. ... □Xₙ. S, S is not X. (I.e. there is always an occurrence of □ or □ between a binder □X and an applied occurrence of X. $\mathcal{T}_{\text{m}}$ denotes the set of contractive □-types.

*Can define a function treeof : $\mathcal{T}_{\text{m}} \square \mathcal{T}$ mapping □-types to tree types.*

### Subtype relation for □-Types

The generating function $S_m$ for the subtyping relation on $\mathcal{T}'_m$ is given by

$$S_m(R) = \{(S, \text{Top}) \mid S \square \mathcal{T}'_m\}$$

- $\{(S_1 \square S_2, T_1 \square T_2) \mid (S_1, T_1), (S_2, T_2) \square R\}$
- $\{(S_1 \bowtie S_2, T_1 \bowtie T_2) \mid (T_1, S_1), (S_2, T_2) \bowtie R\}$
- $\{(S, \square X.T) \mid (S, [X \to □X.T]T) \square R\}$
- $\{([X:S, T] \mid ([X \to □X.S]S, T) \square R, T \neq \text{Top}, T \neq □Y.T)\}$
Subtype relation for □-Types

$S_m$ is invertible with support function

\[
\text{support}(S,T) = \begin{cases} 
\emptyset & \text{if } T = \text{Top} \\
\{(S_1,T_1), (S_2,T_2)\} & \text{if } S = S_1 \sqcap S_2 \text{ and } T = T_1 \sqcap T_2 \\
\{(T_1,S_1), (S_2,T_2)\} & \text{if } S = S_1 \sqcap S_2 \text{ and } T = T_1 \sqcap T_2 \\
\{(S, [X \to □X,T_1]T_1)\} & \text{if } T = □X,T_1 \\
\{(X \to □X,S_1]S_1, T)\} & \text{if } S = □X,S_1, T \neq \text{Top}, T \neq □Y,T_1 \\
\uparrow & \text{otherwise}
\end{cases}
\]

Thm: \((S,T) \sqsubseteq S_m\) iff \(\text{treeof}(S,T) \sqsubseteq S\)

Subtyping algorithm

Specialize \(\text{gfp}^l\) to \(S_m\).

\[
\text{subtype} (S,T) = \begin{cases} 
\text{if } (S,T) \sqsubseteq A \text{ then } A \\
\text{else let } A_0 = A \langle S,T \rangle \text{ in} \\
\quad \text{if } T = \text{Top} \text{ then } A_0 \\
\quad \text{else if } S = S_1 \sqcap S_2 \text{ and } T = T_1 \sqcap T_2 \text{ then} \\
\quad\quad \text{let } A_1 = \text{subtype} (A_0,S_1,T_1) \text{ in } \text{subtype} (A_1,S_2,T_2) \\
\quad \text{else if } S = S_1 \sqcap S_2 \text{ and } T = T_1 \sqcap T_2 \text{ then} \\
\quad\quad \text{let } A_1 = \text{subtype} (A_0,T_1,S_1) \text{ in } \text{subtype} (A_1,S_2,T_2) \\
\quad \text{else if } T = □X,T_1 \text{ then } \text{subtype} (A_0,S,[X \to □X,T_1]T_1) \\
\quad \text{else if } S = □X,S_1 \text{ then } \text{subtype} (A_0,[X \to □X,S_1]S_1) \\
\quad \text{else } \text{fail}
\end{cases}
\]
Subtyping Iso-recursive types

The Amber rule:

\[ \frac{\Gamma, \ X <: \ Y \vdash S <: T}{\Gamma \vdash \forall \cdot X.\ S <: \forall \cdot Y.\ T} \quad \text{(S-Amber)} \]

\[ \frac{X <: Y \square \square}{\square \vdash X <: Y} \quad \text{(S-Assumption)} \]