Exercise 1.1-1.4

1.1

Prove by structural induction on ASTs that if $\mathcal{X} \subseteq \mathcal{Y} \implies \mathcal{A}[\mathcal{X}] \subseteq \mathcal{A}[\mathcal{Y}]$

Proof:

Suppose $\mathcal{X} \subseteq \mathcal{Y}$. Let $a \in \mathcal{A}[\mathcal{X}]$ be an AST. Do `Struct Induct` on a.

1. If $a = x$, then $a = y$ for some $y \in \mathcal{Y}$ by definition. Then $a \in \mathcal{A}[\mathcal{Y}]$.
2. Otherwise, $a = o(a_1; ...; a_n)$ and by induction, $a_i \in \mathcal{A}[\mathcal{Y}]$ for all $1 \leq i \leq n$. By definition, $a \in \mathcal{A}[\mathcal{Y}]$.

$\square$

1.2

Prove by structural induction modulo renaming on ABTs that if $\mathcal{X} \subseteq \mathcal{Y} \implies \mathcal{B}[\mathcal{X}] \subseteq \mathcal{B}[\mathcal{Y}]$

Proof:

Let $a \in \mathcal{B}[\mathcal{X}]$ be an ABT. Do `Struct Induct /` α on a.

1. Same as above
2. Otherwise, $a = o(\vec{x}_1.a_1; ...; \vec{x}_n.a_n)$ and by induction, for each $1 \leq i \leq n$, $\hat{\rho}_i(a_i) \in \mathcal{B}[\mathcal{Y}]$ where ρ_i is a fresh renaming bijection. Then by definition, $a \in \mathcal{B}[\mathcal{Y}]$.

$\square$

1.3

Show that if $a =_\alpha a'$ and $b =_\alpha b'$ and both $[b/x]a$ and $[b'/x]a'$ are defined, then $[b/x]a =_\alpha [b'/x]a'$.

Proof:

By `Struct Induct /` α on a,

1. If $a = x$, then by the first α-equivalence condition and the assumption that $x =_\alpha a'$, then $x = a'$. Hence by assumption:

$$[b/x]a = [b/x]x = b =_\alpha b' = [b'/x]x = [b'/x]a'$$

2. The case where $a = y$ and $y \neq x$ is trivial by the definition of α-equivalence.
3. Otherwise, $a = o(\vec{x}_1.a_1; ...; \vec{x}_n.a_n)$. By the second α-equivalence condition, $a' = o(\vec{x}'_1.a'_1; ...; \vec{x}'_n.a'_n)$ where we have bijections$\hat{\rho}_i(a_i)$ rename each variable. Also, by induction, for each $1 \leq i \leq n$, $[b/x]a_i =_\alpha [b'/x]a'_i$. Therefore:

$$[b/x]o(\vec{x}_1.a_1; ...; \vec{x}_n.a_n) = o(\vec{x}_1.\tilde{a_1}; ...; \vec{x}_n.\tilde{a_n}) =_\alpha o(\vec{x}'_1.\tilde{a_1}'; ...; \vec{x}'_n.\tilde{a_n}') = [b'/x]o(\vec{x}'_1.a'_1; ...; \vec{x}'_n.a'_n)$$

where $\tilde{a}_i = [b/x]a_i$ if $x \notin \vec{x}_i$ and $\tilde{a}_i = a_i$ otherwise. The same is for a'. Sorry for the overloading notations used here.

$\square$

1.4

Bound variables can be seen as the formal analogs of pronouns (she, it, they, etc.) in natural languages.

Abstract binding graphs (abg's) are constructed as follows:

- Free variables are atomic nodes with no outgoing edges
- Operators with n arguments are n-ary nodes, with one outgoing edge directed at each of their children.
- Abstractors are nodes with one edge directed to the scope of the abstracted variable.
- Bound variables are back edges directed at the abstractor that introduced it.

Asts are *acyclic* directed graphs, whereas general abts can be *cyclic*.

Exp 1:

```
let(num[2]; x.plus(x; num[3]))
```

Exp 2:

```
let(y; x.let(num[2]; z.plus(x; z)))
```