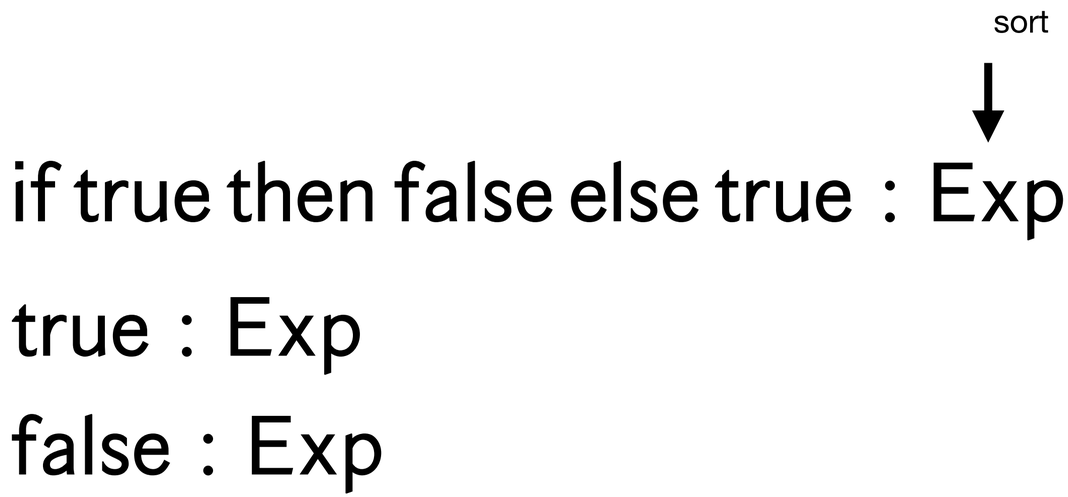
Syntactic Objects

Surface Syntax

```
if true then false else true
```

Surface Syntax

Surface Syntax



Surface Syntax

$$\text{if } - \text{ then } - \text{ else } - : (\text{Exp} \times \text{Exp} \times \text{Exp}) \to \text{Exp}$$

$$\text{true} : \text{Exp}$$

$$\text{false} : \text{Exp}$$

$$\text{if true then false else true} : \text{Exp}$$

Abstract Syntax

$$\text{if}(-,-,-) : (\text{Exp} \times \text{Exp} \times \text{Exp}) \to \text{Exp}$$

$$\text{true} : \text{Exp}$$

$$\text{false} : \text{Exp}$$

$$\text{if}(\text{true}, \text{false}, \text{true}) : \text{Exp}$$

Abstract Syntax Trees

$$\text{plus} : \text{Exp} \times \text{Exp} \to \text{Exp}$$

$$\text{num} : \mathbb{N} \to \text{Exp}$$

$$\text{num}[42] : \text{Exp}$$

$$\text{num}[4] : \text{Exp}$$

$$\text{plus}(\text{num}[4], \text{num}[42]) : \text{Exp}$$

Abstract Syntax Trees: Variables

$$\text{times} : \text{Exp} \times \text{Exp} \to \text{Exp}$$

$$\text{num} : \mathbb{N} \to \text{Exp}$$

variable → $x : \text{Exp}$

$$\text{num}[4] : \text{Exp}$$

$$\text{times}(x, \text{num}[4]) : \text{Exp}$$

Abstract Syntax Trees: Variable Substitution

$$\text{plus} : \text{Exp} \times \text{Exp} \to \text{Exp}$$
$$\text{times} : \text{Exp} \times \text{Exp} \to \text{Exp}$$
$$\text{num} : \mathbb{N} \to \text{Exp}$$
$$\text{num}[42] : \text{Exp}$$
$$\text{num}[4] : \text{Exp}$$
$$\text{times}(\text{plus}(\text{num}[4], \text{num}[42]), \text{num}[4]) : \text{Exp}$$

Abstract Syntax Trees: Variable Substitution

$$\text{plus} : \text{Exp} \times \text{Exp} \to \text{Exp}$$
$$\text{times} : \text{Exp} \times \text{Exp} \to \text{Exp}$$
$$\text{num} : \mathbb{N} \to \text{Exp}$$
$$\text{num}[42] : \text{Exp}$$
$$\text{num}[4] : \text{Exp}$$
$$\text{times}(\text{plus}(\text{num}[4], \text{num}[42]), \text{num}[4]) : \text{Exp}$$

Abstract Syntax Trees Defined

Let S be a finite set of **sorts**, $\{O_s\}_{s \in S}$ be a sort-indexed family of operators o of sort s with arity $\mathsf{ar}(o) = (s_1, \ldots, s_n)$, and $\{\chi_s\}_{s \in S}$ be a sort-indexed family of variables x of sort s. The family $A[\chi] = A[\chi_s]_{s \in S}$ of abstract syntax trees (ASTs) of sort s is defined as follows:

- if $x \in \chi_s$, then $x \in A[\chi]_s$
- if $o \in O_s$ with $\mathsf{ar}(o) = (s_1, \ldots, s_n)$ and $a_1 \in A[\chi]_{s_1}, \ldots, a_n \in A[\chi]_{s_n}$, then $o(a_1, \ldots, a_n) \in A[\chi]_s$

Abstract Syntax Trees Defined

Let S be a finite set of **sorts**, $\{O_s\}_{s \in S}$ be a sort-indexed family of operators o of sort s with arity $\mathsf{ar}(o) = (s_1, \ldots, s_n)$, and $\{\chi_s\}_{s \in S}$ be a sort-indexed family of variables x of sort s. The family $A[\chi] = A[\chi_s]_{s \in S}$ of abstract syntax trees (ASTs) of sort s is defined as follows:

- if $x \in \chi_s$, then $x \in A[\chi]_s$
- if $o \in O_s$ with $\mathsf{ar}(o) = (s_1, \ldots, s_n)$ and $a_1 \in A[\chi]_{s_1}, \ldots, a_n \in A[\chi]_{s_n}$, then $o(a_1, \ldots, a_n) \in A[\chi]_s$

Exercise: Come up with three example ASTs.

Abstract Syntax Trees: Structural Induction

To show that a property P for every AST it suffices to show $\mathsf{P}(a)$ holds for every $a \in A[\chi]$, which holds when:

1. (Base Case) if $x \in \chi_s$, then $\mathsf{P}_s(x)$, and
2. (Step Case) if $o \in O_s$ with $\mathsf{ar}(o) = (s_1, \ldots, s_n)$, then if $\mathsf{P}_{s_1}(a_1), \ldots, \mathsf{P}_{s_n}(a_n)$ all hold, then $\mathsf{P}_s(o(a_1, \ldots, a_n))$ holds.

Abstract Syntax Trees: Structural Induction

Lemma: If $\mathcal{X} \subseteq \mathcal{Y}$, then $A[\mathcal{X}] \subseteq A[\mathcal{Y}]$.

Proof. By structural induction.

1. (Base Case) If $x \in \mathcal{X}_s$ which implies that $x \in A[\mathcal{X}]_s$, then by assumption $x \in \mathcal{Y}$, and hence, by definition $x \in A[\mathcal{Y}]_s$.

2. (Step Case) Suppose $o \in \mathcal{O}_s$, $\mathsf{ar}(o) = (s_1, \ldots, s_n)$, $\mathcal{X} \subseteq \mathcal{Y}$. Then by induction:
   $a_1 \in A[\mathcal{X}]_{s_1} \iff a_1 \in A[\mathcal{Y}], \ldots, a_n \in A[\mathcal{X}] \iff a_n \in A[\mathcal{Y}]_{s_n}$. Then by definition
   $o(a_1, \ldots, a_n) \in A[\mathcal{X}]_s \iff o(a_1, \ldots, a_n) \in A[\mathcal{Y}]_s$.

Abstract Syntax Trees: Substitution

Variables are given their meaning through

$$
\begin{aligned}
&[\mathsf{num}[42]/x](\mathsf{plus}(x, \mathsf{mult}(\mathsf{num}[3], x)) \\
&\quad = \mathsf{plus}([\mathsf{num}[42]/x]x, [\mathsf{num}[42]/x]\mathsf{mult}(\mathsf{num}[3], x) \\
&\quad = \mathsf{plus}([\mathsf{num}[42]/x]x, \mathsf{mult}([\mathsf{num}[42]/x]\mathsf{num}[3], [\mathsf{num}[42]/x]x) \\
&\quad = \mathsf{plus}(\mathsf{num}[42], \mathsf{mult}(\mathsf{num}[3], \mathsf{num}[42])
\end{aligned}
$$

Abstract Syntax Trees: Substitution

Substitution $[b_1/x]b_2 \in A[\mathcal{X}]_{s_2}$ on any AST $A[\mathcal{X}]$ where x is a variable of sort s_1, $b_1 \in A[\mathcal{X}]_{s_1}$, $b_2 \in A[\mathcal{X}, x]_{s_2}$ is defined as follows:

1. $[b_1/x]x = b_1$
2. $[b_1/x]y = y$, when $x \neq y$
3. $[b_1/x]o(a_1, \ldots, a_n) = o([b_1/x]a_1, \ldots, [b_1/x]a_n)$

Abstract Syntax Trees: Extension

Let $\mathcal{X}$ be a sort-indexed family of variables. Then:
$(\mathcal{X}, x)$ where x is a variable of sort s such that $x \notin \mathcal{X}_s$, to stand for the sort-indexed family $\mathcal{Y}$ such that $\mathcal{Y}_s = \mathcal{X}_s \cup \{x\}$ and $\mathcal{Y}_{s'} = \mathcal{X}_{s'}$ for all $s' \neq s$.

This is also known a **adjoining.**

Bound Variables: Examples

Swift:

```
func exampleFunc(arg1: T1,...,argi: Ti) -> T {
  ...
}
```

Bound Variables: Examples

C#:

```
T exampleFunc(T1 arg1, ..., Ti argi) {
  ...
}
```

Bound Variables: Examples

Javascript:

```
function exampleFunc(arg1, ..., argi) {
    ...
}
```

Bound Variables: Examples

Lets:

```
let x = 2 + 2 in x * 6
```

Bound Variables: Examples

Lets:

```
let x = 2 + 2 in x * 6
```

```
let x be 2 + 2 in x * 6
```

Abstract Binding Trees: Example

Lets:

```
let x = 2 + 2 in x * 6
```

```
let x be 2 + 2 in x * 6
```

```
let(2 + 2, x.x * 6) : Exp

  ar(let) = (Exp, (Exp)Exp)
```

Abstract Binding Trees: Example

Lets:

```
let x = 2 + 2 in x * 6
```

```
let x be 2 + 2 in x * 6
```

```
let(2 + 2, x.x * 6) : Exp

  ar(let) = (Exp, (Exp)Exp)
```

Valence

Abstract Binding Trees: Valence

A **valence** of the form $(s_1, \ldots, s_i)s$ specifies an argument of sort s which binds i-variables of sorts $s_1, \ldots, s_i$ within it.

- $\overrightarrow{s}$ denotes a sequence of sorts $s_1, \ldots, s_i$
- $\overrightarrow{x}$ denotes a sequence of variables $x_1, \ldots, x_i$
- $\overrightarrow{x}$ has sort $\overrightarrow{s}$ denotes that each x_i of $\overrightarrow{x}$ having sort s_i of $\overrightarrow{s}$.

Abstract Binding Trees: Example

Lets:

```
let(2 + 2, x.x * 6) : Exp

  ar(let) = (Exp, (Exp)Exp)
```

Valence

Abstract Binding Trees Defined(?)

Let S be a finite set of **sorts**, $\{O_s\}_{s \in S}$ be a sort-indexed family of operators o of sort s with arity $\mathsf{ar}(o) = (s_1, \ldots, s_n)$, and $\{\chi_s\}_{s \in S}$ be a sort-indexed family of variables x of sort s. The family $B[\chi] = B[\chi_s]_{s \in S}$ of abstract syntax trees (ASTs) of sort s is defined as follows:

- if $x \in \chi_s$, then $x \in A[\chi]_s$
- if $o \in O_s$ with $\mathsf{ar}(o) = ((\vec{s_1})s_1, \ldots, (\vec{s_n})s_n)$ and if, for each $1 \leq i \leq n$, $\vec{x_i}$ is of sort $\vec{s_i}$ and $a_i \in B[\chi, \vec{x_i}]_{s_i}$, then $o(\vec{x_1} \,.\, a_1, \ldots, \vec{x_n} \,.\, a_n) \in B[\chi]_s$

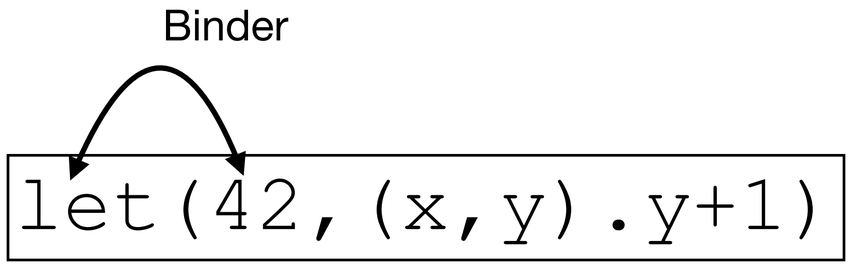
Abstract Binding Trees: Binders

```
let(42,(x,y).y+1)
```

Binders specify the names of bound variables.

Binders in BSTs are denoted by "x."

Abstract Bindings Trees: Binders



Binders specify the names of bound variables.

Binders in BSTs are denoted by "x."

Abstract Binding Trees: Free vs Bound

```
let(42,x.x+1)
```

```
let(42,(x,y).y+1)
```

```
let(42,x
```

```
let(let(42,x.x*2),(x,y).let(x+1,x.y+x+1))
```

Abstract Binding Trees: Free vs Bound

x is bound

```
let(42,x.x+1)
```

x is bound
y is bound

```
let(42,(x,y).y+1)
```

x is bound
y is free

```
let(42,x
```

x is bound

x is bound

x is bound
y is bound

```
let(let(42,x.x*2),(x,y).let(x+1,x.y+x+1))
```

Abstract Binding Trees Defined(?)

What's wrong with this definition?

Let S be a finite set of **sorts**, $\{O_s\}_{s \in S}$ be a sort-indexed family of operators o of sort s with arity $\mathsf{ar}(o) = (s_1, \ldots, s_n)$, and $\{\chi_s\}_{s \in S}$ be a sort-indexed family of variables x of sort s. The family $B[\chi] = B[\chi_s]_{s \in S}$ of abstract syntax trees (ASTs) of sort s is defined as follows:

- if $x \in \chi_s$, then $x \in A[\chi]_s$
- if $o \in O_s$ with $\mathsf{ar}(o) = ((\vec{s_1})s_1, \ldots, (\vec{s_n})s_n)$ and if, for each $1 \leq i \leq n$, $\vec{x_i}$ is of sort $\vec{s_i}$ and $a_i \in B[\chi, \vec{x_i}]_{s_i}$, then $o(\vec{x_1} \,.\, a_1, \ldots, \vec{x_n} \,.\, a_n) \in B[\chi]_s$

Abstract Binding Trees Defined(?)

```
let(plus(2,3),x.let(5,x.times(x
```

Let S be a finite set of **sorts**, $\{O_s\}_{s \in S}$ be a sort-indexed family of operators o of sort s with arity $\mathsf{ar}(o) = (s_1, \ldots, s_n)$, and $\{\chi_s\}_{s \in S}$ be a sort-indexed family of variables x of sort s. The family $B[\chi] = B[\chi_s]_{s \in S}$ of abstract syntax trees (ASTs) of sort s is defined as follows:

- if $x \in \chi_s$, then $x \in A[\chi]_s$
- if $o \in O_s$ with $\mathsf{ar}(o) = ((\vec{s_1})s_1, \ldots, (\vec{s_n})s_n)$ and if, for each $1 \leq i \leq n$, $\vec{x_i}$ is of sort $\vec{s_i}$ and $a_i \in B[\chi, \vec{x_i}]_{s_i}$, then $o(\vec{x_1} \,.\, a_1, \ldots, \vec{x_n} \,.\, a_n) \in B[\chi]_s$

Abstract Binding Trees Defined(!)

Let S be a finite set of **sorts**, $\{O_s\}_{s \in S}$ be a sort-indexed family of operators o of sort s with arity $\mathsf{ar}(o) = (s_1, \ldots, s_n)$, and $\{\chi_s\}_{s \in S}$ be a sort-indexed family of variables x of sort s. The family $B[\chi] = B[\chi_s]_{s \in S}$ of abstract syntax trees (ASTs) of sort s is defined as follows:

- if $x \in \chi_s$, then $x \in A[\chi]_s$
- if $o \in O_s$ with $\mathsf{ar}(o) = ((\vec{s_1})s_1, \ldots, (\vec{s_n})s_n)$ and if, for each $1 \leq i \leq n$ and for each renaming $\pi_i : \vec{x} \leftrightarrow \vec{x}'$, where $\vec{x}' \notin \mathcal{X}$, and $\vec{x_i}$ is of sort $\vec{s_i}$ and $\pi_i \cdot a_i \in B[\chi, \vec{x_i'}]_{s_i}$, then $o(\vec{x_1} . a_1, \ldots, \vec{x_n} . a_n) \in B[\chi]_s$

Abstract Binding Trees Defined(!)

```
let(plus(2,3),x.let(5,x.times(x

let(plus(2,3),y.let(5,z.times(z
```

Let S be a finite set of **sorts**, $\{O_s\}_{s \in S}$ be a sort-indexed family of operators o of sort s with arity $\mathsf{ar}(o) = (s_1, \ldots, s_n)$, and $\{\chi_s\}_{s \in S}$ be a sort-indexed family of variables x of sort s. The family $B[\chi] = B[\chi_s]_{s \in S}$ of abstract syntax trees (ASTs) of sort s is defined as follows:

- if $x \in \chi_s$, then $x \in A[\chi]_s$
- if $o \in O_s$ with $\mathsf{ar}(o) = ((\overrightarrow{s_1})s_1, \ldots, (\overrightarrow{s_n})s_n)$ and if, for each $1 \leq i \leq n$ and for each renaming $\pi_i : \overrightarrow{x_1} \leftrightarrow \overrightarrow{x_i}'$, where $\overrightarrow{x_i}' \notin \mathcal{X}$, and $\overrightarrow{x_i'}$ is of sort $\overrightarrow{s_i}$ and $\pi_i \cdot a_i \in B[\chi, \overrightarrow{x_i'}]_{s_i}$, then $o(\overrightarrow{x_1} . a_1, \ldots, \overrightarrow{x_n} . a_n) \in B[\chi]_s$

Abstract Bindings Trees: Structural Induction

To show that a property P for every BST it suffices to show $\mathsf{P}[\mathcal{X}](a)$ holds for every $a \in B[\mathcal{X}]$, which holds when:

1. (Base Case) if $x \in \chi_s$, then $\mathsf{P}[\mathcal{X}]_s(x)$, and
2. (Step Case) if $o \in O_s$ with $\mathsf{ar}(o) = ((\vec{s_1})s_1, \ldots, ((\vec{s_n})s_n)$, and foreach $1 \leq i \leq n$, we have $\mathsf{P}[\mathcal{X}, \vec{x_i'}]_{s_i}(\pi_i \cdot a_i)$ hold for every renaming $\pi_i : \vec{x_i} \leftrightarrow \vec{x_i'}$, then $\mathsf{P}[\mathcal{X}]_s(o(\vec{x_1} \,.\, a_1, \ldots, \vec{x_n} \,.\, a_n))$ holds.

Abstract Binding Trees: Substitution

Substitution $[b_1/x]b_2 \in B[\mathcal{X}]_{s_2}$ on any AST $B[\mathcal{X}]$ where x is a variable of sort s_1, $b_1 \in A[\mathcal{X}]_{s_1}$, $b_2 \in A[\mathcal{X}, x]_{s_2}$ is defined as follows:

1. $[b_1/x]x = b_1$
2. $[b_1/x]y = y$, when $x \neq y$
3. $[b_1/x]o(\overrightarrow{x_1}.a_1, \ldots, \overrightarrow{x_n}.a_n) = o(\overrightarrow{x_1}.a_1', \ldots, \overrightarrow{x_n}.a_n')$, where for each $1 \leq i \leq n$, we require that $\overrightarrow{x_i} \notin \mathrm{FV}(b)$, and we set $a_i' = [b_1/x]a_i$ if $x \notin \overrightarrow{x_i}$ and $a_i' = a_i$ otherwise.

Abstract Binding Trees: Substitution

Substitution $[b_1/x]b_2 \in B[\mathcal{X}]_{s_2}$ on any AST $B[\mathcal{X}]$ where x is a variable of sort s_1, $b_1 \in A[\mathcal{X}]_{s_1}$, $b_2 \in A[\mathcal{X}, x]_{s_2}$ is defined as follows:

1. $[b_1/x]x = b_1$
2. $[b_1/x]y = y$, when $x \neq y$
3. $[b_1/x]o(\vec{x_1}.a_1, \ldots, \vec{x_n}.a_n) = o(\vec{x_1}.a'_1, \ldots, \vec{x_n}.a'_n)$, where for each $1 \leq i \leq n$, we require that $\vec{x_i} \notin \mathrm{FV}(b)$, and we set $a'_i = [b_1/x]a_i$ if $x \notin \vec{x_i}$ and $a'_i = a_i$ otherwise.

The capture avoidance condition.

Abstract Binding Trees: α-Equivalence

$a =_\alpha b$:

1. $x =_\alpha x$
2. $o(\overrightarrow{x_1}.a_1, \ldots, \overrightarrow{x_n}.a_n) =_\alpha o(\overrightarrow{x_1}.a'_1, \ldots, \overrightarrow{x_n}.a'_n)$, if for every $1 \leq i \leq n$, $\pi_i \cdot a_i =_\alpha \pi'_i \cdot a'_i$ for all fresh renaming $\pi_i : \overrightarrow{x_i} \leftrightarrow \overrightarrow{z_i}$ and $\pi'_i : \overrightarrow{x'_i} \leftrightarrow \overrightarrow{z_i}$.

All BSTs are identified up to α-equivalence.