We will be using finite trees as our core programming construct.

Q: Why trees? When I code I write on text!
A: I don't want to worry about parsing!

We define tree languages with Grammars!
$0 \mid 0 \mid \cdots \mid 0$,

Every $a$ in any $S^k_0$ for some $k \geq 0$.

**Ex.**

$m := Tm$

$\mid Vm$

The language has no elements / is uninhabited.

$L(m) = \emptyset$
Language for addition

Expression set and trees defined by:

\[ e := \begin{cases} \frac{1}{e} & \text{if } e \neq 1 \\ 0 & \text{if } e = 1 \end{cases} \]

0 1
0.1

2 + 3

2 + (3 + 4)

Language for polynomials
poly

\[ p = \deg p \]
\[ \forall \alpha \in \mathbb{R}, \neq 0 \]
\[ v = \alpha^2 \]
\[ \text{below new term} \]

\[ 3x^2 + 4x + 2 \]
\[2x^2 - 2x^2\]
\[\Rightarrow\]
\[2x^2 + (-2x^2)\]
\[\Rightarrow\]
\[\frac{\tilde{\kappa} \cdot x \cdot 1}{\frac{1}{\tilde{\kappa}}} = \tilde{\kappa}\]
We are using algebraic notation to express...

"easily" human-readable fashion

Proof by Structural Induction

A proof by structural induction is a more general type of induction than over the natural numbers. Instead, we induct over the structure of the terms.

\[ n : \text{nat} \Rightarrow S^n_0 \]

Theorem: If \( n \in \text{nat} \), \( 0 \) is a subtree of \( n \).

Proof by structural induction!
Case 1: \( n = 0 \)
0 is a subtree of \( 0 \) \( \checkmark \)

Case 2: \( n = S n' \)
by IH, 0 is a subtree of \( n' \) \( \subseteq n' \), \( n' \) is a subtree of \( n \)

\( \checkmark \)

\( 0 \) is a subtree of \( n \)

\( \checkmark \)

\[ \alpha = b + b \]

\[ 10 \]
\begin{equation}
\begin{aligned}
b &= a \cdot a \\
    &= a - a \\
    &= 0
\end{aligned}
\end{equation}

**Thm:** \forall a \in \text{alphabet}, \ a \ has \ \emptyset \ as \ a \ \text{subtree}

**Lemma:** \forall a \in \text{alphabet}, \ a \ has \ \emptyset \ as \ a \ \text{subtree}

\forall b \in \mathbb{B}, \ b \ has \ \emptyset \ as \ a \ \text{subtree}

**Proof:** by structural induction!

**Case:** \( a = b \cdot b \)

by IH \( b_1 \) and \( b_2 \) has \ \emptyset \ as \ a \ \text{subtree}

\( b \) is a subtree of \( a \)

transitivity of subtree

\( a \) has \ \emptyset \ as \ a \ \text{subtree}

**Case:** \( a = 0 \)

\( 0 \) is a subtree of \( 0 \)
Case: \( b = a, -a \)

By IH, \( a \) and \( a \) has 0 as a subtee

Case: \( b = \emptyset \)

\( \emptyset \) is a subtee of 0

---

Inductive Relation

Syntax is not very interesting. We wish to add meaning to our syntax.
\[ n \in \text{nat} \quad n := 0 \quad | \quad \text{S\text{n}} \]

\[ [ n ] = i \quad \text{when} \quad n \in \text{nat} \quad \text{and} \quad i \in \mathbb{N} \]

\[ [\ldots] = \ldots \subseteq \text{nat} \times \mathbb{N} \]

\[ \prod_{i} = 0_{\mathbb{N}} \quad \text{conclusion} \]

\[ \text{premises} \]
\[ L = i \]

Let \[ L = i + 1 \]

\( i \) is the price of syntax \( n \) relates to \( I \).

Then the syntax \( S_n \) relates to \( i + 1 \).

\[ 0 = 0 \]

\[ \text{S} = 1 \]

\[ S_S = 2 \]

\[ SSSS = 3 \]

\[ SSSSSS = 4 \]
- I can make