CS 430/530
Formal Semantics

Zhong Shao

Yale University
Department of Computer Science

Judgments and Rules
February 28, 2025

Abstract Syntax Tree (AST)

An ordered tree whose leaves are variables; and whose interior nodes are operators whose arguments are its children

- A variable of a specified sort
- An operator of sort s with arguments of sorts $s_1, \ldots, s_n$

As an example, consider a language of arithmetic expressions built from numbers, addition, and multiplication. The abstract syntax of such a language consists of a single sort Exp generated by these operators:

1. An operator $\texttt{num}[n]$ of sort Exp for each $n \in \mathbb{N}$.
2. Two operators, `plus` and `times`, of sort Exp, each with two arguments of sort Exp.

The expression $2 + (3 \times x)$, which involves a variable, x, would be represented by the ast

$$\texttt{plus}(\texttt{num}[2]; \texttt{times}(\texttt{num}[3]; x))$$

Structural Induction over AST

The tree structure of ast's provides a very useful principle of reasoning, called *structural induction*. Suppose that we wish to prove that some property $\mathcal{P}(a)$ holds for all ast's a of a given sort. To show this, it is enough to consider all the ways in which a can be generated and show that the property holds in each case under the assumption that it holds for its constituent ast's (if any). So, in the case of the sort Exp just described, we must show

1. The property holds for any variable x of sort Exp: prove that $\mathcal{P}(x)$.
2. The property holds for any number, $\texttt{num}[n]$: for every $n \in \mathbb{N}$, prove that $\mathcal{P}(\texttt{num}[n])$.
3. Assuming that the property holds for a_1 and a_2, prove that it holds for $\texttt{plus}(a_1; a_2)$ and $\texttt{times}(a_1; a_2)$: if $\mathcal{P}(a_1)$ and $\mathcal{P}(a_2)$, then $\mathcal{P}(\texttt{plus}(a_1; a_2))$ and $\mathcal{P}(\texttt{times}(a_1; a_2))$.

Formal Definition of AST

For the sake of precision, we now give precise definitions of these concepts. Let $\mathcal{S}$ be a finite set of sorts. For a given set $\mathcal{S}$ of sorts, an *arity* has the form $(s_1, \ldots, s_n)s$, which specifies the sort $s \in \mathcal{S}$ of an operator taking $n \geq 0$ arguments, each of sort $s_i \in \mathcal{S}$. Let $\mathcal{O} = \{\mathcal{O}_\alpha\}$ be an arity-indexed family of disjoint sets of *operators* $\mathcal{O}_\alpha$ of arity α. If o is an operator of arity $(s_1, \ldots, s_n)s$, we say that o has sort s and has n arguments of sorts $s_1, \ldots, s_n$.

Fix a set $\mathcal{S}$ of sorts and an arity-indexed family $\mathcal{O}$ of sets of operators of each arity. Let $\mathcal{X} = \{\mathcal{X}_s\}_{s \in \mathcal{S}}$ be a sort-indexed family of disjoint finite sets $\mathcal{X}_s$ of *variables* x of sort s. When $\mathcal{X}$ is clear from context, we say that a variable x is of sort s if $x \in \mathcal{X}_s$, and we say that x is *fresh for* $\mathcal{X}$, or just *fresh* when $\mathcal{X}$ is understood, if $x \notin \mathcal{X}_s$ for any sort s. If x is fresh for $\mathcal{X}$ and s is a sort, then $\mathcal{X}, x$ is the family of sets of variables obtained by adding x to $\mathcal{X}_s$. The notation is ambiguous in that the sort s is not explicitly stated but determined from context.

Formal Definition of AST

The family $\mathcal{A}[\mathcal{X}] = \{\, \mathcal{A}[\mathcal{X}]_s \,\}_{s \in \mathcal{S}}$ of *abstract syntax trees*, or *ast's*, of sort s is the smallest family satisfying the following conditions:

1. A variable of sort s is an ast of sort s: if $x \in \mathcal{X}_s$, then $x \in \mathcal{A}[\mathcal{X}]_s$.
2. Operators combine ast's: if o is an operator of arity $(s_1, \ldots, s_n)s$, and if $a_1 \in \mathcal{A}[\mathcal{X}]_{s_1}$, $\ldots$, $a_n \in \mathcal{A}[\mathcal{X}]_{s_n}$, then $o(a_1; \ldots; a_n) \in \mathcal{A}[\mathcal{X}]_s$.

It follows from this definition that the principle of *structural induction* can be used to prove that some property $\mathcal{P}$ holds of every ast. To show $\mathcal{P}(a)$ holds for every $a \in \mathcal{A}[\mathcal{X}]$, it is enough to show:

1. If $x \in \mathcal{X}_s$, then $\mathcal{P}_s(x)$.
2. If o has arity $(s_1, \ldots, s_n)s$ and $\mathcal{P}_{s_1}(a_1)$ and $\ldots$ and $\mathcal{P}_{s_n}(a_n)$, then $\mathcal{P}_s(o(a_1; \ldots; a_n))$.

Substitution over AST

Variables are given meaning by *substitution*. If $a \in \mathcal{A}[\mathcal{X}, x]_{s'}$, and $b \in \mathcal{A}[\mathcal{X}]_s$, then $[b/x]a \in \mathcal{A}[\mathcal{X}]_{s'}$ is the result of substituting b for every occurrence of x in a. The ast a is called the *target*, and x is called the *subject*, of the substitution. Substitution is defined by the following equations:

1. $[b/x]x = b$ and $[b/x]y = y$ if $x \neq y$.
2. $[b/x]o(a_1; \ldots; a_n) = o([b/x]a_1; \ldots; [b/x]a_n)$.

For example, we may check that

$$[\texttt{num}[2]/x]\texttt{plus}(x; \texttt{num}[3]) = \texttt{plus}(\texttt{num}[2]; \texttt{num}[3]).$$

We may prove by structural induction that substitution on ast's is well-defined.

Theorem 1.1. *If $a \in \mathcal{A}[\mathcal{X}, x]$, then for every $b \in \mathcal{A}[\mathcal{X}]$ there exists a unique $c \in \mathcal{A}[\mathcal{X}]$ such that $[b/x]a = c$*

Abstract Binding Tree (ABT)

- An ast that can introduce new variables and symbols, called a binding with a scope (a range within which the bound identifier can be used)

As a motivating example, consider the expression `let` x `be` a_1 `in` a_2, which introduces a variable x for use within the expression a_2 to stand for the expression a_1. The variable x is bound by the `let` expression for use within a_2; any use of x within a_1 refers to a different variable that happens to have the same name. For example, in the expression `let` x `be` 7 `in` $x + x$ occurrences of x in the addition refer to the variable introduced by the `let`. On the other hand, in the expression `let` x `be` $x * x$ `in` $x + x$, occurrences of x within the multiplication refer to a different variable than those occurring within the addition. The

Abstract Binding Tree (ABT)

- An ordered tree whose leaves are variables; and whose interior nodes are operators whose arguments are its children
  - A variable of a specified sort
  - An operator of sort s with arguments of *generalized sorts* (or valences) $v_1,\ldots, v_n$ where a valence v has the form $s_1,\ldots s_k.s'$

Thus, to specify that the operator `let` has arity (Exp, Exp.Exp)Exp indicates that it is of sort Exp whose first argument is of sort Exp and binds no variables and whose second argument is also of sort Exp and within which is bound one variable of sort Exp. The informal expression $\texttt{let}\, x\, \texttt{be}\, 2+2\, \texttt{in}\, x \times x$ may then be written as the abt

$$\texttt{let}(\texttt{plus}(\texttt{num}[2];\texttt{num}[2]);x.\texttt{times}(x;x))$$

in which the operator `let` has two arguments, the first of which is an expression, and the second of which is an abstractor that binds one expression variable.

Abstract Binding Tree (ABT)

Fix a set $\mathcal{S}$ of sorts and a family $\mathcal{O}$ of disjoint sets of operators indexed by their generalized arities. For a given family of disjoint sets of variables $\mathcal{X}$, the family of *abstract binding trees*, or *abt's* $\mathcal{B}[\mathcal{X}]$, is defined similarly to $\mathcal{A}[\mathcal{X}]$, except that $\mathcal{X}$ is not fixed throughout the definition but rather changes as we enter the scopes of abstractors.

1. If $x \in \mathcal{X}_s$, then $x \in \mathcal{B}[\mathcal{X}]_s$.
2. For each operator o of arity $(\vec{s}_1.s_1, \ldots, \vec{s}_n.s_n)s$, if $a_1 \in \mathcal{B}[\mathcal{X}, \vec{x}_1]_{s_1}$, $\ldots$, and $a_n \in \mathcal{B}[\mathcal{X}, \vec{x}_n]_{s_n}$, then $o(\vec{x}_1.a_1; \ldots; \vec{x}_n.a_n) \in \mathcal{B}[\mathcal{X}]_s$.

Abstract Binding Tree (ABT)

Fix a set $\mathcal{S}$ of sorts and a family $\mathcal{O}$ of disjoint sets of operators indexed by their generalized arities. For a given family of disjoint sets of variables $\mathcal{X}$, the family of *abstract binding trees*, or *abt's* $\mathcal{B}[\mathcal{X}]$, is defined similarly to $\mathcal{A}[\mathcal{X}]$, except that $\mathcal{X}$ is not fixed throughout the definition but rather changes as we enter the scopes of abstractors.

1. If $x \in \mathcal{X}_s$, then $x \in \mathcal{B}[\mathcal{X}]_s$.
2. For each operator o of arity $(\vec{s}_1.s_1, \ldots, \vec{s}_n.s_n)s$, if $a_1 \in \mathcal{B}[\mathcal{X}, \vec{x}_1]_{s_1}$, $\ldots$, and $a_n \in \mathcal{B}[\mathcal{X}, \vec{x}_n]_{s_n}$, then $o(\vec{x}_1.a_1; \ldots; \vec{x}_n.a_n) \in \mathcal{B}[\mathcal{X}]_s$.

fresh renamings, which are bijections between sequences of variables. Specifically, a fresh renaming (relative to $\mathcal{X}$) of a finite sequence of variables $\vec{x}$ is a bijection $\rho : \vec{x} \leftrightarrow \vec{x}'$ between $\vec{x}$ and $\vec{x}'$, where $\vec{x}'$ is fresh for $\mathcal{X}$. We write $\widehat{\rho}(a)$ for the result of replacing each occurrence of x_i in a by $\rho(x_i)$, its fresh counterpart.

For each operator o of arity $(\vec{s}_1.s_1, \ldots, \vec{s}_n.s_n)s$, if for each $1 \leq i \leq n$ and each fresh renaming $\rho_i : \vec{x}_i \leftrightarrow \vec{x}_i'$, we have $\widehat{\rho_i}(a_i) \in \mathcal{B}[\mathcal{X}, \vec{x}_i']$, then $o(\vec{x}_1.a_1; \ldots; \vec{x}_n.a_n) \in \mathcal{B}[\mathcal{X}]_s$.

Abstract Binding Tree (ABT)

The principle of structural induction extends to abt's and is called *structural induction modulo fresh renaming*. It states that to show that $\mathcal{P}[\mathcal{X}](a)$ holds for every $a \in \mathcal{B}[\mathcal{X}]$, it is enough to show the following:

1. if $x \in \mathcal{X}_s$, then $\mathcal{P}[\mathcal{X}]_s(x)$.
2. For every o of arity $(\vec{s}_1.s_1, \ldots, \vec{s}_n.s_n)s$, if for each $1 \leq i \leq n$, $\mathcal{P}[\mathcal{X}, \vec{x}_i']_{s_i}(\widehat{\rho_i}(a_i))$ holds for every $\rho_i : \vec{x}_i \leftrightarrow \vec{x}_i'$ with $\vec{x}_i' \notin \mathcal{X}$, then $\mathcal{P}[\mathcal{X}]_s(o(\vec{x}_1.a_1; \ldots; \vec{x}_n.a_n))$.

The second condition ensures that the inductive hypothesis holds for *all* fresh choices of bound variable names, and not just the ones actually given in the abt.

The relation $a =_\alpha b$ of *α-equivalence* (so-called for historical reasons) means that a and b are identical up to the choice of bound variable names. The α-equivalence relation is the strongest congruence containing the following two conditions:

1. $x =_\alpha x$.
2. $o(\vec{x}_1.a_1; \ldots; \vec{x}_n.a_n) =_\alpha o(\vec{x}_1'.a_1'; \ldots; \vec{x}_n'.a_n')$ if for every $1 \leq i \leq n$, $\widehat{\rho_i}(a_i) =_\alpha \widehat{\rho_i'}(a_i')$ for all fresh renamings $\rho_i : \vec{x}_i \leftrightarrow \vec{z}_i$ and $\rho_i' : \vec{x}_i' \leftrightarrow \vec{z}_i$.

Judgments

We start with the notion of a *judgment*, or *assertion*, about an abstract binding tree. We shall make use of many forms of judgment, including examples such as these:

n nat	n is a natural number
$n_1 + n_2 = n$	n is the sum of n_1 and n_2
τ type	τ is a type
$e : \tau$	expression e has type τ
$e \Downarrow v$	expression e has value v

A judgment states that one or more abstract binding trees have a property or stand in some relation to one another. The property or relation itself is called a *judgment form*, and the judgment that an object or objects have that property or stand in that relation is said to be an *instance* of that judgment form. A judgment form is also called a *predicate*, and the objects constituting an instance are its *subjects*. We write a J or J a, for the judgment asserting that J holds of the abt a. Correspondingly, we sometimes notate the judgment form J by $-$ J, or J $-$, using a dash to indicate the absence of an argument to J. When it is

Inference Rules

An *inductive definition* of a judgment form consists of a collection of *rules* of the form

$$\frac{J_1 \quad \ldots \quad J_k}{J} \tag{2.1}$$

$$\frac{}{\texttt{zero nat}} \qquad \frac{}{\texttt{empty tree}}$$

$$\frac{a\ \texttt{nat}}{\texttt{succ}(a)\ \texttt{nat}} \qquad \frac{a_1\ \texttt{tree} \quad a_2\ \texttt{tree}}{\texttt{node}(a_1;a_2)\ \texttt{tree}}$$

Derivations

To show that an inductively defined judgment holds, it is enough to exhibit a *derivation* of it. A derivation of a judgment is a finite composition of rules, starting with axioms and ending with that judgment. It can be thought of as a tree in which each node is a rule whose children are derivations of its premises. We sometimes say that a derivation of J is evidence for the validity of an inductively defined judgment J.

We usually depict derivations as trees with the conclusion at the bottom, and with the children of a node corresponding to a rule appearing above it as evidence for the premises of that rule. Thus, if

$$\frac{J_1 \quad \ldots \quad J_k}{J}$$

is an inference rule and $\nabla_1, \ldots, \nabla_k$ are derivations of its premises, then

$$\frac{\nabla_1 \quad \ldots \quad \nabla_k}{J}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{zero nat}}}{\texttt{succ(zero) nat}}}{\texttt{succ(succ(zero)) nat}}}{\texttt{succ(succ(succ(zero))) nat}}\,.$$

$$\dfrac{\dfrac{\dfrac{}{\texttt{empty tree}} \quad \dfrac{}{\texttt{empty tree}}}{\texttt{node(empty;empty) tree}} \quad \dfrac{}{\texttt{empty tree}}}{\texttt{node(node(empty;empty);empty) tree}}\,.$$

Rule Induction

Because an inductive definition specifies the *strongest* judgment form closed under a collection of rules, we may reason about them by *rule induction*. The principle of rule induction states that to show that a property $a\ \mathcal{P}$ holds whenever $a\ \mathsf{J}$ is derivable, it is enough to show that $\mathcal{P}$ is *closed under*, or *respects*, the rules defining the judgment form J. More precisely, the property $\mathcal{P}$ respects the rule

$$\frac{a_1\ \mathsf{J} \quad \ldots \quad a_k\ \mathsf{J}}{a\ \mathsf{J}}$$

if $\mathcal{P}(a)$ holds whenever $\mathcal{P}(a_1), \ldots, \mathcal{P}(a_k)$ do. The assumptions $\mathcal{P}(a_1), \ldots, \mathcal{P}(a_k)$ are called the *inductive hypotheses*, and $\mathcal{P}(a)$ is called the *inductive conclusion* of the inference.

Iterated Inductive Definitions

Inductive definitions are often *iterated*, meaning that one inductive definition builds on top of another. In an iterated inductive definition, the premises of a rule

$$\frac{J_1 \quad \ldots \quad J_k}{J}$$

may be instances of either a previously defined judgment form, or the judgment form being defined. For example, the following rules define the judgment form $-$ list, which states that a is a list of natural numbers:

$$\frac{}{\texttt{nil}\ \mathsf{list}} \tag{2.7a}$$

$$\frac{a\ \mathsf{nat} \quad b\ \mathsf{list}}{\texttt{cons}(a;b)\ \mathsf{list}} \tag{2.7b}$$

Simultaneous Inductive Definitions

Frequently two or more judgments are defined at once by a *simultaneous inductive definition*. A simultaneous inductive definition consists of a set of rules for deriving instances of several different judgment forms, any of which may appear as the premise of any rule.

$$\frac{}{\texttt{zero even}} \tag{2.8a}$$

$$\frac{b\ \text{odd}}{\texttt{succ}(b)\ \text{even}} \tag{2.8b}$$

$$\frac{a\ \text{even}}{\texttt{succ}(a)\ \text{odd}} \tag{2.8c}$$

The principle of rule induction for these rules states that to show simultaneously that $\mathcal{P}(a)$ whenever a even and $\mathcal{Q}(b)$ whenever b odd, it is enough to show the following:

1. $\mathcal{P}(\texttt{zero})$;
2. if $\mathcal{Q}(b)$, then $\mathcal{P}(\texttt{succ}(b))$;
3. if $\mathcal{P}(a)$, then $\mathcal{Q}(\texttt{succ}(a))$.

Hypothetical Judgments: Derivability

- Rules for expressing the validity of a conclusion conditional on the validity of one or more hypotheses.

For a given set $\mathcal{R}$ of rules, we define the *derivability* judgment, written $J_1, \ldots, J_k \vdash_{\mathcal{R}} K$, where each J_i and K are basic judgments, to mean that we may derive K from the *expansion* $\mathcal{R} \cup \{ J_1, \ldots, J_k \}$ of the rules $\mathcal{R}$ with the axioms

$$\frac{}{J_1} \quad \ldots \quad \frac{}{J_k}.$$

We use capital Greek letters, usually Γ or Δ, to stand for a finite set of basic judgments, and write $\mathcal{R} \cup \Gamma$ for the expansion of $\mathcal{R}$ with an axiom corresponding to each judgment in Γ. The judgment $\Gamma \vdash_{\mathcal{R}} K$ means that K is derivable from rules $\mathcal{R} \cup \Gamma$, and the judgment $\vdash_{\mathcal{R}} \Gamma$ means that $\vdash_{\mathcal{R}} J$ for each J in Γ. An equivalent way of defining $J_1, \ldots, J_n \vdash_{\mathcal{R}} J$ is to say that the rule

$$\frac{J_1 \quad \ldots \quad J_n}{J} \tag{3.1}$$

is *derivable* from $\mathcal{R}$, which means that there is a derivation of J composed of the rules in $\mathcal{R}$ augmented by treating $J_1, \ldots, J_n$ as axioms.

Hypothetical Judgments: Derivability

Theorem 3.1 (Stability). *If* $\Gamma \vdash_{\mathcal{R}} J$, *then* $\Gamma \vdash_{\mathcal{R} \cup \mathcal{R}'} J$.

Reflexivity Every judgment is a consequence of itself: $\Gamma, J \vdash_{\mathcal{R}} J$. Each hypothesis justifies itself as conclusion.

Weakening If $\Gamma \vdash_{\mathcal{R}} J$, then $\Gamma, K \vdash_{\mathcal{R}} J$. Entailment is not influenced by un-exercised options.

Transitivity If $\Gamma, K \vdash_{\mathcal{R}} J$ and $\Gamma \vdash_{\mathcal{R}} K$, then $\Gamma \vdash_{\mathcal{R}} J$. If we replace an axiom by a derivation of it, the result is a derivation of its consequent without that hypothesis.

Hypothetical Judgments: Admissibility

Admissibility, written $\Gamma \models_{\mathcal{R}} J$, is a weaker form of hypothetical judgment stating that $\vdash_{\mathcal{R}} \Gamma$ implies $\vdash_{\mathcal{R}} J$. That is, the conclusion J is derivable from rules $\mathcal{R}$ when the assumptions Γ are all derivable from rules $\mathcal{R}$. In particular if any of the hypotheses are *not* derivable relative to $\mathcal{R}$, then the judgment is *vacuously* true. An equivalent way to define the judgment $J_1, \ldots, J_n \models_{\mathcal{R}} J$ is to state that the rule

$$\frac{J_1 \quad \ldots \quad J_n}{J} \tag{3.5}$$

is *admissible* relative to the rules in $\mathcal{R}$. Given any derivations of $J_1, \ldots, J_n$ using the rules in $\mathcal{R}$, we may build a derivation of J using the rules in $\mathcal{R}$.

For example, the admissibility judgment

$$\mathtt{succ}(a)\ \mathsf{even} \models_{(2.8)} a\ \mathsf{odd} \tag{3.6}$$

is valid, because any derivation of $\mathtt{succ}(a)$ even from rules (2.2) must contain a sub-derivation of a odd from the same rules, which justifies the conclusion. This fact can be

Hypothetical Judgments: Admissibility

Theorem 3.2. *If* $\Gamma \vdash_{\mathcal{R}} J$, *then* $\Gamma \models_{\mathcal{R}} J$.

Reflexivity If J is derivable from the original rules, then J is derivable from the original rules: $J \models_{\mathcal{R}} J$.

Weakening If J is derivable from the original rules assuming that each of the judgments in Γ are derivable from these rules, then J must also be derivable assuming that Γ and K are derivable from the original rules: if $\Gamma \models_{\mathcal{R}} J$, then $\Gamma, K \models_{\mathcal{R}} J$.

Transitivity If $\Gamma, K \models_{\mathcal{R}} J$ and $\Gamma \models_{\mathcal{R}} K$, then $\Gamma \models_{\mathcal{R}} J$. If the judgments in Γ are derivable, so is K, by assumption, and hence so are the judgments in Γ, K, and hence so is J.

Hypothetical Inductive Definitions

A *hypothetical inductive definition* consists of a set of *hypothetical rules* of the following form:

$$\frac{\Gamma\ \Gamma_1 \vdash J_1 \quad \ldots \quad \Gamma\ \Gamma_n \vdash J_n}{\Gamma \vdash J}. \tag{3.9}$$

The hypotheses Γ are the *global hypotheses* of the rule, and the hypotheses Γ_i are the *local hypotheses* of the ith premise of the rule. Informally, this rule states that J is a derivable consequence of Γ when each J_i is a derivable consequence of Γ, augmented with the hypotheses Γ_i. Thus, one way to show that J is derivable from Γ is to show, in turn, that each J_i is derivable from $\Gamma\ \Gamma_i$. The derivation of each premise involves a "context switch" in which we extend the global hypotheses with the local hypotheses of that premise, establishing a new set of global hypotheses for use within that derivation.

Hypothetical Rule Induction

The principle of *hypothetical rule induction* is just the principle of rule induction applied to the formal hypothetical judgment. So to show that $\mathcal{P}(\Gamma \vdash J)$ when $\Gamma \vdash_{\mathcal{R}} J$, it is enough to show that $\mathcal{P}$ is closed under the rules of $\mathcal{R}$ and under the structural rules.[1] Thus, for each rule of the form (3.9), whether structural or in $\mathcal{R}$, we must show that

$$\text{if } \mathcal{P}(\Gamma\ \Gamma_1 \vdash J_1) \text{ and } \ldots \text{ and } \mathcal{P}(\Gamma\ \Gamma_n \vdash J_n), \text{ then } \mathcal{P}(\Gamma \vdash J).$$

Where the structural rules are:

$$\frac{}{\Gamma, J \vdash J} \tag{3.11a}$$

$$\frac{\Gamma \vdash J}{\Gamma, K \vdash J} \tag{3.11b}$$

$$\frac{\Gamma \vdash K \quad \Gamma, K \vdash J}{\Gamma \vdash J} \tag{3.11c}$$

General Judgments

General judgments codify the rules for handling variables in a judgment. As in mathematics in general, a variable is treated as an *unknown*, ranging over a specified set of objects. A *generic* judgment states that a judgment holds for any choice of objects replacing designated variables in the judgment. Another form of general judgment codifies the handling of symbolic parameters. A *parametric* judgment expresses generality over any choice of fresh renamings of designated symbols of a judgment. To keep track of the active variables and symbols in a derivation, we write $\Gamma \vdash^{\mathcal{U};\mathcal{X}}_{\mathcal{R}} J$ to say that J is derivable from Γ according to rules $\mathcal{R}$, with objects consisting of abt's over symbols $\mathcal{U}$ and variables $\mathcal{X}$.

Generic Derivability

Generic derivability judgment is defined by

$$\mathcal{Y} \mid \Gamma \vdash^{\mathcal{X}}_{\mathcal{R}} J \quad \text{iff} \quad \Gamma \vdash^{\mathcal{X}\,\mathcal{Y}}_{\mathcal{R}} J,$$

where $\mathcal{Y} \cap \mathcal{X} = \emptyset$. Evidence for generic derivability consists of a *generic derivation* ∇ involving the variables $\mathcal{X}\,\mathcal{Y}$. So long as the rules are uniform, the choice of $\mathcal{Y}$ does not matter, in a sense to be explained shortly.

For example, the generic derivation ∇,

$$\dfrac{\dfrac{\overline{x\ \mathsf{nat}}}{\mathtt{succ}(x)\ \mathsf{nat}}}{\mathtt{succ}(\mathtt{succ}(x))\ \mathsf{nat}}\ ,$$

is evidence for the judgment

$$x \mid x\ \mathsf{nat} \vdash^{\mathcal{X}}_{(2.2)} \mathtt{succ}(\mathtt{succ}(x))\ \mathsf{nat}$$

provided $x \notin \mathcal{X}$. Any other choice of x would work just as well, as long as all rules are uniform.

Generic Derivability

The generic derivability judgment enjoys the following *structural properties* governing the behavior of variables, provided that $\mathcal{R}$ is uniform.

Proliferation If $\mathcal{Y} \mid \Gamma \vdash^{\mathcal{X}}_{\mathcal{R}} J$, then $\mathcal{Y}, y \mid \Gamma \vdash^{\mathcal{X}}_{\mathcal{R}} J$.

Renaming If $\mathcal{Y}, y \mid \Gamma \vdash^{\mathcal{X}}_{\mathcal{R}} J$, then $\mathcal{Y}, y' \mid [y \leftrightarrow y']\Gamma \vdash^{\mathcal{X}}_{\mathcal{R}} [y \leftrightarrow y']J$ for any $y' \notin \mathcal{X}\,\mathcal{Y}$.

Substitution If $\mathcal{Y}, y \mid \Gamma \vdash^{\mathcal{X}}_{\mathcal{R}} J$ and $a \in \mathcal{B}[\mathcal{X}\,\mathcal{Y}]$, then $\mathcal{Y} \mid [a/y]\Gamma \vdash^{\mathcal{X}}_{\mathcal{R}} [a/y]J$.

Proliferation is guaranteed by the interpretation of rule schemes as ranging over all expansions of the universe. Renaming is built into the meaning of the generic judgment. It is left implicit in the principle of substitution that the substituting abt is of the same sort as the substituted variable.

Generic Inductive Definitions

A *generic inductive definition* admits generic hypothetical judgments in the premises of rules, with the effect of augmenting the variables, as well as the rules, within those premises. A *generic rule* has the form

$$\frac{\mathcal{Y}\,\mathcal{Y}_1 \mid \Gamma\,\Gamma_1 \vdash J_1 \quad \ldots \quad \mathcal{Y}\,\mathcal{Y}_n \mid \Gamma\,\Gamma_n \vdash J_n}{\mathcal{Y} \mid \Gamma \vdash J}\ . \tag{3.12}$$

The variables $\mathcal{Y}$ are the *global variables* of the inference, and, for each $1 \leq i \leq n$, the variables $\mathcal{Y}_i$ are the *local variables* of the ith premise. In most cases, a rule is stated for *all* choices of global variables and global hypotheses. Such rules can be given in *implicit form*,

$$\frac{\mathcal{Y}_1 \mid \Gamma_1 \vdash J_1 \quad \ldots \quad \mathcal{Y}_n \mid \Gamma_n \vdash J_n}{J}\ . \tag{3.13}$$

A generic inductive definition is just an ordinary inductive definition of a family of *formal generic judgments* of the form $\mathcal{Y} \mid \Gamma \vdash J$. Formal generic judgments are identified up to renaming of variables, so that the latter judgment is treated as identical to the judgment $\mathcal{Y}' \mid \widehat{\rho}(\Gamma) \vdash \widehat{\rho}(J)$ for any renaming $\rho : \mathcal{Y} \leftrightarrow \mathcal{Y}'$. If $\mathcal{R}$ is a collection of generic rules, we write $\mathcal{Y} \mid \Gamma \vdash_{\mathcal{R}} J$ to mean that the formal generic judgment $\mathcal{Y} \mid \Gamma \vdash J$ is derivable from rules $\mathcal{R}$.

Generic Rule Induction

When specialized to a set of generic rules, the principle of rule induction states that to show $\mathcal{P}(\mathcal{Y} \mid \Gamma \vdash J)$ when $\mathcal{Y} \mid \Gamma \vdash_{\mathcal{R}} J$, it is enough to show that $\mathcal{P}$ is closed under the rules $\mathcal{R}$. Specifically, for each rule in $\mathcal{R}$ of the form (3.12), we must show that

if $\mathcal{P}(\mathcal{Y}\,\mathcal{Y}_1 \mid \Gamma\,\Gamma_1 \vdash J_1)$ $\ldots$ $\mathcal{P}(\mathcal{Y}\,\mathcal{Y}_n \mid \Gamma\,\Gamma_n \vdash J_n)$ then $\mathcal{P}(\mathcal{Y} \mid \Gamma \vdash J)$.

To ensure that the formal generic judgment behaves like a generic judgment, we must always ensure that the following *structural rules* are admissible:

$$\frac{}{\mathcal{Y} \mid \Gamma, J \vdash J}$$

$$\frac{\mathcal{Y} \mid \Gamma \vdash J}{\mathcal{Y} \mid \Gamma, J' \vdash J}$$

$$\frac{\mathcal{Y} \mid \Gamma \vdash J}{\mathcal{Y}, x \mid \Gamma \vdash J}$$

$$\frac{\mathcal{Y}, x' \mid [x \leftrightarrow x']\Gamma \vdash [x \leftrightarrow x']J}{\mathcal{Y}, x \mid \Gamma \vdash J}$$

$$\frac{\mathcal{Y} \mid \Gamma \vdash J \quad \mathcal{Y} \mid \Gamma, J \vdash J'}{\mathcal{Y} \mid \Gamma \vdash J'}$$

$$\frac{\mathcal{Y}, x \mid \Gamma \vdash J \quad a \in \mathcal{B}[\mathcal{Y}]}{\mathcal{Y} \mid [a/x]\Gamma \vdash [a/x]J}$$

Parametric Derivability

Parametric derivability is defined analogously to generic derivability, albeit by generalizing over symbols, rather than variables. Parametric derivability is defined by

$$\mathcal{V} \parallel \mathcal{Y} \mid \Gamma \vdash_{\mathcal{R}}^{\mathcal{U};\mathcal{X}} J \quad \text{iff} \quad \mathcal{Y} \mid \Gamma \vdash_{\mathcal{R}}^{\mathcal{U}\mathcal{V};\mathcal{X}} J,$$

where $\mathcal{V} \cap \mathcal{U} = \emptyset$. Evidence for parametric derivability consists of a derivation ∇ involving the symbols $\mathcal{V}$. Uniformity of $\mathcal{R}$ ensures that any choice of parameter names is as good as any other; derivability is stable under renaming.

The concept of a generic inductive definition extends to parametric judgments as well. Briefly, rules are defined on formal parametric judgments of the form $\mathcal{V} \parallel \mathcal{Y} \mid \Gamma \vdash J$, with symbols $\mathcal{V}$, as well as variables, $\mathcal{Y}$. Such formal judgments are identified up to renaming of its variables and its symbols to ensure that the meaning is independent of the choice of variable and symbol names.