Chapter 5

Intersection Types

5.1 The Syntax of Intersection Types

We consider the set of types defined by the abstract grammar

$$\theta ::= \rho \mid \theta \to \theta \mid \iota{:}\,\theta \mid \mathrm{ns} \mid \ \theta \,\&\, \theta$$

where the metavariable ι ranges over identifiers, and the metavariable ρ ranges over primitive types. Here $\to$ is the usual operator for constructing types of functions, & is the binary *intersection operator*, and ns (called *nonsense*) denotes the intersection of zero types. The construction $\iota{:}\,\theta$ is meant to denote a type that is isomorphic to θ, but (in an intuitive sense) disjoint from θ. (In programming terminology, one can regard $\iota{:}\,\theta$ as the type of objects or records with a single field named ι ranging over the type θ.)

We write $\theta \leq \theta'$ to indicate that θ is a subtype of θ'. The subtype relation $\leq$ is defined by the following inference rules:

$$\frac{}{\theta \leq \theta} \qquad \frac{\theta \leq \theta' \qquad \theta' \leq \theta''}{\theta \leq \theta''}$$

$$\frac{}{\theta \leq \mathrm{ns}} \qquad \frac{}{\theta_1 \,\&\, \theta_2 \leq \theta_1} \qquad \frac{}{\theta_1 \,\&\, \theta_2 \leq \theta_2}$$

$$\frac{\theta \leq \theta_1 \qquad \theta \leq \theta_2}{\theta \leq \theta_1 \,\&\, \theta_2}$$

$$\frac{\rho \leq_{\text{prim}} \rho'}{\rho \leq \rho'} \qquad \frac{\theta \leq \theta'}{\iota\colon \theta \leq \iota\colon \theta'}$$

$$\frac{\theta'_1 \leq \theta_1 \qquad \theta_2 \leq \theta'_2}{\theta_1 \to \theta_2 \leq \theta'_1 \to \theta'_2}$$

$$\frac{}{\iota\colon \theta_1 \mathbin{\&} \iota\colon \theta_2 \leq \iota\colon (\theta_1 \mathbin{\&} \theta_2)}$$

$$\frac{}{(\theta \to \theta_1) \mathbin{\&} (\theta \to \theta_2) \leq \theta \to (\theta_1 \mathbin{\&} \theta_2)}$$

$$\frac{}{\text{ns} \leq \iota\colon \text{ns}} \qquad \frac{}{\text{ns} \leq \theta \to \text{ns}}$$

where $\leq_{\text{prim}}$ is a given subtype preorder for primitive types. The first two rules insure that $\leq$ is a preorder. It is not, however, a partial order; thus it useful to write $\theta \simeq \theta'$ when θ and θ' are equivalent, i.e. when $\theta \leq \theta'$ and $\theta' \leq \theta$.

The next four rules establish that ns is a greatest type and that $\theta_1 \mathbin{\&} \theta_2$ is a greatest lower bound of θ_1 and θ_2. Note that we say “a” rather than “the”; since we have a preorder rather than a partial order, greatest types and greatest lower bounds are unique only to within equivalence.

The next three laws in the definition of $\leq$ assert that primitive types are related by $\leq_{\text{prim}}$, that the object-type constructor is monotone, and that $\to$ is antimonotone in its first operand and monotone in its second operand. The final four laws can be strengthened to equivalences:

$$\iota\colon (\theta_1 \mathbin{\&} \theta_2) \simeq \iota\colon \theta_1 \mathbin{\&} \iota\colon \theta_2$$

$$\theta \to (\theta_1 \mathbin{\&} \theta_2) \simeq (\theta \to \theta_1) \mathbin{\&} (\theta \to \theta_2)$$

$$\iota\colon \text{ns} \simeq \text{ns} \qquad \theta \to \text{ns} \simeq \text{ns}.$$

The first two of these equivalences show that conjunction distributes with object constructors (modulo $\simeq$) and with the right side (but not the left) of $\to$. The last two equivalences are analogous laws for the conjunction of zero types.

We will not make a specific choice of the primitive types and their subtype relation. However, we will require that the preordered set of primitive types must be pairwise bounded complete, i.e. every pair of primitive types with an upper bound must have a least upper bound.

The subtype relation can be explicated by viewing types as standing for finite sets of "canonical types". A *canonical type* is one of the following:

1. A primitive type,

2. $\iota{:}\,\omega$, where ω is a canonical type.

3. $\sigma \to \omega$, where σ is a finite set of canonical types and ω is a canonical type,

The subtype preorder is defined for both canonical types and finite sets of canonical types by the following simultaneous recursive definition: For canonical types, $\omega \leq \omega'$ iff one of the following holds:

1. There are primitive types ρ and ρ' such that $\omega = \rho$, $\omega' = \rho'$, and $\rho \leq_{\text{prim}} \rho'$,

2. There are ι, σ_1, and σ'_1 such that $\omega = \iota{:}\,\omega_1$, $\omega' = \iota{:}\,\omega'_1$, and $\omega_1 \leq \omega'_1$.

3. There are σ_1, σ'_1, ω_2, and ω'_2 such that $\omega = \sigma_1 \to \omega_2$, $\omega' = \sigma'_1 \to \omega'_2$, $\sigma'_1 \leq \sigma_1$, and $\omega_2 \leq \omega'_2$,

For finite sets of canonical types, $\sigma \leq \sigma'$ iff

$$(\forall \omega' \in \sigma')(\exists \omega \in \sigma)\, \omega \leq \omega'.$$

The function $(-)^*$ maps types into finite sets of canonical types as follows:

$$\rho^* = \{\rho\} \qquad (\iota{:}\,\theta)^* = \{\iota{:}\,\omega \mid \omega \in \theta^*\} \qquad (\theta_1 \to \theta_2)^* = \{\theta_1^* \to \omega_2 \mid \omega_2 \in \theta_2^*\}$$

$$\text{ns}^* = \{\} \qquad (\theta_1 \,\&\, \theta_2)^* = \theta_1^* \cup \theta_2^*.$$

It can be shown that this function preserves the subtype ordering, i.e., that

$$\theta \leq \theta' \text{ iff } \theta^* \leq \theta'^*.$$

We can also map both canonical types and finite sets of canonical types back into ordinary types. For notational simplicity, we use the same notation $(-)^\dagger$ for both mappings:

$$\rho^\dagger = \rho \qquad (\iota{:}\,\omega)^\dagger = \iota{:}\,\omega^\dagger \qquad (\sigma \to \omega)^\dagger = \sigma^\dagger \to \omega^\dagger$$

$$\{\}^\dagger = \text{ns} \qquad \{\omega_1, \ldots, \omega_n\}^\dagger = \omega_1^\dagger \,\&\, \cdots \,\&\, \omega_n^\dagger.$$

It is easy to show that $*$ inverts $\dagger$ (but not vice-versa):

$$(\omega^\dagger)^* = \{\omega\} \qquad (\sigma^\dagger)^* = \sigma,$$

and that, as a consequence,

$$\sigma \leq \sigma' \text{ iff } \sigma^\dagger \leq \sigma'^\dagger.$$

It can also be shown that

Proposition 5.1 *If $\theta_1 \mathbin{\&} \theta_2 \leq \theta'$ then there are θ'_1 and θ'_2 such that $\theta_1 \leq \theta'_1$, $\theta_2 \leq \theta'_2$, and $\theta' \simeq \theta'_1 \mathbin{\&} \theta'_2$.*

Proposition 5.2 *Every pair of types has a least upper bound. In particular, there is a function $\sqcup$ mapping pairs of types into types such that $\theta_1 \sqcup \theta_2$ is a least upper bound of θ_1 and θ_2, and*

$$\theta \sqcup \mathrm{ns} \simeq \mathrm{ns}$$

$$\theta_1 \sqcup (\theta_2 \mathbin{\&} \theta_3) \simeq (\theta_1 \sqcup \theta_2) \mathbin{\&} (\theta_1 \sqcup \theta_3)$$

$$\rho \sqcup \iota\colon \theta \simeq \mathrm{ns}$$

$$\rho \sqcup (\theta_1 \to \theta_2) \simeq \mathrm{ns}$$

$$\iota\colon \theta_1 \sqcup (\theta_2 \to \theta_3) \simeq \mathrm{ns}$$

$$\rho_1 \sqcup \rho_2 \simeq \rho_1 \sqcup_{\mathrm{prim}} \rho_2 \text{ when } \rho_1 \sqcup_{\mathrm{prim}} \rho_2 \text{ exists}$$

$$\rho_1 \sqcup \rho_2 \simeq \mathrm{ns} \text{ when } \rho_1 \sqcup_{\mathrm{prim}} \rho_2 \text{ does not exist}$$

$$\iota\colon \theta_1 \sqcup \iota\colon \theta_2 \simeq \iota\colon (\theta_1 \sqcup \theta_2)$$

$$\iota_1\colon \theta_1 \sqcup \iota_2\colon \theta_2 \simeq \mathrm{ns} \text{ when } \iota_1 \neq \iota_2$$

$$(\theta_1 \to \theta'_1) \sqcup (\theta_2 \to \theta'_2) \simeq (\theta_1 \mathbin{\&} \theta_2) \to (\theta'_1 \sqcup \theta'_2).$$

(As with greatest lower bounds, least upper bounds are unique only to within an equivalence. Note, by the way, that the function $\sqcup$, unlike &, is not a type constructor.)

5.2 The Intrinsic Semantics of Types

In general, the intrinsic semantics of intersection types is based upon a category $\mathcal{K}$ with the following properties:

1. The category $\mathcal{K}$ is Cartesian closed.

2. There is a subcategory $\mathcal{S}$ of $\mathcal{K}$ such that every finite diagram in $\mathcal{S}$ has a limit in $\mathcal{S}$ that is also a limit in $\mathcal{K}$. (In particular, we assume that $\mathcal{S}$ and $\mathcal{K}$ have the same distinguished product.)

3. There is a functor $\underset{\mathcal{S}}{\Rightarrow}$ from $\mathcal{S}^{\text{op}} \times \mathcal{S}$ to $\mathcal{S}$ that is a restriction of the exponentiation functor $\underset{\mathcal{K}}{\Rightarrow}$ from $\mathcal{K}^{\text{op}} \times \mathcal{K}$ to $\mathcal{K}$. (This does not imply that $\mathcal{S}$ is Cartesian closed, since it need not contain application and abstraction morphisms.) For each object k of $\mathcal{S}$, the functor $k \underset{\mathcal{S}}{\Rightarrow} -$ preserves limits in $\mathcal{S}$.

4. There is a functor $[\![-]\!]_{\text{prim}}$ from the preordered set of primitive types (viewed as a category) to $\mathcal{S}$ that assigns meanings to the primitive types ρ and implicit conversions to the morphisms $\rho \leq_{\text{prim}} \rho'$ of the primitive subtype relation.

In the case of domain-theoretic semantics, there are two possibilities:

1. $\mathcal{K}$ is the category whose objects are CPO's and whose morphisms are continuous functions; $\mathcal{S}$ is the subcategory in which the morphisms preserve $\perp$.

2. $\mathcal{K}$ is the category whose objects are DCPO's with glb's of all lower-bounded subsets and whose morphisms are continuous functions; $\mathcal{S}$ is the subcategory in which the morphisms preserve glb's of all lower-bounded subsets. (One may also require the objects of these categories to be continuous or algebraic, and also to be countably based, as in Proposition 4.13.)

(Another possibility, which goes beyond domain theory, and which we will not explore here, takes both $\mathcal{K}$ and $\mathcal{S}$ to be subcategories of the functor category CPO^{Σ} used by F. Oles to describe the semantics of Algol-like languages. Specifically, $\mathcal{K}$ is the full subcategory whose objects are functors that map morphisms into strict functions, $\mathcal{S}$ is the further subcategory whose morphisms are natural transformations that map objects into strict functions, and Σ is an appropriate category of "store shapes".)

In the general case, we can define two functors, $[\![-]\!]_{\mathrm{ct}}$ from the preordered set of canonical types to $\mathcal{S}$, and $[\![-]\!]_{\mathrm{sct}}$ from the preordered set of finite sets of canonical types to $\mathcal{S}$, as follows:

- Primitive Types

$$[\![\rho]\!]_{\mathrm{ct}} = [\![\rho]\!]_{\mathrm{prim}}$$
$$[\![\rho \leq \rho']\!]_{\mathrm{ct}} = [\![\rho \leq_{\mathrm{prim}} \rho']\!]_{\mathrm{prim}}.$$

- Function Types

$$[\![\sigma \to \omega]\!]_{\mathrm{ct}} = [\![\sigma]\!]_{\mathrm{sct}} \underset{\mathcal{S}}{\Rrightarrow} [\![\omega]\!]_{\mathrm{ct}}$$
$$[\![\sigma \to \omega \leq \sigma' \to \omega']\!]_{\mathrm{ct}} = [\![\sigma' \leq \sigma]\!]_{\mathrm{sct}} \underset{\mathcal{S}}{\Rrightarrow} [\![\omega \leq \omega']\!]_{\mathrm{ct}}.$$

  In a domain-theoretic semantics, $[\![\sigma]\!]_{\mathrm{sct}} \underset{\mathcal{S}}{\Rrightarrow} [\![\omega]\!]_{\mathrm{ct}}$ is the pointwise-ordered set of continuous functions from $[\![\sigma]\!]_{\mathrm{sct}}$ to $[\![\omega]\!]_{\mathrm{ct}}$, and

$$[\![\sigma \to \omega \leq \sigma' \to \omega']\!]_{\mathrm{ct}} f = [\![\sigma' \leq \sigma]\!]_{\mathrm{sct}} \,;\, f \,;\, [\![\omega \leq \omega']\!]_{\mathrm{ct}}.$$

- Record Types

$$[\![\iota\colon \omega]\!]_{\mathrm{ct}} = [\![\omega]\!]_{\mathrm{ct}}$$
$$[\![\iota\colon \omega \leq \iota\colon \omega']\!]_{\mathrm{ct}} = [\![\omega \leq \omega']\!]_{\mathrm{ct}}.$$

- Finite Sets of Canonical Types

  Suppose $\sigma = \{\omega_1, \ldots, \omega_m\}$. Let S_σ (called a *shape*) be the set

$$S_\sigma = \sigma \cup \{\omega_i \sqcup \omega_{i'} \mid 1 \leq i < i' \leq m \text{ and } \omega_i \sqcup \omega_{i'} \text{ exists}\},$$

  with an ordering inherited from the preorder of canonical types. Let $\mathcal{J}_\sigma$ (called a *diagram*) be the restriction of the functor $[\![-]\!]_{\mathrm{ct}}$ to S_σ. Then $[\![\sigma]\!]_{\mathrm{sct}}$ is the vertex of the limit of $\mathcal{J}_\sigma$. We write $\pi_i \in [\![\sigma]\!]_{\mathrm{sct}} \underset{\mathcal{S}}{\to} [\![\omega_i]\!]_{\mathrm{ct}}$ for the m morphism parts of this limit. (Usually, we will not refer to the morphism parts from $[\![\sigma]\!]_{\mathrm{ct}}$ to $[\![\omega_i \sqcup \omega_{i'}]\!]_{\mathrm{ct}}$.)

  Now suppose $\sigma \leq \sigma'$, where $\sigma' = \{\omega'_1, \ldots, \omega'_n\}$. For each j in 1 to n, there will be an i_j in 1 to m such that $\omega_{i_j} \leq \omega'_j$. Thus we can define the morphisms

$$\delta_j = \pi_{i_j} \,;\, [\![\omega_{i_j} \leq \omega'_j]\!]_{\mathrm{ct}} \in [\![\sigma]\!]_{\mathrm{sct}} \underset{\mathcal{S}}{\to} [\![\omega'_j]\!]_{\mathrm{ct}}.$$

Of course, there may be more than one member of σ that is a subtype of ω'_j, but this will not make the definition of δ_j ambiguous. For suppose $\omega_{i_j} \leq \omega'_j$ and $\omega_{i'_j} \leq \omega'_j$. Then, since ω_{i_j} and $\omega_{i'_j}$ are bounded (by ω'_j) and thus have a lub,

$$\begin{aligned}
\pi_{i_j} \mathbin{;} [\![\omega_{i_j} \leq \omega'_j]\!]_{\mathrm{ct}} &= \pi_{i_j} \mathbin{;} [\![\omega_{i_j} \leq \omega_{i_j} \sqcup \omega_{i'_j}]\!]_{\mathrm{ct}} \mathbin{;} [\![\omega_{i_j} \sqcup \omega_{i'_j} \leq \omega'_j]\!]_{\mathrm{ct}} \\
&= \pi_{i'_j} \mathbin{;} [\![\omega_{i'_j} \leq \omega_{i_j} \sqcup \omega_{i'_j}]\!]_{\mathrm{ct}} \mathbin{;} [\![\omega_{i_j} \sqcup \omega_{i'_j} \leq \omega'_j]\!]_{\mathrm{ct}} \\
&= \pi_{i'_j} \mathbin{;} [\![\omega_{i'_j} \leq \omega'_j]\!]_{\mathrm{ct}},
\end{aligned}$$

where the second equality holds because π_{i_j} and $\pi_{i'_j}$ are parts of a limit satisfying $\pi_{i_j} \mathbin{;} [\![\omega_{i_j} \leq \omega_{i_j} \sqcup \omega_{i'_j}]\!]_{\mathrm{ct}} = \pi_{i'_j} \mathbin{;} [\![\omega_{i'_j} \leq \omega_{i_j} \sqcup \omega_{i'_j}]\!]_{\mathrm{ct}}$.

In fact, there is a cone for the diagram $\mathcal{J}_{\sigma'}$ whose vertex is $[\![\sigma]\!]_{\mathrm{sct}}$ and whose morphism parts are $\delta_j \in [\![\sigma]\!]_{\mathrm{sct}} \xrightarrow[\mathcal{S}]{} [\![\omega'_j]\!]_{\mathrm{ct}}$. To see this, suppose $\omega'_j, \omega'_{j'} \in \sigma'$ are such that $\omega'_j \sqcup \omega'_{j'}$ exists. Then there are $\omega_{i_j}, \omega_{i_{j'}} \in \sigma$ such that $\omega_{i_j} \leq \omega'_j$ and $\omega_{i_{j'}} \leq \omega'_{j'}$, so that ω_{i_j} and $\omega_{i_{j'}}$ are bounded by $\omega'_j \sqcup \omega'_{j'}$. Thus

$$\begin{aligned}
\delta_j \mathbin{;} [\![\omega'_j \leq \omega'_j \sqcup \omega'_{j'}]\!]_{\mathrm{ct}} &= \pi_{i_j} \mathbin{;} [\![\omega_{i_j} \leq \omega'_j]\!]_{\mathrm{ct}} \mathbin{;} [\![\omega'_j \leq \omega'_j \sqcup \omega'_{j'}]\!]_{\mathrm{ct}} \\
&= \pi_{i_j} \mathbin{;} [\![\omega_{i_j} \leq \omega'_j \sqcup \omega'_{j'}]\!]_{\mathrm{ct}} \\
&= \pi_{i_j} \mathbin{;} [\![\omega_{i_j} \leq \omega_{i_j} \sqcup \omega_{i_{j'}}]\!]_{\mathrm{ct}} \mathbin{;} [\![\omega_{i_j} \sqcup \omega_{i_{j'}} \leq \omega'_j \sqcup \omega'_{j'}]\!]_{\mathrm{ct}} \\
&= \pi_{i_{j'}} \mathbin{;} [\![\omega_{i_{j'}} \leq \omega_{i_j} \sqcup \omega_{i_{j'}}]\!]_{\mathrm{ct}} \mathbin{;} [\![\omega_{i_j} \sqcup \omega_{i_{j'}} \leq \omega'_j \sqcup \omega'_{j'}]\!]_{\mathrm{ct}} \\
&= \pi_{i_{j'}} \mathbin{;} [\![\omega_{i_{j'}} \leq \omega'_j \sqcup \omega'_{j'}]\!]_{\mathrm{ct}} \\
&= \pi_{i_{j'}} \mathbin{;} [\![\omega_{i_{j'}} \leq \omega'_{j'}]\!]_{\mathrm{ct}} \mathbin{;} [\![\omega'_{j'} \leq \omega'_j \sqcup \omega'_{j'}]\!]_{\mathrm{ct}} \\
&= \delta_{j'} \mathbin{;} [\![\omega'_{j'} \leq \omega'_j \sqcup \omega'_{j'}]\!]_{\mathrm{ct}},
\end{aligned}$$

where the fourth equality holds because π_{i_j} and $\pi_{i_{j'}}$ are morphism parts of a limit of $\mathcal{J}_\sigma$.

Finally, we can define $[\![\sigma \leq \sigma']\!]_{\mathrm{sct}}$ to be the unique mediating morphism from the vertex $[\![\sigma]\!]_{\mathrm{sct}}$ of the cone on $\mathcal{J}_{\sigma'}$ to the vertex $[\![\sigma']\!]_{\mathrm{sct}}$ of the limit of $\mathcal{J}_{\sigma'}$.

In a domain-theoretic semantics, the limit is

$$\begin{aligned}
&[\![\{\omega_1, \ldots, \omega_m\}]\!]_{\mathrm{sct}} = \\
&\quad \{\langle x_1, \ldots, x_m\rangle \mid x_1 \in [\![\omega_1]\!]_{\mathrm{ct}}, \ldots, x_m \in [\![\omega_m]\!]_{\mathrm{ct}} \\
&\qquad \text{and } \forall i, i'.\ \text{if } 1 \leq i < i' \leq m \text{ and } \omega_i \sqcup \omega_{i'} \text{ exists then} \\
&\qquad\quad [\![\omega_i \leq \omega_i \sqcup \omega_{i'}]\!]_{\mathrm{ct}} x_i = [\![\omega_{i'} \leq \omega_i \sqcup \omega_{i'}]\!]_{\mathrm{ct}} x_{i'}\},
\end{aligned}$$

and the mediating morphism is given by

$$\begin{aligned}&[\![\{\omega_1,\ldots,\omega_m\} \leq \{\omega'_1,\ldots,\omega'_n\}]\!]_{\mathrm{sct}}\langle x_1,\ldots,x_m\rangle \\ &\qquad = \langle [\![\omega_{i_1} \leq \omega'_1]\!]_{\mathrm{ct}} x_{i_1},\ldots,[\![\omega_{i_n} \leq \omega'_n]\!]_{\mathrm{ct}} x_{i_n}\rangle,\end{aligned}$$

where each ω_{i_j} is a subtype of ω'_j.

(Actually, the above definition of the domain-theoretic meaning of a finite set of canonical types depends upon the order in which the elements of the set appear in the notation $\{\omega_1,\ldots,\omega_n\}$. However, one can resolve this ambiguity by imposing some fixed arbitrary total ordering on the canonical types.)

A *type assignment* π is a mapping from a finite set of identifiers into finite sets of canonical types; the appropriate preordering is that $\pi \leq \pi'$ whenever $\mathrm{dom}\,\pi' \subseteq \mathrm{dom}\,\pi$ and, for all $\iota \in \mathrm{dom}\,\pi'$, $\pi\iota \leq \pi'\iota$. We write $[\![-]\!]_{\mathrm{ta}}$ for the functor from the preordered set of type assignments (viewed as a category) to $\mathcal{S}$, such that

$$[\![\pi]\!]_{\mathrm{ta}} \stackrel{\mathrm{def}}{=} \prod_{\iota\in\mathrm{dom}\,\pi}^{\mathcal{S}} [\![\pi\iota]\!]_{\mathrm{sct}},$$

and $[\![\pi \leq \pi']\!]_{\mathrm{ta}}$ is the unique morphism making the diagram

$$\begin{tikzcd}
\prod\limits_{\iota\in\mathrm{dom}\,\pi}^{\mathcal{S}} [\![\pi\iota]\!]_{\mathrm{sct}} \arrow[r, "\mathsf{p}"] \arrow[d, "{[\![\pi \leq \pi']\!]_{\mathrm{ta}}}"'] & {[\![\pi\iota]\!]_{\mathrm{sct}}} \arrow[d] \\
\prod\limits_{\iota\in\mathrm{dom}\,\pi'}^{\mathcal{S}} [\![\pi'\iota]\!]_{\mathrm{sct}} \arrow[r, "\mathsf{p}'"] & {[\![\pi'\iota]\!]_{\mathrm{sct}}}
\end{tikzcd}$$

(where p and p' denote projections) commute in $\mathcal{S}$ for every $\iota \in \mathrm{dom}\,\pi'$.

5.3 Phrases and their Intrinsic Semantics

We write $\pi \vdash p : \omega$ (or $\pi \vdash p : \sigma$) for the judgement that a phrase p has the canonical type ω (or the finite set of canonical types σ) under the type assignment π. The valid typing judgements of our language are those that are provable from a certain set of inference rules. At the outset, it is important to note that some of these rules, such as the rule for subtyping:

$$\frac{\pi \vdash p : \omega_1}{\pi \vdash p : \omega} \quad \text{when } \omega_1 \leq \omega$$

are not syntax-directed, since they allow one to infer a judgement about a phrase from a previous judgement about the *same* phrase.

In an intrinsic semantics of a typed language, a phrase will have a meaning only when it satisfies a typing judgment, and if it satisfies several typing judgements, it will have a different meaning for each judgement. Thus it is the typing judgement, rather than just the phrase, that is actually mapped into meanings. Specifically,

$$\text{If } \pi \vdash p : \omega \text{ then } [\![\pi \vdash p : \omega]\!] \in [\![\pi]\!]_{\text{ta}} \xrightarrow[\mathcal{K}]{} [\![\omega]\!]_{\text{ct}}.$$

$$\text{If } \pi \vdash p : \sigma \text{ then } [\![\pi \vdash p : \sigma]\!] \in [\![\pi]\!]_{\text{ta}} \xrightarrow[\mathcal{K}]{} [\![\sigma]\!]_{\text{sct}}.$$

Conventionally, one might expect to define these meanings by a collection of semantic equations, one for each "reason" that a phrase satisfies a judgement, i.e. for each inference rule. But as soon as one tries to write a semantic equation for a non-syntax-directed rule, such as the intuitively obvious equation

$$[\![\pi \vdash p : \omega]\!] = [\![\pi \vdash p : \omega_1]\!] \,;\, [\![\omega_1 \leq \omega]\!]_{\text{ct}} \qquad \text{when } \omega_1 \leq \omega$$

for the rule for subtyping, it becomes clear that something is awry: If semantic equations are cases of a definition by induction on the size of phrases, they should not define the meaning of a phrase in terms of another meaning of the same phrase.

In fact, however, our semantic equations will define the meaning of typing judgements, not by structural induction on phrases, but by structural induction on the proofs of judgments. Thus the definition is by induction on the size of the proof, with a case analysis on the rule used at the main step of the proof.

For each inference rule

$$\frac{J_1 \quad J_2 \quad \ldots \quad J_n}{J}$$

the corresponding semantic equation will give the meaning $[\![J]\!]$ of a proof that uses the rule at its main step, as a function of the meanings $[\![J_1]\!]$, ..., $[\![J_n]\!]$ of the subproofs of the judgements that are premisses of the rule.

Unfortunately, there is a complication: The same typing judgement may have several proofs, and there is no a priori reason why these proofs should induce the same meaning. Thus it is important to show that a language is *coherent*, i.e. that all proofs of the same judgment give rise to the same meaning for that judgement. In fact, the language we are describing is coherent, but the proof is beyond the scope of these notes.

5.4 Inference Rules and Semantic Equations

We now give the inference rules for typing judgements and the corresponding semantic equations. In cases where there is a significant difference, we will give semantic equations for both the general category-theoretic case and the more specific domain-theoretic case.

- Subtyping

$$\frac{\pi \vdash p : \omega_1}{\pi \vdash p : \omega} \quad \text{when } \omega_1 \leq \omega \qquad \begin{array}{c} [\![\pi \vdash p : \omega]\!] = [\![\pi \vdash p : \omega_1]\!] \,;\, [\![\omega_1 \leq \omega]\!]_{\mathrm{ct}} \\ [\![\pi \vdash p : \omega]\!]\eta = [\![\omega_1 \leq \omega]\!]_{\mathrm{ct}}([\![\pi \vdash p : \omega_1]\!]\eta) \end{array}$$

- Identifiers

$$\frac{}{\pi \vdash \iota : \omega_i} \quad \text{when } \iota \in \operatorname{dom}\pi \text{ and } \omega_i \in \pi\iota \qquad \begin{array}{c} [\![\pi \vdash \iota : \omega]\!] = \mathsf{p} \,;\, \pi_i \\ [\![\pi \vdash \iota : \omega]\!]\eta = (\eta\iota)_i, \end{array}$$

where p is the projection morphism from $[\![\pi]\!]_{\mathrm{ta}} = \prod^{\mathcal{S}}_{\iota\in\operatorname{dom}\pi}[\![\pi\iota]\!]_{\mathrm{sct}}$ to $[\![\pi\iota]\!]_{\mathrm{sct}}$, and π_i is the vertex-to-component morphism from $[\![\pi\iota]\!]_{\mathrm{sct}}$ to $[\![\omega]\!]_{\mathrm{ct}}$.

- Constants

$$\frac{}{\pi \vdash c : \omega_c} \qquad \begin{array}{c} [\![\pi \vdash c : \omega_c]\!] = u_\pi \,;\, \mu_c \\ [\![\pi \vdash c : \omega_c]\!]\eta = m_c. \end{array}$$

Here u_π is the unique morphism from $[\![\pi]\!]_{\mathrm{ta}}$ to the terminal element of $\mathcal{K}$, ω_c is the type of the constant c, μ_c is its categorical meaning, which must be a morphism from the terminal element to $[\![\omega_c]\!]_{\mathrm{ct}}$, and m_c is its domain-theoretic value.

The semantics of the next two rules relies upon the Cartesian closed nature of $\mathcal{K}$. For all objects B and C there is an *exponentiation* object $B \Rightarrow C$ and an *application* morphism $\mathsf{ap} \in (B \Rightarrow C) \times B \to C$ such that, for all objects A and morphisms $\rho \in A \times B \to C$, there is exactly one *abstraction* morphism $\mathsf{ab}\,\rho \in A \to (B \Rightarrow C)$ making

$$\begin{tikzcd}
A \times B \arrow[r, "\mathsf{ab}\,\rho \times I_B"] \arrow[dr, "\rho"'] & (B \Rightarrow C) \times B \arrow[d, "\mathsf{ap}"] \\
& C
\end{tikzcd}$$

commute in $\mathcal{K}$. Moreover, the exponentiation operator $\Rightarrow$ extends to a functor from $\mathcal{K}^{\mathrm{op}} \times \mathcal{K}$ to $\mathcal{K}$ with the following action on morphisms: if $\phi \in B' \to B$ and $\psi \in C \to C'$ then $\phi \Rightarrow \psi$ is the unique morphism in $(B \Rightarrow C) \to (B' \Rightarrow C')$ making

$$\begin{tikzcd}
(B \Rightarrow C) \times B' \arrow[r, "(\phi \Rightarrow \psi) \times I_{B'}"] \arrow[d, "I_{B \Rightarrow C} \times \phi"'] & (B' \Rightarrow C') \times B' \arrow[dd, "\mathsf{ap}'"] \\
(B \Rightarrow C) \times B \arrow[d, "\mathsf{ap}"'] & \\
C \arrow[r, "\psi"'] & C'
\end{tikzcd}$$

commute in $\mathcal{K}$. Note that the subcategory $\mathcal{S}$ is assumed to be closed under products and exponentiations of both objects and morphisms.

In the domain-theoretic case, $B \Rightarrow C$ is the pointwise-ordered set of continuous functions from B to C, and

$$\mathsf{ap}\langle f, x\rangle = f\,x \qquad \mathsf{ab}\,\rho\,x\,y = \rho\langle x, y\rangle$$

$$(\phi \Rightarrow \psi) f\,x' = \psi(f(\phi x')).$$

- Applications

$$\frac{\pi \vdash p_1 : \sigma \to \omega \qquad \pi \vdash p_2 : \sigma}{\pi \vdash p_1\,p_2 : \omega}$$

$$[\![\pi \vdash p_1\,p_2 : \omega]\!] = ([\![\pi \vdash p_1 : \sigma \to \omega]\!] \otimes [\![\pi \vdash p_2 : \sigma]\!])\,;\,\mathsf{ap}$$

$$[\![\pi \vdash p_1\,p_2 : \omega]\!]\eta = ([\![\pi \vdash p_1 : \sigma \to \omega]\!]\eta)([\![\pi \vdash p_2 : \sigma]\!]\eta),$$

where $(-\otimes-)$ is the mediating morphism from the object $[\![\pi]\!]_{\mathrm{ta}}$ to the product $([\![\sigma]\!]_{\mathrm{sct}} \underset{\mathcal{K}}{\Longrightarrow} [\![\omega]\!]_{\mathrm{ct}}) \times [\![\sigma]\!]_{\mathrm{sct}}$, and ap is the application morphism from this product to $[\![\omega]\!]_{\mathrm{ct}}$.

- Abstractions

$$\frac{[\,\pi \mid \iota{:}\,\sigma\,] \vdash p_1 : \omega}{\pi \vdash \lambda\iota.\ p_1 : \sigma \to \omega}$$

$$[\![\pi \vdash \lambda\iota.\ p_1 : \sigma \to \omega]\!] = \mathsf{ab}(\P\,;[\![[\,\pi \mid \iota{:}\,\sigma\,] \vdash p_1 : \omega]\!])$$

$$[\![\pi \vdash \lambda\iota.\ p_1 : \sigma \to \omega]\!]\eta x = [\![[\,\pi \mid \iota{:}\,\sigma\,] \vdash p_1 : \omega]\!][\,\eta \mid \iota{:}\,x\,].$$

Here $[\,\pi \mid \iota{:}\,\sigma\,]$ denotes the type assignment with domain $\operatorname{dom}\pi \cup \{\iota\}$ that maps ι into σ and all other ι' in its domain into $\pi\iota'$, while $\P$ denotes the unique morphism making

$$\begin{tikzcd} {[\![\pi]\!]_{\mathrm{ta}} \times [\![\sigma]\!]_{\mathrm{sct}}} \arrow[d, "\P"'] \arrow[dr] & \\ {[\![[\,\pi \mid \iota{:}\,\sigma\,]]\!]_{\mathrm{ta}}} \arrow[r] & {[\![\sigma]\!]_{\mathrm{sct}}} \end{tikzcd} \quad \text{and} \quad \begin{tikzcd} {[\![\pi]\!]_{\mathrm{ta}} \times [\![\sigma]\!]_{\mathrm{sct}}} \arrow[r] \arrow[d, "\P"'] & {[\![\pi]\!]_{\mathrm{ta}}} \arrow[d] \\ {[\![[\,\pi \mid \iota{:}\,\sigma\,]]\!]_{\mathrm{ta}}} \arrow[r] & {[\![\pi\iota']\!]_{\mathrm{sct}}} \end{tikzcd}$$

commute, the latter for all $\iota' \in \operatorname{dom}\pi - \{\iota\}$. (The unlabeled arrows in these diagrams denote the evident projection morphisms.)

Notice that abstractions do not contain type information, so that our inference rules describe an implicitly typed language. Of course, in practice one must use an explicitly typed language, since there is no type-inference decision procedure for intersection types. But the explicit type information is not relevant to the semantics of the language.

The next two cases deal with operations for constructing objects and extracting their fields. Intuitively, $\iota \equiv p$ constructs an object with a single field whose name is ι and whose value is p, while $p.\iota$ extracts the value of the field named ι from the object p. In the actual semantics, however, $[\![\iota{:}\,\omega]\!]_{\mathrm{ct}} = [\![\omega]\!]$, i.e. single-field objects are identical to their field values, so that the semantic equations are trivial.

- Object Construction

$$\frac{\pi \vdash p_1 : \omega}{\pi \vdash (\iota \equiv p_1) : (\iota{:}\,\omega)} \qquad \begin{aligned} [\![\pi \vdash (\iota \equiv p_1) : (\iota{:}\,\omega)]\!] &= [\![\pi \vdash p_1 : \omega]\!] \\ [\![\pi \vdash (\iota \equiv p_1) : (\iota{:}\,\omega)]\!]\eta &= [\![\pi \vdash p_1 : \omega]\!]\eta. \end{aligned}$$

- Field Selection

$$\frac{\pi \vdash p_1 : (\iota{:}\,\omega)}{\pi \vdash p_1.\iota : \omega} \qquad \begin{array}{c}[\![\pi \vdash p_1.\iota : \omega]\!] = [\![\pi \vdash p_1 : (\iota{:}\,\omega)]\!] \\ [\![\pi \vdash p_1.\iota : \omega]\!]\eta = [\![\pi \vdash p_1 : (\iota{:}\,\omega)]\!]\eta.\end{array}$$

The next four rules describe the "merging" operation used in the programming language Forsythe. The essential idea is that a phrase p_1, p_2 should take on the meanings of both p_1 and p_2. However, to avoid ambiguity, the phrase p_2 is required to be either a lambda expression or an object constructor, whose meaning "overwrites" the corresponding type of meaning of p_1. (In the semantic equations, P and C stand for the typing judgements that are the premiss and conclusion of the corresponding inference rule.)

- Merging

$$\frac{\pi \vdash \lambda\iota.\ p_2 : \sigma \to \omega}{\pi \vdash (p_1, \lambda\iota.\ p_2) : \sigma \to \omega} \qquad \begin{array}{c}[\![C]\!] = [\![P]\!] \\ [\![C]\!]\eta = [\![P]\!]\eta\end{array}$$

$$\frac{\pi \vdash p_1 : \omega}{\pi \vdash (p_1, \lambda\iota.\ p_2) : \omega} \quad \begin{array}{l}\text{when } \omega \text{ is primitive or} \\ \text{has the form } \iota'{:}\,\omega_1\end{array} \quad \begin{array}{c}[\![C]\!] = [\![P]\!] \\ [\![C]\!]\eta = [\![P]\!]\eta\end{array}$$

$$\frac{\pi \vdash \iota \equiv p_2 : (\iota{:}\,\omega)}{\pi \vdash (p_1, \iota \equiv p_2) : (\iota{:}\,\omega)} \qquad \begin{array}{c}[\![C]\!] = [\![P]\!] \\ [\![C]\!]\eta = [\![P]\!]\eta\end{array}$$

$$\frac{\pi \vdash p_1 : \omega}{\pi \vdash (p_1, \iota \equiv p_2) : \omega} \quad \begin{array}{l}\text{when } \omega \text{ is primitive or} \\ \text{has the form } \sigma \to \omega_1 \\ \text{or } \iota'{:}\,\omega_1 \text{ where } \iota' \neq \iota\end{array} \quad \begin{array}{c}[\![C]\!] = [\![P]\!] \\ [\![C]\!]\eta = [\![P]\!]\eta.\end{array}$$

Next we have two rules for introducing and eliminating finite sets of canonical types:

- Finite Sets of Canonical Types: Introduction

$$\frac{\pi \vdash p : \omega_1 \qquad \cdots \qquad \pi \vdash p : \omega_n}{\pi \vdash p : \{\omega_1, \ldots, \omega_n\}} \qquad [\![\pi \vdash p : \{\omega_1, \ldots, \omega_n\}]\!]_{\text{sct}} = \alpha.$$

To define $\alpha \in [\![\pi]\!]_{\text{ta}} \xrightarrow[\mathcal{K}]{} [\![\{\omega_1, \ldots, \omega_n\}]\!]_{\text{sct}}$, we first note that, whenever $\omega_i \sqcup \omega_{i'}$ exists, the rule for subtyping gives both

$$[\![\pi \vdash p : \omega_i]\!] \,;\, [\![\omega_i \leq \omega_i \sqcup \omega_{i'}]\!]_{\text{ct}} = [\![\pi \vdash p : \omega_i \sqcup \omega_{i'}]\!]$$

$$[\![\pi \vdash p : \omega_{i'}]\!] \,;\, [\![\omega_{i'} \leq \omega_{i'} \sqcup \omega_{i'}]\!]_{\text{ct}} = [\![\pi \vdash p : \omega_i \sqcup \omega_{i'}]\!],$$

so that, assuming coherence, the left sides of these equations must be equal. Thus, there is a cone on the diagram $\mathcal{J}_\sigma$ (where $\sigma = \{\omega_1, \ldots, \omega_n\}$) whose vertex is $[\![\pi]\!]_{\mathrm{ta}}$ and whose morphism parts are $[\![\pi \vdash p : \omega_i]\!]$. Then α is the unique mediating morphism from this cone to the limit of the same diagram.

For a domain-theoretic semantics, the semantic equation is

$$[\![\pi \vdash p : \{\omega_1, \ldots, \omega_n\}]\!]\eta = \langle [\![\pi \vdash p : \omega_1]\!]\eta, \ldots, [\![\pi \vdash p : \omega_n]\!]\eta \rangle.$$

- Finite Sets of Canonical Types: Elimination

$$\frac{\pi \vdash p : \{\omega_1, \ldots, \omega_n\}}{\pi \vdash p : \omega_i} \qquad \begin{array}{l} [\![\pi \vdash p : \omega_i]\!] = [\![\pi \vdash p : \{\omega_1, \ldots, \omega_n\}]\!] \,;\, \pi_i \\ [\![\pi \vdash p : \omega_i]\!]\eta = ([\![\pi \vdash p : \{\omega_1, \ldots, \omega_n\}]\!]\eta)_i, \end{array}$$

where π_i is the ith morphism part of the limit of $\mathcal{J}_\sigma$.

Finally, we give rules for fixed-point and conditional constructions. For these rules, we consider only the special case where $\mathcal{K}$ is a category of (some kind of) domains and continuous functions.

- Fixed Points

$$\frac{[\,\pi \mid \iota{:}\,\sigma\,] \vdash p_1 : \sigma}{\pi \vdash \mu\iota.\ p_1 : \sigma}$$

$$[\![\pi \vdash \mu\iota.\ p_1 : \sigma]\!] = \mathsf{ab}(\P \,;\, [\![\,[\,\pi \mid \iota{:}\,\sigma\,] \vdash p_1 : \sigma]\!]) \,;\, \mathsf{Y}_{[\![\sigma]\!]_{\mathrm{sct}}}$$

$$[\![\pi \vdash \mu\iota.\ p_1 : \sigma]\!]\eta = \mathsf{Y}_{[\![\sigma]\!]_{\mathrm{sct}}}(\lambda x \in [\![\sigma]\!]_{\mathrm{sct}}.\ [\![\,[\,\pi \mid \iota{:}\,\sigma\,] \vdash p_1 : \sigma]\!][\,\eta \mid \iota{:}\,x\,],$$

where Y_P denotes the least fixed-point operator for the domain P.

- Conditionals

$$\frac{\pi \vdash p_1 : \mathrm{bool} \qquad \pi \vdash p_2 : \omega \qquad \pi \vdash p_3 : \omega}{\pi \vdash \mathbf{if}\ p_1\ \mathbf{then}\ p_2\ \mathbf{else}\ p_3 : \omega}$$

$$\begin{aligned} &[\![\pi \vdash \mathbf{if}\ p_1\ \mathbf{then}\ p_2\ \mathbf{else}\ p_3 : \omega]\!] \\ &\qquad = ([\![\pi \vdash p_1 : \mathrm{bool}]\!] \otimes [\![\pi \vdash p_2 : \omega]\!] \otimes [\![\pi \vdash p_3 : \omega]\!]) \,;\, \mathsf{cd}_{[\![\omega]\!]_{\mathrm{ct}}} \end{aligned}$$

$$\begin{aligned} &[\![\pi \vdash \mathbf{if}\ p_1\ \mathbf{then}\ p_2\ \mathbf{else}\ p_3 : \omega]\!]\eta \\ &\qquad = \begin{cases} \bot & \text{when } [\![\pi \vdash p_1 : \mathrm{bool}]\!]\eta = \bot \\ [\![\pi \vdash p_2 : \omega]\!]\eta & \text{when } [\![\pi \vdash p_1 : \mathrm{bool}]\!]\eta = \mathbf{true} \\ [\![\pi \vdash p_3 : \omega]\!]\eta & \text{when } [\![\pi \vdash p_1 : \mathrm{bool}]\!]\eta = \mathbf{false}. \end{cases} \end{aligned}$$

Here cd_P denotes the conditional operator for the domain P:

$$\mathsf{cd}_P\langle b, x_1, x_2\rangle = \begin{cases} \perp & \text{when } b = \perp \\ x_1 & \text{when } b = \mathbf{true} \\ x_2 & \text{when } b = \mathbf{false}. \end{cases}$$