Semantics of a function symbol of Haskell is a certain function between domains. But we don’t need all of these functions for the semantics.

For the semantics, we only use "total" functions, whose argument or result can also be \( \bot \). This allows us to express non-strict functions that return a result \( \neq \bot \) even if their argument is \( \bot \). 

**Def 2.4.6 (Extension and Strictness of Functions)**

Let \( f : A \rightarrow B \) be a function. Every function \( f' : A_\bot \rightarrow B \)

is called an extension of \( f \) iff \( A_\bot = A \cup \{ \bot \} \), and

\[
f(d) = f'(d) \quad \text{for all } d \in A.
\]

A function \( g : D_1 \times \ldots \times D_n \rightarrow D \) for domains \( D_1, \ldots, D_n \) is strict iff \( g(d_1, \ldots, d_n) = \bot \) whenever \( d_i = \bot \) for some \( i \).

**Ex:**

\[ \text{one :: Int } \rightarrow \text{ Int } \]

\[ \text{one } x = \text{ if } x = 1 \text{ then } \text{one } x \text{ else } \bot \]

This corresponds to a function \( f \) that maps numbers from \( \mathbb{Z} \) to \( \mathbb{N} \). But the semantics of "one" should be a function \( f' \) that is an extension of \( f \).

There are several possible extensions:

\[ f'_n(x) = \begin{cases} 1 & \text{for all } x \in \mathbb{Z}, \\ 1 & \text{if } x = \bot \end{cases} \]

This should be the semantics of "one" in Haskell.
(a) $f_1^\prime(x) = 1$ for all $x \in \mathbb{Z}$, $f_1^\prime(1) = 1$

(b) $f_2^\prime(x) = n \quad \rightarrow \quad f_2^\prime(1) = 1$

(c) $f_3^\prime(x) = n \quad \rightarrow \quad f_3^\prime(1) = 0$

Depending on whether its argument is defined or not, this fact returns 1 or 0.

This function solves the halting problem! So $f_3^\prime$ is not computable and it should not be contained in our function domain.

Functions $g$ where $g(1) \neq 1$ and $g(d) \neq g(1)$ for some $d \neq 1$

are not computable. => Such functions should be excluded from the function domain.

=> We restrict ourselves to **monotonic** functions.

**Def. 2.1.7 (Monotonic Functions)**
Let $E_{D_1}$, $E_{D_2}$ be partial orders on $D_1$ and $D_2$, resp.
A function $f : D_1 \rightarrow D_2$ is **monotonic** iff $d \in D_{D_1}$ implies $f(d) \in D_{D_2}$.

So: if the argument gets more defined, then the result of $f$ also gets more defined (or remains
equal).

In our example, $f_1'$ and $f_2'$ are monotonic.
But $f_3'$ is not monotonic.
All computable functions are monotonic!

More examples:

$$id : \mathbb{I} \rightarrow \mathbb{I}$$

$$id \ x = x$$

The semantics of $id$ should be the function $f : \mathbb{I} \rightarrow \mathbb{I}$ with $f(x) = x$ for $x \in \mathbb{I}$ and $f(1) = 1$. (Otherwise, $f$ would not be monotonic).
Here, $f$ is strict.
In fact, strictness implies monotonicity.

Lemma 2.1.8 (Strictness implies Monotonicity)

Let $D_1, \ldots, D_n, D$ be domains, where $D_1, \ldots, D_n$ are flat.
If $f : D_1 \times \ldots \times D_n \rightarrow D$ is strict, then $f$ is also monotonic.

Proof: Let $(d_1, \ldots, d_n) \preceq (d_1', \ldots, d_n')$.
To show: $f(d_1, \ldots, d_n) \preceq f(d_1', \ldots, d_n')$.
If $(d_1, \ldots, d_n) = (d_1', \ldots, d_n')$, then this is trivial by reflexivity of $\preceq$.
Otherwise, there must be some $1 \leq i \leq n$ where $d_i = 1$ (Since $D_1, \ldots, D_n$ are flat).

Thus: $f(d_1, \ldots, d_i, \ldots, d_n) = 1 \preceq 1 = f(d_1', \ldots, d_n')$. \[\square\]
But since Haskell uses lazy evaluation, we also need non-strict functions for the semantics.

\[
\text{cond} :: (\text{Bool}, \text{Int}, \text{Int}) \to \text{Int}
\]

\[
\text{cond } (\text{True}, x, y) = x
\]

\[
\text{cond } (\text{False}, x, y) = y
\]

The semantics of \text{cond} should be a function \( f : \mathbb{B}_1 \times \mathbb{Z}_\perp \times \mathbb{Z}_\perp \to \mathbb{Z}_\perp \)

where

\[
f(b, x, y) = \begin{cases} 
  x, & \text{if } b = \text{True} \\
  y, & \text{if } b = \text{False} \\
  \perp, & \text{if } b = \perp
\end{cases}
\]

This function is monotonic, but not strict:

\[
f(\text{True}, 2, \perp) = 2.
\]

How does the step from a Haskell declaration of a function to the semantics of that function work in general? To this end: regard chains of elements that become more and more defined.

\textbf{Def 2.1.9 (Chain)}

Let \( E \) be a partial ordering on a set \( D \). A non-empty subset \( \{d_1, d_2, \ldots\} \) of \( D \) is called a \underline{chain}.
iff \[d_1 \sqsubseteq d_2 \sqsubseteq d_3 \sqsubseteq \ldots\].

Chain on \(\mathbb{Z}_1\): \(\{1, 5\}\), because \(1 \sqsubseteq \mathbb{Z}_1 5\).

Chain on \(\mathbb{Z}_1 \times \mathbb{Z}_1\): \(\{1, (5, 1), (5, 8)\}\), because

\[
(1, 1) \sqsubseteq (5, 1) \sqsubseteq (5, 8)
\]

\[\mathbb{Z}_1 \times \mathbb{Z}_1\]

Chain on \(\mathbb{Z}_1 \rightarrow \mathbb{Z}_1\): \(\{\text{fact}_0, \text{fact}_1, \text{fact}_2, \ldots\}\) (Slide 31)

\(\text{fact}_1(x)\) computes the factorial function for \(x < 1\)

\[
\text{fact}_2(x) \quad : \quad x < 2
\]

\[
\text{fact}_0 \sqsubseteq \text{fact}_1 \sqsubseteq \text{fact}_2 \sqsubseteq \text{fact}_3 \sqsubseteq \ldots
\]

Chains can be used to define the semantics of functions. The "limes" of the chain \(\text{fact}_0 \sqsubseteq \text{fact}_1 \sqsubseteq \ldots\) is the factorial function.

More precisely, we look for least upper bound of such chains.

**Def 2.1.10 (Least Upper Bound)**

Let \(\sqsubseteq\) be a partial ordering on a set \(D\), let \(S \subseteq D\). Then \(d \in D\) is an upper bound of \(S\) iff \(d \sqsubseteq d'\) for all \(d' \in S\).
The element $d$ is the least upper bound of $S$ (denoted as "supremum") iff moreover $d \in S$ holds for all other upper bounds $e$ of $S$. We denote the lub of $S$ by $\bigcup S$.

Lub of $\{1, 5\}$ is 5.

Lub of $\{1, (5, 1), (5, 8)\}$ is $(5, 8)$.

Here, $(5, 8)$ is the only upper bound of the chain.

Lub of $\{\text{fact}, \text{fact}, \ldots\}$ is fact.

This chain has infinitely many upper bounds: they are like fact for arguments from $\mathbb{Z}$, but can return arbitrary results for argument $\bot$. The smallest of these upper bounds is fact, because it returns the result $\bot$ if the argument is $\bot$.

For base domains, it is trivial to compute lub’s of chains. The following lemma shows how to compute lub’s for product and function domains.

Lemma 2.1.11 (Lub’s for Product and Function Domains)

Let $D_1, \ldots, D_n, D, D'$ be domains.

(a) Let $S \subseteq D_1 \times \ldots \times D_n$. For all $1 \leq i \leq n$, let $S_i = \{d_i \mid \text{there exists } (d_1, \ldots, d_i, \ldots, d_n) \in S \}$

- If $S = \{(1, 1), (5, 1)\}$, then $S = \{(5, 1)\}$
- If $S = \{(1, 1)\}$, then $S = \{(1, 1)\}$
- If $S = \{(5, 1)\}$, then $S = \{(5, 1)\}$

• lub exists only if $S_i$ exists for all $1 \leq i \leq n$
\[ S = \{ 1, 5 \} \]
\[ S_2 = \{ 1, 8 \} \]

(b) Let \( S \) be a set of functions from \( D \rightarrow D \). For all \( i \in D \), let \( S_i = \{ f(i) \mid f \in S \} \). 

- \( LUS \) exists iff \( LUS_i \) exists for all \( 1 \leq i \leq n \) 
- If \( LUS \) exists, then \( LUS = (LUS_1, \ldots, LUS_n) \).

Proof of (a): (Proof of (b) is analogous.)

\( \Leftarrow \): To show: If all \( LUS_i \) exist, then \( (LUS_1, \ldots, LUS_n) \) is the lub of \( S \).

We first show that \( (LUS_1, \ldots, LUS_n) \) is an upper bound of \( S \).

Let \( (d_1, \ldots, d_n) \in S \). Then \( d_1 \in S_1, \ldots, d_n \in S_n \). Hence: \( d_1 \in LUS_1, \ldots, d_n \in LUS_n \).

This implies \( (d_1, \ldots, d_n) \in (LUS_1, \ldots, LUS_n) \).

Now we show that \( (LUS_1, \ldots, LUS_n) \subseteq (e_1, \ldots, e_n) \) holds for any upper bound \( (e_1, \ldots, e_n) \) of \( S \).

If \( (e_1, \ldots, e_n) \) is an upper bound of \( S \), then \( e_i \) is an upper bound of \( S_i \) for any \( 1 \leq i \leq n \).

This implies \( (LUS_1, \ldots, LUS_n) \subseteq (e_1, \ldots, e_n) \).

\( \Rightarrow \): Now we assume that \( LUS = (m_1, \ldots, m_n) \) exists.

We have to show that \( m_1 \) is lub of \( S_1 \), \( m_n \) is lub of \( S_n \).

We first show that \( m_i \) is an upper bound of \( S_i \) (for any \( 1 \leq i \leq n \)).

Let \( d_i \in S_i \). There exists \( (d_1, \ldots, d_i, \ldots, d_n) \in S \).

As \( (m_1, \ldots, m_i, \ldots, m_n) \) is an upper bound of \( S \), we have \( (d_1, \ldots, d_i, \ldots, d_n) \subseteq (m_1, \ldots, m_i, \ldots, m_n) \). Hence: \( d_i \geq m_i \).

Now we show that \( m_i \leq e_i \) for any upper bound \( e_i \) of \( S_i \).
If $e_i$ is an upper bound of $S_i$, then $(m_1, \ldots, e_i, \ldots, m_n)$ is an upper bound of $S$.

Since $(m_1, \ldots, m_i, \ldots, m_n)$ is the least upper bound of $S$, we have $(m_1, \ldots, m_i, \ldots, m_n) \preceq (m_1, \ldots, e_i, \ldots, m_n)$, which implies $m_i \preceq e_i$.

The essential property that we need for domains $D$ is that $\bumpeq_D$ must be complete.

**Def 2.1.12 (Complete Partial Order, cpo)**

A reflexive partial order $\preceq$ on a set $D$ is **complete** if

1. $D$ has a smallest element (denoted $\bot_D$)
2. Every chain $S \preceq D$ has a lub $\bigcup S \in D$.

The domains $D$ that we regarded up to now are indeed complete (i.e., $\bumpeq_D$ is a cpo).

**Thm 2.1.13 (Completeness of Domains)**

Let $D_1, \ldots, D_n$ be domains.

1. Every reflexive partial order that has a smallest element and where all chains are finite is a cpo. Thus, the flat base domains $\mathbb{N}_1$, $\mathbb{B}_1$, $\mathbb{C}_1$, $\mathbb{F}_1$ are cpo's.
2. If $\bumpeq_{D_1}, \ldots, \bumpeq_{D_n}$ are cpo's, then $\bumpeq_{D_1 \times \ldots \times D_n}$ is also a cpo on $D_1 \times \ldots \times D_n$. 
(c) If $E_{D_2}$ is a cpo, then $E_{D_n \to D_2}$ is also a cpo on $D_n \to D_2$.

Proof of (a) and (b): (c) is analogous to (b).

(a) If $S = \{ d_i^n \}_{i=1}^d$ is a finite chain with $d_1 \leq d_2 \leq \ldots \leq d_n$, then the lub of $S$ is $d_n$.

$\forall d_i \in d_n \text{ holds by transitivity } \implies d_n$ is an upper bound.

For any other upper bound $e$ of $S$, we also have $d_n \leq e$.

(b) The smallest element of $D_n \times \ldots \times D_n$ is $(\bot_{D_1}, \ldots, \bot_{D_n})$.

Let $S \subseteq D_n \times \ldots \times D_n$ be a chain, i.e., $S = \{ (d_i^1, \ldots, d_i^n), (d_i^2, \ldots, d_i^n), \ldots \}$ where $(d_i^1, \ldots, d_i^n) \leq (d_i^2, \ldots, d_i^n) \leq \ldots$.

Hence: $d_i^1 \leq d_i^2 \leq \ldots$, i.e., $S_i = \{ d_i \mid \text{there exists } (d_i^1, \ldots, d_i^n) \in S \}$ is also a chain for all $1 \leq i \leq n$.

Since $E_{D_i}$ is a cpo for all $1 \leq i \leq n$, the lub of $S_i$ exists.

By Lemma 2.1.11 (a): \text{LUS} = (\text{LUS}_1, \ldots, \text{LUS}_n) exists as well.