The Store

- primitive domains Location , Storable
- domains contain $\perp$
- locations can be unallocated (*unused*)
- locations can be allocated but *undefined*
- *store*: a mapping from locations (refs) to values

$$\text{Store} = \text{Location} \rightarrow (\mathit{stored}\ \text{Storable} + \mathit{undefined} + \mathit{unused})$$

injection maps for tagged union:

$\mathit{stored} : \text{Storable} \rightarrow (\mathit{stored}\ \text{Storable} + \mathit{undefined} + \mathit{unused})$
$\mathit{undefined} : (\mathit{stored}\ \text{Storable} + \mathit{undefined} + \mathit{unused})$
$\mathit{unused} : (\mathit{stored}\ \text{Storable} + \mathit{undefined} + \mathit{unused})$

Update Operator

- $[a \mapsto b]$ is an operator that takes a function f to another function $f[a \mapsto b]$. It is written postfix.

- *Definition:* Let $f : X \rightarrow Y$ and let a, b be any values. The function $f[a \mapsto b] :$ $X \cup \{a\} \rightarrow Y \cup \{b\}$ is defined by:

$$(f[a \mapsto b])(x) = \begin{cases} b & \mathit{if}\ x = a \\ f(x) & \mathit{if}\ x \neq a \end{cases}$$

- We can extend this notation to multiple successive changes as follows:

$$f[a_1 \mapsto b_1, a_2 \mapsto b_2] = (f[a_1 \mapsto b_1])[a_2 \mapsto b_2]$$

- **Example:** Semantics of assignment. Suppose the identifier `x` is bound to the location l. Then executing the assignment `x:=e` has the effect of changing memory:

$$\mathit{execute}[\![\mathtt{x:=e}]\!]\ \mathit{sto} = \mathit{sto}[l \mapsto \mathit{eval}[\![\mathtt{e}]\!]\ \mathit{sto}]$$

Auxiliary Functions

$$
\begin{array}{lll}
empty-store & : & \mathsf{Store} \\
allocate & : & \mathsf{Store} \rightarrow \mathsf{Store} \times \mathsf{Location} \\
deallocate & : & \mathsf{Store} \times \mathsf{Location} \rightarrow \mathsf{Store} \\
update & : & \mathsf{Store} \times \mathsf{Location} \times \mathsf{Storable} \rightarrow \mathsf{Store} \\
fetch & : & \mathsf{Store} \times \mathsf{Location} \rightarrow \mathsf{Storable}
\end{array}
$$

$$empty-store = \lambda\, loc\,.\ unused$$

$$
\begin{array}{l}
allocate\ sto = \\
\qquad \textbf{let}\quad loc = any-unused-location(sto)\ \ \textbf{in} \\
\qquad (\ sto[loc \mapsto undefined],\ loc\ )
\end{array}
$$

$$
\begin{array}{l}
deallocate(sto, loc) = \\
\qquad sto[loc \mapsto unused]
\end{array}
$$

$$
\begin{array}{l}
update(sto, loc, stble) = \\
\qquad sto[loc \mapsto stored\ stble]
\end{array}
$$

$$
\begin{array}{l}
fetch(sto, loc) = \\
\qquad \textbf{let}\quad stored-value(stored\ stble) = stble \\
\qquad \qquad\ \ stored-value(undefined) = \bot \\
\qquad \qquad\ \ stored-value(unused) = \bot \\
\qquad \textbf{in} \\
\qquad stored-value(sto(loc))
\end{array}
$$

Example

A simple language with expressions and assignment

Syntax

```
Command    ::= Identifier  := Expression
             | Command ; Command

Expression ::= Expression  + Expression
             | Numeral
             | Identifier
```

Semantics

semantic function binds names to locations:

location : `Identifier` $\rightarrow$ Location

semantic map:

execute : `Command` $\rightarrow$ Store $\rightarrow$ Store

$$\begin{aligned} &\textit{execute}[\![I := E]\!]\ \textit{sto} = \\ &\qquad \textbf{let}\ \textit{int} = \textit{evaluate}[\![E]\!]\ \textit{sto} \\ &\qquad \textbf{in} \\ &\qquad \textit{update}(\textit{sto}, \textit{location}\ I, \textit{int}) \end{aligned}$$

$$\begin{aligned} &\textit{execute}[\![C_1\ ;\ C_2]\!]\ \textit{sto} = \\ &\qquad \textbf{let}\ \textit{sto}' = \textit{execute}[\![C_1]\!]\ \textit{sto} \\ &\qquad \textbf{in} \\ &\qquad \textit{execute}[\![C_2]\!]\ \textit{sto}' \end{aligned}$$