# MSc Project Notes

Opale

April 6, 2024

## 1 Memory model constructor cheatsheet

In italic, "primitives", ie. types that aren't memory models.

Note $X^? \overset{\text{def}}{=} X \uplus \bot$, $X^\emptyset \overset{\text{def}}{=} X \uplus \emptyset$

| Language | Memory Model |
|---|---|
| WISL | PMap(*Loc*, OneShot(List(Exc(*Val*)))) |
| JSIL | PMap(*Loc*, PMap(*Str*,Exc(*Val*$^\emptyset$)) × PMap(*Loc*,Ag(*Val*)) |

| Name | Purpose | Type | Actions | Predicates |
|---|---|---|---|---|
| Exc | Exclusive ownership of a specific resources | $\tau^?$ | `load`, `store` *[1] | `PointsTo` |
| Ag | Multiple parties agree on the same value for a resource | $\tau$ | | `Agree` |
| Frac | Allow partial (readonly) ownership of an object | $\tau \times (0,1]$ | | `Frac` |
| List | Ensure continuous memory allocation | $(\mathbb{N} \overset{fin}{\rightharpoonup} \tau) \times \mathbb{N}^?$ *[2] | | lift with index in-param *[4] |
| OneShot | The program only has one go at something (eg. freeing memory) | Exc($\tau$) + Ag($\{\emptyset\}$) | `free` | |
| PMap | Define memory as a map of address (a sort I) to value | $(\mathrm{I} \overset{fin}{\rightharpoonup} \tau) \times \mathcal{P}(I)^?$ *[3] | | lift with index in-param |
| Product (×) | Two simultaneous states, each being updated separately (eg. List) | $\tau_1 \times \tau_2$ | lift with `A1`, `A2` | |
| Sum (+) | Either of two states existing (eg. OneShot) | $\tau_1 \uplus \tau_2$ | lift with `A1`, `A2` | |

*[1] Would we define `load` and `store` at this level, or at a more primitive "Value" memory model level?

*[2] Full definition: $\left\{(b, n^?) \in (\mathbb{N} \overset{fin}{\rightharpoonup} \tau) \times \mathbb{N}^? \mid \mathrm{dom}(b) \subseteq [0, n^?)\right\}$

*[3] Full definition: $\left\{(h, d) \mid h \in (\mathrm{I} \overset{fin}{\rightharpoonup} \tau) \wedge d \in \mathcal{P}(I)^? \wedge \mathrm{dom}(h)^? \subseteq d\right\}$