1 - Introduction & Motivation

- Burak Ekici, PhD Student at University Joseph Fourier (Grenoble I). Supervised by: Jean-Guillaume Dumas, Dominique Duval & Damien Pous.

- **Modeling the computational effects**, such as State, Exceptions, IO, Non-Determinism, of non-functional programming languages by using decorated logic [Dominguez&Duval’10]

- Developing the model framework in the Coq proof management system to verify its internal validity & consistency.
2 - States Effect Specification

- Sets (Objects):
  - \( N \) (values), \( U \) (unit : \{\*\})

- Functions (Morphisms):
  - \( id_N : N \rightarrow N \), \( id_U : U \rightarrow U \), \( \langle \rangle_N : N \rightarrow U \)
  - \( lookup_i : U \rightarrow N \), \( update_i : N \rightarrow U \)

where \( i \in \{X, Y, \ldots\} \) is a memory location identifier. [Dumas et al.’12]
Functions are Classified:

- **pure functions**: E.g. $id_N^{(0)} : N \rightarrow N$, $id_U^{(0)} : U \rightarrow U$, $\langle \rangle_N^{(0)} : N \rightarrow U$

- **accessors**: E.g. $lookup_i^{(1)} : U \rightarrow N$

- **modifiers**: E.g. $update_i^{(2)} : N \rightarrow U$

Equations are Classified:

- **strong equality ($f \equiv g$)**: if $f$ and $g$ calculate the same result with the same effects on the state structure.

- **weak equality ($f \sim g$)**: if $f$ and $g$ calculate the same result and might have different effects on the state structure.

  E.g. $(lookup_i \circ update_i) \sim id_i$
Properties of the State Structure by Plotkin et al.

1. Annihilation lookup-update. \( \forall i \in \text{Loc}, u_i \circ l_i \equiv \text{id}_U : U \rightarrow U \)

2. Interaction lookup-lookup. \( \forall i \in \text{Loc}, l_i \circ \langle \rangle_i \circ l_i \equiv l_i : U \rightarrow N \)

3. Interaction update-update. \( \forall i \in \text{Loc}, u_i \circ \pi_2 \circ (u_i \times \text{id}_i) \equiv u_i \circ \pi_2 : N \times N \rightarrow U \)

4. Interaction update-lookup. \( \forall i \in \text{Loc}, l_i \circ u_i \sim \text{id}_i : N \rightarrow N \)

5. Commutation lookup-lookup. \( \forall i \neq j \in \text{Loc}, (\text{id}_i \times l_j) \circ l_i \equiv \text{perm}_{j,i} \circ (\text{id}_j \times l_i) \circ l_j : U \rightarrow N \times N \)

6. Commutation update-update. \( \forall i \neq j \in \text{Loc}, u_j \circ \pi_2 \circ (u_i \times \text{id}_j) \equiv u_i \circ \pi_1 \circ (\text{id}_i \times u_j) : N \times N \rightarrow U \)

7. Commutation update-lookup. \( \forall i \neq j \in \text{Loc}, l_j \circ u_i \circ \pi_1 \equiv \pi_2 \circ (u_i \times \text{id}_j) \circ (\text{id}_i \times l_j) : N \times U \rightarrow U \)
Proofs and Proof Verification Structure in Coq

To prove propositions by Plotkin et al.:

- 75 rules have been used.
  - 23 rules ⇒ Monadic Equational Logic.
  - 9 rules ⇒ Categorical Products.
  - 43 rules ⇒ Derived.

- ≈ 1100 lines of Coq Code have been written for proof verifications.

What is next?

- To model other effects (with combinations) and use Coq for related proof verifications.
References


Many thanks for your patience!
Questions?