Logical Relations for Mutable State

Hakan Dingenc

1 Logical Relations for Mutable State

```
@phdthesis{Ahmed:2004:STM:1037736,
  author = {Ahmed, Amal Jamil},
  title = {Semantics of Types for Mutable State},
  year = {2004},
  note = {AAI3136691},
  publisher = {Princeton University},
  address = {Princeton, NJ, USA},
}
```

Summary: In the first half of her dissertation, Ahmed assigns a *meaning* to each type in three different languages based on the lambda calculus: a pure language, a language with immutable state, and a language with mutable state. In each setting, the meaning of a type is formulated as a certain set using the operational semantics of the language. By carefully crafting the meaning of each type, Ahmed is able to prove the typing rules of the language, which enables a proof of type safety, that is, evaluation of well-typed terms do not get stuck. While the semantics of immutable references uses a possible-world model that describes what kind of stores are reachable in the future from a given store, the semantics of mutable references uses a *step-indexed* possible-worlds model that is based on how many operational steps are taken when considering a certain term. Ahmed includes for each language a description of how her model corresponds to the world of logical relations.

Evaluation: Giving meaning to types sets Ahmed's work apart from the usual syntactic approaches such as progress and preservation that consider types as mere attributes of terms. The trade-off here is avoiding having to type the intermediate terms in the operational semantics that don't appear in the original language such as stores at the cost of having to carefully

come up with meaning of types that leads to the sought proofs. Logical relations have been used to prove a wide range of important properties of programming languages and Ahmed's work on step-indexing provides the foundations for dealing with semantically challenging features such as mutable references and impredicative polymorphism.