This paper gives semantics for language interoperability [1].[1] The presentation is divided into three sections: initially foreign values are considered opaque, next they are given meaning in a type system, and finally they are monitored by higher-order contracts. The benefits and drawbacks of each approach are clearly stated.

Strengths

- Identifies the problem of defining *type-preserving* semantics between higher-order languages.
- Offers three reasonable solutions and makes connections to pragmatic industry solutions. Even the "naïve" lump embedding had practical applications before this paper gave it a firm semantics.
- Bold use of color. This might be the paper's most enduring academic contribution.

Weaknesses

- The Scheme and ML variants are very similar. Although the authors consider the problem of different integer encodings, real languages vary in many ways. For example:
  - User-defined datatypes in one language may not have an obvious encoding in another language. Can the (non-lump) semantics be extended with user-defined encodings for these data structures, and include a check that the encodings are isomorphisms?
  - Coq and Agda assume that all functions are pure and depend heavily on this assumption. Languages like OCaml, however, are stateful. This makes linking OCaml and Coq code potentially dangerous. Can we extend the semantics with similar higher-level properties?
  - Potentially non-terminating FFI calls can also violate one language's invariants. For example, the LiquidHaskell language uses an SMT solver to solve type-level equations, and must take care to not send diverging variables to the constraint solver. It does so by statically analyzing Haskell code. Could this paper's semantics offer termination guarantees for FFI calls?

[1] The title is a complete description.

– `C` and `C++` are plainly unsafe; they can read and write to arbitrary memory locations. Is there hope for a semantics that rules out memory model violations? At the very least, can we detect such abuses? I'm imagining a C program that corrupts ML's contract-checking code.

References

[1] Jacob Matthews and Robert Bruce Findler. Operational semantics for multi-language programs. In *POPL*, 2007.