Hoare Type Theory is proposed as a uniform specification and implementation language [1]. Computations in HTT may interact with their underlying machine heap and the HTT type system can express constraints and invariants about this heap. The point being, one can do many things in HTT and all these actions can be accurately described with an HTT type.

This has gone over my head again. Even the title confuses me: the only algebraic datatypes I saw were the built-in natural numbers and axiomatized *list* and neither of these seemed mutable per se. I understand these datatypes live on the heap, and every heap location can be strongly updated, but I don't think this would let me mutate the tail of a list or replace the argument to the s constructor with a larger natural. The syntax says that these constructors take terms, not heap locations, as their arguments. (Does this mean that my entire list resides in a single heap location?)

Strengths

- Brings the power of separation logic into a type system i.e. into a programming language.
- Hoare types allow non-termination & state updates.
- Specifies a memory allocator.

Weaknesses

- The paper's contributions and improvements over prior work are not clear. From the introduction, I expected that we'd add ECC to HTT, primarily to define invariants on local state. But reading felt like a general introduction to HTT, or maybe an extended abstract for the technical report. Neither ECC or HTT were clearly defined, and informal descriptions of HTT are spread throughout the paper. One frustrating point was the in-example introduction of "two essential features" on page 8.
- Without syntax for defining inductive (and coinductive) types, I have a hard time getting excited about higher-order logic predicates. (Even the paper's main example had to "assume as primitive" list operations.)
- I worry about one feature: that the unspecified part of memory is assumed invariant. This means that every function's signature needs to somehow include the signatures of the helper functions it calls.
  - Won't the types for new functions be linear in the size of the whole program so far?

– Unless library authors are careful to use existential types, they will force users to clutter new programs with implementation details of old functions.

References

[1] Aleksandar Nanevski, Amal Ahmed, Greg Morrisett, and Lars Birkedal. Abstract predicates and mutable adts in hoare type theory. In *ESOP*, 2007.