On Idris Elaboration

Peter Vanderbilt
pvanderbilt@acm.org

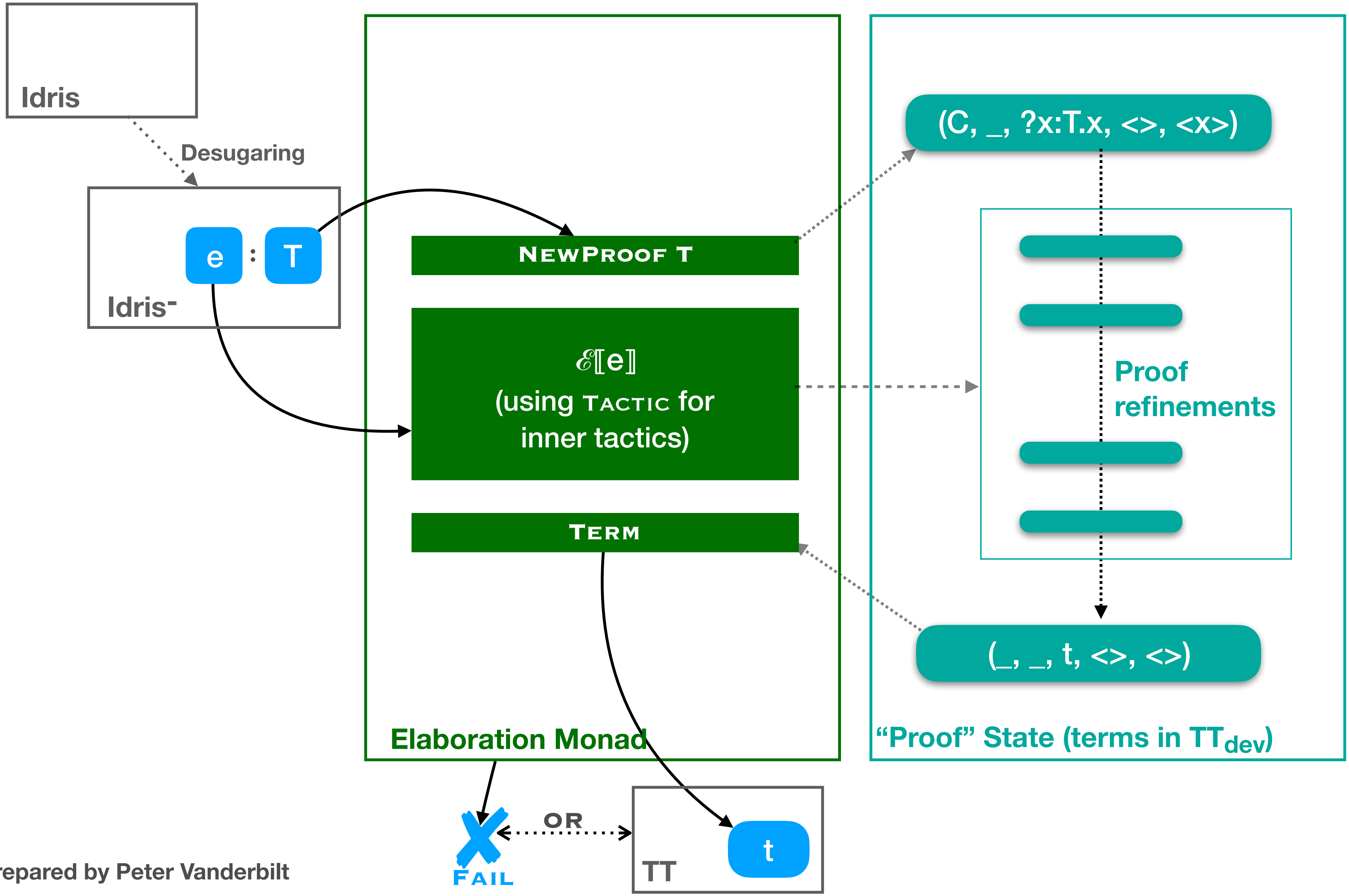
Version 1.1, 18-11-19

- From “Idris, a General Purpose Dependently Typed Programming Language: Design and Implementation”
  - By Edwin Brady

My understanding of Idris elaboration

- Given an expression and its expected type
  - Desugar to Idris$^-$, giving, say, e:T
  - Create a proof state where the goal is a hole of type T
  - Refine the goal to have the same structure as e
  - Extract the goal from the final proof state, giving t $\in$ TT
- Fail if anything goes wrong

Idris Elaboration



Prepared by Peter Vanderbilt

On “proof”

- Brady uses Curry-Howard to position the problem:
  - The type, T, is the proposition to be proved
  - The proof, t, is an inhabitant of T (so t:T in TT)
- *However* we don’t want just any inhabitant — it must be one that parallels the given (Idris) expression, e
- But the “proof” terminology is convenient, so let’s use it
- Note: he is *not* proving $C \vdash e : T$ in the TT logic!

On proof state

- The proof state is a tuple, (C, Δ, e, P, Q) where
  - C is the global context (definitions and types)
  - e is the proof term, a proof of the initial proposition
    - It may contain typed *holes* or *guesses* so $\in TT_{dev}$
  - P is a set of blocked unification problems, $(\Gamma, e1, e2)$
  - Q is a priority queue of hole names

On the proof process

- The initial state is (C, _, ?x:T.x, <>, <x>), so
    - The proof term is a single hole of type T
- As the proof progresses, driven by $\mathcal{E}[\![e]\!]$, holes in the proof term are filled in with terms that might have further (typed) holes
    - The invariant is that the proof term is of type T
    - Holes and guesses are “solved” by tactics SUBST (typically from UNIFY) and SOLVE
- The final state is (C, _, t, <>, <>)

On $\mathscr{E}$

- $\mathscr{E}[\![e]\!]$ generates a tactic that should refine the starting proof state to one that has t such that
    - $t \in TT$ (so t is *pure*, which means it has no holes)
    - $C \vdash t : T$ (in the TT logic)
    - t has a structure parallel to e

On Tactic

- Within tactic $\mathscr{E}[\![e]\!]$, calls of certain tactics are via Tactic
    - These "inner" tactics are Attack, Claim, Fill, Solve, Lambda, Pi, Let, Pat
- Tactic finds the sub term, *h*, in the proof term that is a hole or guess and whose name is at the front of *Q*
    - Γ is set to the binders leading to *h*
    - The inner tactic is called with arguments Γ and *h*
        - This may have side-effects (on the proof state)
    - The value returned by the inner tactic replaces *h* in the goal