```
theory Cell
  imports "Deriving.Derive" Base
begin

datatype 'a::"countable" cell = Array "'a array" | Upd nat "'a" "'a cell ref"

derive countable cell

instance cell :: (heap) heap ..

datatype 'a::"countable" cell' = Array' "'a list" | Upd' nat "'a" "'a cell ref"

fun cell_assn where
  "cell_assn (Array' xs) (Array a) = a ↦a xs"
| "cell_assn (Upd' i' val' p') (Upd i val p) = ↑(i = i' ∧ val = val' ∧ p = p')"
| "cell_assn _ _ = false"

lemma cell_assn_array [simp]: "cell_assn (Array' xs) c = (∃A a. ↑(c = Array a) * a ↦a xs)"
  by(cases c)(auto simp: ent_iff entails_def)

lemma cell_assn_upd [simp]: "cell_assn (Upd' i x p) c = ↑(c = Upd i x p)"
  by(cases c) auto

end
```