```
theory Hnr_Diff_Arr
  imports Hnr Diff_Arr_Safe
begin

named_theorems hnr_rule_diff_arr

definition master_assn' where
  "master_assn' S = (∃_At. master_assn t * ↑(∀ (xs, xsi) ∈ S. t ⊢ xs ∼ xsi))"

lemma hnr_pass_diff_arr [hnr_rule_diff_arr]:
  "hnr
      (master_assn' (insert (xs, xsi) S))
      (return xsi)
      (λxs' xsi'. master_assn' (insert (xs', xsi') S))
      (Some xs)"

definition New_Diff_Arr where
  "New_Diff_Arr a = a"

lemma hnr_from_array [hnr_rule_diff_arr]:
  "hnr
    (array_assn xs xsi)
    (Diff_Arr.from_array xsi)
    (λxs xsi. master_assn' { (xs, xsi) })
    (Some (New_Diff_Arr xs))"

lemma hnr_from_list [hnr_rule_diff_arr]:
  "hnr
    emp
    (Diff_Arr.from_list xs)
    (λxs xsi. master_assn' { (xs, xsi) })
    (Some (New_Diff_Arr xs))"

lemma hnr_lookup [hnr_rule_diff_arr]:
  "hnr
    (master_assn' (insert (xs, xsi) S) * id_assn i ii)
    (Diff_Arr_Safe.lookup xsi ii)
    (λr ri. id_assn r ri * master_assn' S)
    (Some (xs ! i))"

lemma hnr_realize:
  "hnr
    (master_assn' (insert (xs, xsi) S))
    (Diff_Arr.realize xsi)
    (λ r ri. master_assn' S * array_assn r ri)
    (Some xs)"

lemma hnr_update [hnr_rule_diff_arr]:
  "hnr
    (master_assn' (insert (xs, xsi) S) * id_assn i ii * id_assn v vi)
    (Diff_Arr_Safe.update xsi ii vi)
    (λxs' xsi'. master_assn' (insert (xs', xsi') S))
    (Some (xs [i:= v]))"

lemma hnr_length [hnr_rule_diff_arr]:
  "hnr
    (master_assn' (insert (xs, xsi) S))
    (Diff_Arr.length xsi)
    (λr ri. master_assn' S * id_assn r ri)
    (Some (length xs))"

end
```