```
theory Hnr_Diff_Arr
  imports Hnr Diff_Arr_Safe
begin

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

lemma master_assn'_cong:
  assumes
    "S = S'"
  shows
    "master_assn' S ⟹_A master_assn' S'"

definition Si_Tag where
  "Si_Tag x = x"

lemma si_initialize: "A ∪ {} = Si_Tag B ⟹ A = B"

lemma si_move_tag: "Si_Tag (insert x B) = insert x (Si_Tag B)"

lemma si_rotate: "A ∪ insert x B = C ⟹ insert x A ∪ B = C"

lemma si_match: "insert x A ∪ B = C ⟹ insert x A ∪ B = insert x C"

lemma si_rotate_back: "insert x A ∪ B = C ⟹ A ∪ insert x B = C"

lemma si_finish: "A ∪ {} = Si_Tag A"

method si_try_match = then_else
  ‹rule si_match›
  ‹((rule si_rotate_back)+)?›
  ‹rule si_rotate, si_try_match›

method si_initialize = rule si_initialize, (simp(no_asm) only: si_move_tag)?

method set_inference_keep = si_initialize, ((rule si_finish | si_try_match)+)?

method set_inference = set_inference_keep; fail

method hnr_diff_arr_match_atom = then_else
  ‹rule master_assn'_cong›
  ‹set_inference›
  ‹rule ent_refl›

end
```