```
theory Hnr_Diff_Arr
  imports Hnr Diff_Arr_Safe
begin

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

lemma kdm_init:
  assumes
    "S' ⊆ S"
    "S' = S''"
  shows
    "master_assn' S ⟹A master_assn' S''"

lemma kdm_keep:
  assumes
    "S' ⊆ S"
  shows
    "insert x S' ⊆ insert x S"

lemma kdm_drop:
  assumes
    "S' ⊆ S"
  shows
    "S' ⊆ insert x S"

method kdm_subset = ((rule kdm_keep | rule kdm_drop)+)?, rule subset_refl

method kdm_check_not_empty = then_else
  ‹rule refl[of "{}"]›
  ‹fail›
  ‹rule refl›

method kdm = rule kdm_init, kdm_subset, kdm_check_not_empty

method diff_arr_kdm = rule ent_refl | kdm

end
```