```
lemma hnr_recursion:
  assumes
    mono_option "⋀x. mono_option (λr. f r x)"
  and
    step: "⋀r ri x xi F. (⋀x' xi' F'. hnr (Γ F' x' xi') (ri xi') (Γ' F' x' xi') (r x'))
          ⟹ hnr (Γ F x xi) (fi ri xi) (Γ'' F x xi) (f r x)"
  and
    norm: "⋀F x xi r ri. Norm (Γ'' F x xi r ri) (Γ' F x xi r ri)"
  and
    mono_heap: "⋀x. mono_Heap (λr. fi r x)"
  shows
    "hnr (Γ F x xi) (heap.fixp_fun fi xi) (Γ' F x xi) (option.fixp_fun f x)"

lemma tuple_selector_refl: "fst (a, b) = fst (a, b)" "snd (a, b) = snd (a, b)"

method hnr_recursion
  for Γ::"'F ⇒ 'x ⇒ 'xi ⇒ assn" and Γ'::"'F ⇒ 'x ⇒ 'xi ⇒ 'r ⇒ 'ri ⇒ assn"
  methods frame_match_atom =
  rule hnr_recursion[where Γ=Γ and Γ'=Γ', framed],
  ((subst tuple_selector_refl, simp only: fst_conv snd_conv)+)?,
  hnr_frame_inference frame_match_atom

method_setup partial_function_mono_setup =
  ‹Scan.succeed (SIMPLE_METHOD' o Partial_Function.mono_tac)›

method partial_function_mono = partial_function_mono_setup; fail

method hnr_solve_recursive_call methods frame_match_atom =
  rule hnr_frame[rotated], assumption, hnr_frame_inference frame_match_atom

end
```