```
ncpd goal = residualize ∘ drive ∘ normalize (goal)
drive     = drive_disj ∪ drive_conj

drive_disj :: Disjunction → Process_Tree
drive_disj D@($c_1$, ..., $c_n$) =
  create_or_node ([$c_i$ ← drive_conj ($c_i$)])

drive_conj :: (Conjunction, Substitution) → Process_Tree
drive_conj (($r_1$, ..., $r_n$), subst) =
  C@($r_1$, ..., $r_n$) ← propogate substitution subst on $r_1$, ..., $r_n$
  switch whistle (C) of
    instance (C', subst')       → create_fold_node (C', subst')
    embedded_but_not_instance → create_stop_node (C , subst )
    otherwise →
    | r ← heuristically_select_a_call ($r_1$, ..., $r_n$)
    | if r
    | then
    | | t ← drive ∘ normalize ∘ unfold (r)
    | | if trivial ∘ leafs (t)
    | | then
    | | | C' ← propagate_subst (C \ r, extract_subst (t))
    | | | drive C'[r ↦ extract_calls (t)]
    | | else
    | | | t $\bigwedge$ drive (C \ r, subst)
    | else
    | | $\bigwedge_{i=1}^{n}$ $t_i$ ← drive ∘ normalize ∘ unfold ($r_i$)
```