```
1  ncpd goal = residualize ∘ drive ∘ normalize (goal)
2  drive     = drive_disj ∪ drive_conj
3
4  drive_disj :: Disjunction → Process_Tree
5  drive_disj D@(c_1, ..., c_n) = ⋁_{i=1}^{n} t_i ← drive_conj (c_i)
6
7  drive_conj :: (Conjunction, Substitution) → Process_Tree
8  drive_conj ((r_1, ..., r_n), subst) =
9    C@(r_1, ..., r_n) ← propagate_substitution subst on r_1, ..., r_n
10   case whistle (C) of
11   | instance (C', subst')      ⇒ create_fold_node (C', subst')
12   | embedded_but_not_instance ⇒ create_stop_node (C, subst)
13   | otherwise ⇒
14   | | case heuristically_select_a_call (r_1, ..., r_n) of
15   | | | Just r ⇒
16   | | | | t ← drive ∘ normalize ∘ unfold (r)
17   | | | | if trivial ∘ leafs (t)
18   | | | | then
19   | | | | | C' ← propagate_substitution (C \ r, extract_substitution (t))
20   | | | | | drive C'[r ↦ extract_calls (t)]
21   | | | | else
22   | | | | | t ⋀ drive (C \ r, subst)
23   | | | Nothing ⇒ ⋀_{i=1}^{n} t_i ← drive ∘ normalize ∘ unfold (r_i)
```