```
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) =
6    create_or_node ([c_i ← drive_conj (c_i)])
7
8  drive_conj :: (Conjunction, Substitution) → Process_Tree
9  drive_conj (C@(r_1, ..., r_n), subst) =
10   r_1, ..., r_n ← propogate substitution subst on r_1, ..., r_n
11   switch whistle (C) of
12     instance (C', subst')        → create_fold_node (C', subst')
13     embedded_but_not_instance → create_stop_node (C , subst )
14     otherwise →
15       r ← select_a_call (r_1, ..., r_n)
16       t ← drive ∘ normalize ∘ unfold (r)
17       if trivial ∘ leafs (t)
18       then
19         C' ← propagate_subst (C \ r, extract_subst (t))
20         drive C'[r ↦ extract_calls (t)]
21       else
22         t ⋀ drive (C \ r, subst)
```