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