

```

1  ncpd goal = residualize o drive o normalize (goal)
2  drive      = drive_disj ∪ drive_conj
3
4  drive_disj :: Disjunction → Process_Tree
5  drive_disj D@(c1, ..., cn) =  $\bigvee_{i=1}^n t_i \leftarrow \text{drive\_conj } (c_i)$ 
6
7  drive_conj :: (Conjunction, Substitution) → Process_Tree
8  drive_conj ((r1, ..., rn), subst) =
9    C@(r1, ..., rn) ← propagate_substitution subst on r1, ..., rn
10   switch 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   | | r ← heuristically_select_a_call (r1, ..., rn)
15   | | if r
16   | | then
17   | | | t ← drive o normalize o unfold (r)
18   | | | if trivial o leafs (t)
19   | | | then
20   | | | | C' ← propagate_substitution (C \ r, extract_substitution (t))
21   | | | | drive C'[r ↦ extract_calls (t)]
22   | | | else
23   | | | | t ∧ drive (C \ r, subst)
24   | | else
25   | | |  $\bigwedge_{i=1}^n t_i \leftarrow \text{drive o normalize o unfold } (r_i)$ 

```