1 Explanation Generation

Algorithm 1: CSP-Explain($\mathcal{T}$, f [, $\mathcal{I}_0$])

input : $\mathcal{T}$ *set of constraints*
input : f *a consistent objective function*
optional input: $\mathcal{I}_0$ *a partial interpretation*
output : *Explanation sequence*

begin
  $\mathcal{I}_{end} \leftarrow$ `propagate(`$\mathcal{I}_0$`, `$\mathcal{T}$`)` `// Goal state`
  $\mathcal{I} \leftarrow \mathcal{I}_0$ `// Initial partial interpretation`
  $Seq \leftarrow \emptyset$ `// explanation sequence`
  while $\mathcal{I} \neq \mathcal{I}_{end}$ **do**
    for $i \in \mathcal{I}_{end} \setminus \mathcal{I}$ **do**
      $X_i \leftarrow$ `OMUS(`$\{\neg i\} \wedge \mathcal{I} \wedge \mathcal{S}$`)`
      $E_i \leftarrow \mathcal{I} \cap X_i$ `// Facts used`
      $S_i \leftarrow \mathcal{T} \cap X_i$ `// Constraint used`
      $\mathcal{N}_i \leftarrow$ `propagate(`$E_i \wedge \mathcal{S}_i$`)` `// Newly derived facts`
    end
    $(E_{best}, S_{best}, N_{best}) \leftarrow (E_i, S_i, N_i)$ with lowest $f(E_i, S_i, N_i)$
    append $(E_{best}, S_{best}, N_{best})$ to Seq
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{N_{best}\}$
  end
end

Algorithm 2: CSP-Explain-Incremental($\mathcal{T},\ f\ [,\ \mathcal{I}_0]$)

$\mathcal{I}_{end} \leftarrow$ `propagate(`$\mathcal{I}_0$`,` $\mathcal{T}$`)`
$\mathcal{I} \leftarrow \mathcal{I}_0$
$,\mathcal{W} \leftarrow \emptyset$ `// Collection of MSSes generated during OMUS`
$Seq \leftarrow \emptyset$
while $\mathcal{I} \neq \mathcal{I}_{end}$ **do**
  for $i \in \mathcal{I}_{end} \setminus \mathcal{I}$ **do**
    $X_i,\ \mathcal{MSS} \leftarrow$ `OMUS-DELAYED(`$\{\neg i\} \wedge \mathcal{I} \wedge \mathcal{S}$`,` $\mathcal{M}$`)`
    $E_i \leftarrow \mathcal{I} \cap X_i$
    $S_i \leftarrow \mathcal{T} \cap X_i$
    $\mathcal{N}_i \leftarrow$ `propagate(`$E_i \wedge \mathcal{S}_i$`)`
    $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{MSS}$
  end
  $(E_{best}, S_{best}, N_{best}) \leftarrow (E_i, S_i, N_i)$ with lowest $f(E_i, S_i, N_i)$
  append $(E_{best}, S_{best}, N_{best})$ to Seq
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{N_{best}\}$
end

Algorithm 3: `OMUS-Delayed`($\mathcal{F}, \mathcal{M}, f_{cost}$)

```
   // F = unsatisfiable CNF formula; M = Collection of MSSes; f_cost = cost function
 1 K ← ∅
   // grow mss from input mss
 2 foreach MSS ∈ M do
 3 |  MSS' ← Grow(F ∩ MSS)
 4 |  K ← K ∪ {F \ MSS'}
 5 end
 6 mode ← mode_greedy
 7 while true do
 8 |  while true do
 9 |  |  switch nonOptLevel do
10 |  |  |  case mode_incr
11 |  |  |  |  hs ← FindIncrementalHittingSet(K, C, hs)
12 |  |  |  case mode_greedy
13 |  |  |  |  hs ← FindGreedyHittingSet(K)
14 |  |  end
15 |  |  (sat?, κ) ← SatSolver(hs)
16 |  |  if not sat? then
17 |  |  |  switch nonOptLevel do
18 |  |  |  |  case mode_incr
19 |  |  |  |  |  mode ← mode_greedy
20 |  |  |  |  case mode_greedy
21 |  |  |  |  |  mode ← mode_opt
22 |  |  |  |  |  break
23 |  |  |  end
24 |  |  else
25 |  |  |  MSS ← Grow(hs)
26 |  |  |  M ← M ∪ {MSS}
27 |  |  |  K ← K ∪ {F \ MSS}
28 |  |  |  mode ← mode_incr
29 |  end
30 |  hs ← OptimalHittingSet(K, f_cost)
31 |  (sat?, κ) ← SatSolver(hs)
32 |  if not sat? then
33 |  |  return hs, M
34 |  end
35 |  MSS ← Grow(hs)
36 |  M ← M ∪ {MSS}
37 |  K ← K ∪ {F \ MSS}
38 |  mode ← mode_incr
39 end
```

Algorithm 4: `OMUS-Delayed`$(\mathcal{F}, \mathcal{M}, f_{cost})$

```
   // F = unsatisfiable CNF formula; M = Collection of MSSes; f_cost = cost function
 1 K ← ∅
   // grow mss from input mss
 2 foreach MSS ∈ M do
 3 |   MSS' ← Grow(F ∩ MSS)
 4 |   K ← K ∪ {F \ MSS'}
 5 end
 6 mode ← mode_greedy
 7 while true do
 8 |   switch nonOptLevel do
 9 |   |   case mode_incr
10 |   |   |   MCS ← {F \ MSS}
11 |   |   |   hs ← FindIncrementalHittingSet(K, MCS, hs)
12 |   |   case mode_greedy
13 |   |   |   hs ← FindGreedyHittingSet(K)
14 |   |   case mode_opt
15 |   |   |   hs ← OptimalHittingSet(K, W)
16 |   end
17 |   (sat?, κ) ← SatSolver(hs)
18 |   if not sat? then
19 |   |   switch nonOptLevel do
20 |   |   |   case mode_incr
21 |   |   |   |   mode ← mode_greedy
22 |   |   |   case mode_greedy
23 |   |   |   |   mode ← mode_opt
24 |   |   |   |   break
25 |   |   |   case mode_opt
26 |   |   |   |   return hs, M
27 |   |   end
28 |   else
29 |   |   MSS ← Grow(hs)
30 |   |   M ← M ∪ {MSS}
31 |   |   K ← K ∪ {F \ MSS}
32 |   |   mode ← mode_incr
33 end
```