1 Explanation Generation

Algorithm 1: ExplainCSP$(\mathcal{C}, U, f, I)$

```
Input     : $\mathcal{C}$, a CNF $\mathcal{C}$ over a vocabulary $V$
Input     : $U$, a user vocabulary $U \subseteq V$
Input     : $f$, a cost function $f : 2^{lits(U)} \rightarrow \mathbb{N}$.
Input     : $I$, a partial interpretation over $U$
Output    : $E$, a sequence of explanation steps as implications $I_{expl} \implies N_{expl}$
$\texttt{SAT} \leftarrow$ InitSat$(\mathcal{C})$
$E \leftarrow \langle\rangle$
$I_{end} \leftarrow$ OptimalPropagate$(U, I)$
while $I \neq I_{end}$ do
    $X \leftarrow$ bestStep$(\mathcal{C}, f, I_{end}, I)$
    $I_{best} \leftarrow I \cap X$
    $N_{best} \leftarrow$ OptimalPropagate$(U, I_{best}) \setminus I$
    add $\{I_{best} \implies N_{best}\}$ to $E$
    $I \leftarrow I \cup N_{best}$
end
return $E$
```

Algorithm 2: OptimalPropagate$(\mathcal{U}[, I], \texttt{SAT})$

```
Input     : $U$, a user vocabulary $U \subseteq V$
Optional: $I$, a partial interpretation over $U$.
State     : SAT, a SAT solver initialised with a CNF.
Output    : The projection onto $U$ of the intersection more precise than $I$.
$sat?, \mu \leftarrow \texttt{SAT.solve(}I\texttt{)}$
$\mu \leftarrow \{l \mid \texttt{var}(l) \in U\}$
$b \leftarrow$ a new blocking variable
while $true$ do
    $\texttt{SAT.addClause(}\neg b_i \bigvee_{l \in \mu} \neg l\texttt{)}$
    $sat?, \mu' \leftarrow \texttt{SAT.solve(}I \wedge \{b_i\}\texttt{)}$
    if $not\ sat?$ then
        $\texttt{SAT.addClause(}\neg b_i\texttt{)}$
        return $\mu$
    end
    $\mu \leftarrow \mu \cap \mu'$
end
```

Algorithm 3: bestStep–c-OUS$(f, I_{end}, I, \texttt{SAT})$

```
Input     : $f$, a cost function $f : 2^{\mathcal{G}} \rightarrow \mathbb{N}$ over CNF $\mathcal{G}$.
Input     : $I_{end}$, the cautious consequence, the set of literals that hold in all models.
Input     : $I$, a partial interpretation s.t. $I \subseteq I_{end}$.
State     : SAT, a SAT solver initialised with a CNF.
Output    : a single best explanation step
$\mathcal{A} \leftarrow I \cup (\overline{I_{end} \setminus I})$                // Optimal US is subset of A
set $p \triangleq \sum_{l \in \overline{I_{end}}} l = 1$ i.e. exactly one of $\overline{I_{end}}$ in the unsatisfiable subset.
return c-OUS$(f, p, \mathcal{A})$
```

```
Algorithm 4: c-OUS($f, p, \mathcal{A}$, SAT)
Input  : $f$, a cost function $f : 2^{\mathcal{A}} \rightarrow \mathbb{N}$..
Input  : $p$, a predicate $p : 2^{\mathcal{A}} \rightarrow \{t, f\}$..
Input  : $A$, a set of assumption literals, s.t. $\mathcal{C} \cup A$ is unsatisfiable.
State  : SAT, a SAT solver initialised with a CNF.
Output : $(p, f) - OUS$, a subset $\mathcal{HS}$ of $\mathcal{A}$ that satisfies $p$ s.t. $\mathcal{C} \cup \mathcal{HS}$ is $UNSAT$ and $\mathcal{HS}$ is
         $f$-optimal.
$\mathcal{H} \leftarrow \emptyset$
while true do
    $\mathcal{HS} \leftarrow$ CondOptHittingSet($f, p, \mathcal{A}, \mathcal{H}$)
    Add mode to algorithm for postponing optimisation. Mode switch corresponds to switching
    between heuristic conditional Hitting set computation and the optimizer.
    $sat?,\ \mu \leftarrow$ SAT($\mathcal{HS}$)
    if $\neg$SAT($\mathcal{HS}$) then
        return $\mathcal{HS}$
    end
    $SS \leftarrow$ Grow($C, f, p, A, HS$)
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathcal{A} \setminus \mathcal{SS}\}$
end
```

2 MIP model

We assume $U \subseteq V$ a set of user variables U defined over a vocabulary V of the CNF $\mathcal{C}$. Given an initial assignment I, where $vars(I) \subseteq U$, I_{end} is as the cautious consequence (the set of literals that hold in all models) of $\mathcal{C}$.

The Mixed Integer Programming model for computing c-OUSes has many similarities with a set covering problem. The CondOptHittingSet computes the optimal hitting set over a collection of sets $\mathcal{H}$, where optimal means "among those satisfying p".

In practice, to ensure that MIP model takes advantage of the incrementality of the problem, namely across different c-OUS calls, the specification is defined on the full set of literals of I_{end}. The constrained optimal hitting set is described by

- $x_l = \{0, 1\}$ is a boolean decision variable if the literal is selected or not.
- $w_l = f(l)$ is the cost assigned to having the literal in the hitting set (∞ otherwhise).
- $c_{lj} = \{0, 1\}$ is 1 (0) if the literal l is (not) present in hitting set j.

$$\underset{x}{min} \sum_{l\ \in I_{end} \cup \overline{I_{end}}} w_l \cdot x_l \tag{1}$$

$$\sum_{l\ \in I_{end} \cup \overline{I_{end}}} x_l \cdot c_{lj} \geq 1,\ \forall j \in \{1..|hs|\} \tag{2}$$

The basic specification finds the optimal hitting set

$$\sum_{l\ \in \overline{I_{end}} \setminus \bar{I}} x_l \geq 1,\ \forall j \in \{1..|hs|\} \tag{3}$$

2.1 Optimal Unsatisfiable Subsets

Algorithm 5: OUS($f, \mathcal{A}, \texttt{SAT}$)

```
Input  : $f$, a cost function $f : 2^{\mathcal{A}} \rightarrow \mathbb{N}$..
Input  : $A$, a set of assumption literals, s.t. $\mathcal{C} \cup A$ is unsatisfiable.
State  : SAT, a SAT solver initialised with a CNF.
Output : $f - OUS$, A subset $\mathcal{A}'$ of $\mathcal{A}$ s.t. $\mathcal{C} \cup \mathcal{A}'$ is $UNSAT$ and $\mathcal{A}'$ is $f$-optimal.
$\mathcal{H} \leftarrow \emptyset$
while true do
    $\mathcal{A}' \leftarrow$ OptHittingSet($f, \mathcal{A}, \mathcal{H}$)
    if $\neg SAT(\mathcal{A}')$ then
        return $\mathcal{A}'$
    end
    $A'' \leftarrow$ Grow($f, A', A$)
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathcal{A} \setminus \mathcal{A}''\}$
end
```

Algorithm 6: Postponing hitting set optimization for OUS

```
while true do
    while $|\mathcal{H}| > 0$ do
        $A' \leftarrow A' + min_f$ element of last MCS in $\mathcal{H}$;
        if $\neg SAT(A')$ then
            break
        end
        $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathcal{F} \setminus$ Grow($f, A', A$)$\}$ ;
    end
    $A' \leftarrow$ GreedyHittingSet($\mathcal{H}, f$);
    if $\neg SAT(A')$ then
        break
    end
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathcal{F} \setminus$ Grow($f, A', A$)$\}$ ;
end
```