Supplementary Material for TACAS Submission: Early Verification of Legal Compliance via Bounded Satisfiability Checking

Anonymous Author(s)

March 2022

In this document, we provide the supplementary material for our TACAS submission: "Early Verification of Legal Compliance via Bounded Satisfiability Checking". Specifically, Sec A provides the full semantics of metric first-order temporal logic (MFOTL) and the transformation rules from MFOTL to first-order logic (Fig. 2); Sec B provided the correctness proof for the constructions of over- and under-approximation queries; Sec C illustrated the IBS algorithm; and studied of IBS's correctness (Th. 1), termination (Th,2) and optimality (Th.3).

A MFOTL Semantics and translation from MFOTL to First-Order Logic

In this section, we provide the full semantics of MFOTL (Fig. 1) and the translation rules to FOL (Fig. 2).

A.1 MFOTL semantics

Se.c 2 of the main paper introduced the syntax and semantics of MFOTL. In this section, we provide the full semantics of the logic in Fig. 1.

$$
\begin{array}{lll}
(\bar{D}, \bar{\tau}, v, i) \models t \cong t' & \text{iff} & v(t) \cong v(t') \\
(\bar{D}, \bar{\tau}, v, i) \models t > t' & \text{iff} & v(t) > v(t') \\
(\bar{D}, \bar{\tau}, v, i) \models r(t_1, .., t_{i(r)}) & \text{iff} & r(v(t_1), .., v(t_{i(r)})) \in r^{D_i} \\
(\bar{D}, \bar{\tau}, v, i) \models \neg\phi & \text{iff} & (\bar{D}, \bar{\tau}, v, i) \not\models \phi \\
(\bar{D}, \bar{\tau}, v, i) \models \phi \wedge \psi & \text{iff} & (\bar{D}, \bar{\tau}, v, i) \models \phi \text{ and } (\bar{D}, \bar{\tau}, v, i) \models \psi \\
(\bar{D}, \bar{\tau}, v, i) \models \exists x \cdot (r(\bar{t}^i_x) \wedge \phi) & \text{iff} & (\bar{D}, \bar{\tau}, v[x \to d], i) \models (r(\bar{t}^i_x)) \wedge \phi \text{ for some } d \in |\bar{D}| \\
(\bar{D}, \bar{\tau}, v, i) \models \bullet_I\ \phi & \text{iff} & (\bar{D}, \bar{\tau}, v, i+1) \models \phi \text{ and } \tau_{i+1} - \tau_i \in I \\
(\bar{D}, \bar{\tau}, v, i) \models \bigcirc_I\ \phi & \text{iff} & i \geq 1 \text{ and } (\bar{D}, \bar{\tau}, v, i-1) \models \phi \text{ and } \tau_i - \tau_{i-1} \in I \\
(\bar{D}, \bar{\tau}, v, i) \models \phi\ \mathcal{U}_I\ \psi & \text{iff} & \text{exists } j \geq i \text{ and } (\bar{D}, \bar{\tau}, j, v) \models \psi \text{ and } \tau_j - \tau_i \in I \\
 & & \text{and for all } k \in \mathbb{N}\ i \leq k < j \Rightarrow (\bar{D}, \bar{\tau}, k, v) \models \phi \\
(\bar{D}, \bar{\tau}, v, i) \models \phi\ \mathcal{S}_I\ \psi & \text{iff} & \text{exists } j \leq i \text{ and } (\bar{D}, \bar{\tau}, j, v) \models \psi \text{ and } \tau_i - \tau_j \in I \\
 & & \text{and for all } k \in \mathbb{N}\ i \geq k > j \Rightarrow (\bar{D}, \bar{\tau}, k, v) \models \phi
\end{array}
$$

Figure 1: MFOTL semantics

A.2 Translation from MFOTL to FOL

As mentioned in Sec. 4 of the main paper, the first step of our solution for MFOTL bounded satisfiability checking is to translates MFOTL formula into FOL formula. We provide the complete set of translation rules for translating MFOTL formulas into FOL in fig. 2.

$$
\begin{array}{lll}
T(t = t', \tau_i) & \to & t \cong t' \\
T(t > t', \tau_i) & \to & t > t' \\
T(r(t_1, .., t_{\iota(r)}), \tau_i) & \to & \exists o(arg_1, .., arg_{i(r)}) : r \wedge \bigwedge_{j=1}^{\iota(r)}(arg_j = t_j) \\
 & & \wedge(\tau_i = o.time) \\
T(\neg\phi, \tau_i) & \to & \neg T(\phi, \tau_i) \\
T(\phi \wedge \psi, \tau_i) & \to & T(\phi, \tau_i) \wedge T(\psi, \tau_i) \\
T(\exists x \cdot r(\bar{t}^i_x) \wedge \phi, \tau_i) & \to & \exists o : r \cdot T((r(\bar{t}^i_x) \wedge \phi)[x \to o.arg_i], \tau_i) \\
T(\bullet_I\ \phi, \tau_i) & \to & T(\phi, \tau_{i+1}) \wedge (\tau_{i+1} - \tau_i) \in I \\
T(\bigcirc_I\ \phi, \tau_i) & \to & T(\phi, \tau_{i-1}) \wedge (\tau_i - \tau_{i-1}) \in I \\
T(\phi\ \mathcal{U}_I\ \psi, \tau_i) & \to & \exists time \cdot (time \geq \tau_i \wedge (time - \tau_i) \in I \wedge T(\psi, time) \\
 & & \text{and } \forall time' \cdot (\tau_i \leq time' < time \Rightarrow T(\phi, time'))) \\
T(\phi\ \mathcal{S}_I\ \psi, \tau_i) & \to & \exists time \cdot (time \leq \tau_i \wedge (\tau_i - time) \in I \wedge T(\psi, time) \\
 & & \text{and } \forall time' \cdot (\tau_i \geq time' > time \Rightarrow T(\phi, time'))) \\
T(\phi) & \to & T(\phi, \tau_1)
\end{array}
$$

Figure 2: Translation rules from MFOTL to FOL

Proposition 1 (Quantifiers on relational objects). *For a MFOTL formula ϕ, the quantifiers for the FOL formula $T(\phi)$ are limited only to relational objects.*

B Correctness proof of Over- and Under- Approximation

Sec. 5.1 of the main paper introduced the construction of over and under-approximation query via algorithm GROUND. In this section, we first restate the GROUND algorithm (alg. 1) and the correctness lemmas (lemma 1 and lemma 2), then provide the proof of the lemmas.

Algorithm 1 G: ground a quantified FOL formula.

Input an FOL formula ϕ_f, and a domain of relational objects $D_{AU\downarrow}$.

Output a grounded quantifier-free formula ϕ_g over relational objects.

1: **if** IS_ATOM(ϕ_f) **then return** ϕ_f **end if**
2: **if** $\phi_f.op = \exists$ **then** *//process the existential operator*
3: $Cls \leftarrow \phi_f.class$
4: $newR \leftarrow$ NEWACT(Cls) *//creates a new relational object of class Cls*
5: **return** G (ϕ_f.body[$\phi_f.headAct \leftarrow newR$], $D_{AU\downarrow}$)
6: **end if**
7: **if** $\phi_f.op = \forall$ **then** *//process the universal operator*
8: $Cls \leftarrow \phi_f.class$
9: **return** $\bigwedge_{[r:Cls]\in D_{AU\downarrow}} r \Rightarrow G$ (ϕ_f.body[$\phi_f.head \leftarrow r$], $D_{AU\downarrow}$)
10: **end if**
11: **return** $\phi_f.op(G$ (ϕ_{child}, $D_{AU\downarrow}$) for ϕ_{child} in ϕ_f.body)

Proposition 2. *For every FOL formula ϕ_f translated from MFOTL formula ϕ and domain $D_{AU\downarrow}$, the grounded formula $\phi_g = G(\phi_f, D_{AU\downarrow})$ is quantifier-free and contains a finite number of variables and terms.*

Lemma 1 (Over-approximation Query). *For an FOL formula ϕ_f, and a domain $D_{AU\downarrow}$, if $\phi_g = G(\phi_f, D_{AU\downarrow})$ is UNSAT, then so is ϕ_f.*

Proof of Lemma 1 Suppose ϕ_g is UNSAT but there exists a solution v_f for ϕ_f in some domain D_{AU} (D_{AU} may be different from $D_{AU\downarrow}$). We show that we can always construct a solution v_g that satisfies ϕ_g, which causes a contradiction. First, we construct a solution v'_g for $\phi'_g = G(\phi_f, D_{AU})$ from the solution v_f (for ϕ_f). Then, we construct a solution v_g for ϕ_g from the solution v'_g for ϕ'_g.

We can construct a solution v'_g for ϕ'_g in $D_{AU} \cup NewRs$ where $NewRs$ are the new relational objects added by G. The encoding of G uses the standard way for expanding universally quantified expression by enumerating every relation object in D_{AU} (L:9). For every existentially quantified expression, there exists some relation object $r \in D_{AU}$ enabled by v_f that satisfies the expression in ϕ_f, whereas ϕ'_g contains a new relational object $r' \in NewRs$ for satisfying the same expression (L:5). Let $v_f(r) = v'_g(r')$ for r and r', and then v'_g is a solution to ϕ'_g.

To construct the solution v_g for $\phi_g = G(\phi_f, D_{AU\downarrow})$ from the solution v'_g for $\phi'_g = G(\phi_f, D_{AU})$, we consider the expansion of universally quantified expression in ϕ_f (L:7). For every relational objects in $r^+ \in D_{AU} - D_{AU\downarrow}$, G creates constraints (L:9) in ϕ'_g, but not in ϕ_g. On the other hand, for every relational objects in $r^- \in D_{AU\downarrow} - D_{AU}$, we disable r^- in the solution v_g, (i.e., $v_g(r^-) =$

$\perp$). Therefore, the constraints instantiated by r^- (at L:9) in ϕ_g are vacuously satisfied.

For every relational object $r \in D_{AU\downarrow} \cap D_{AU}$, we let $v_g(r) = v'_g(r)$, and all shared constraints in ϕ_g and ϕ'_g are satisfied by v_g and v'_g, respectively. Therefore, v_g is a solution to ϕ_g. Contradiction. □

Lemma 2 (Under-Approximation Query). *For an FOL formula ϕ_f, and a domain $D_{AU\downarrow}$, let $\phi_g = G(\phi_f, D_{AU\downarrow})$ and $\phi_g^\perp = \text{UNDERAPPROX}(\phi_f, D_{AU\downarrow})$. If σ is a solution to $\phi_g^\perp$, then there exists a solution to ϕ_f.*

Proof of Lemma 2 By construction, $\text{NONEWR}(NewRs, D_{AU\downarrow})$ guarantees that if σ is a solution to $\phi_g^\perp$ in the domain $D_{AU\downarrow}$, there exists a solution σ' to ϕ_g in the domain $D_{AU\downarrow} \cup NewRs$. Since every relational object $r \in D_{AU\downarrow}$ has been used to instantiate a universally-quantified expression (L:9 of Alg. 1), σ' is also a solution to ϕ_f. □

Suppose, for some domain $D_{AU\downarrow}$, an over-approximation query ϕ_g for an FOL formula ϕ_f is satisfiable while the under-approximation query $\phi_g^\perp$ is UNSAT. Then we cannot conclude satisfiability of ϕ_f for $D_{AU\downarrow}$. However, the solution to ϕ_g provides hints on how to expand $D_{AU\downarrow}$ to potentially obtain a satisfying solution for ϕ_f. □

C Incremental Search for Bounded Counterexamples

Sec. 5.2 of the main paper presented our incremental bounded satisfiability checking approach IBS. In this section, we first restate the algorithm IBS (alg. 2) and provide an example illustrating IBS iterations with our IBS algorithm. Then we study of Alg. 2's correctness, termination and optimality.

Algorithm 2 IBS: search for a bounded (by b_{vol}) solution to $T(\phi) \bigwedge_{\psi \in Reqs} T(\psi)$.

Input an MFOTL formula ϕ, and MFOTL requirements $Reqs = \{\psi_1, \psi_2, ...\}$.

Optional Input b_{vol}, the volume bound, and data constraints T_{data}.

Output a counterexample σ, UNSAT or bounded-UNSAT.

```
1:  Reqs_f ← { ψ_f = T(ψ) | ψ ∈ Reqs}
2:  φ_f ← T(φ)
3:  Reqs_↓ ← ∅ //initially empty requirement
4:  D_AU↓ ← ∅ //initially empty domain
5:  while ⊤ do
6:      //over- and under-approximation
7:      φ ← φ_f ∧ Reqs_↓
8:      φ_g ← G(φ, D_AU↓)
9:      φ_g^⊥ ← UnderApprox(φ, D_AU↓)
10:     if Solve(φ_g ∧ T_data) = UNSAT then
11:         return UNSAT
12:     end if
13:     σ ← Solve(φ_g^⊥ ∧ T_data)
14:     if σ = UNSAT then //expand D_AU↓
15:         //compute the minimal solution
16:         σ_min ← Minimize(φ_g)
17:         //expand based on σ_min
18:         D_AU↓ += {act | act ∈ σ_min}
19:         if vol(σ_min) > b_vol then
20:             return bounded-UNSAT
21:         end if
22:     else //checks all requirements
23:         if σ ⊨ ψ for every ψ ∈ Reqs_f then
24:             return σ
25:         else
26:             lesson ← {ψ ∈ Reqs_f | σ ⊭ ψ}
27:             Reqs_↓.add(lesson)
28:         end if
29:     end if
30: end while
```

C.1 Illustration of IBS

Suppose a data collection centre (DCC) *collect*s and *access*es personal data information with three requirements: req_0 stating that no data is allowed to be accessed before being having been collected for 15 days (360 hours); req_1: data can only be updated after having been collected or last updated for more than a week (168 hours); and req_2: data can only be accessed if has been collected or updated within a week (168 hours). The signature S_{data} for DCC contains three binary relations (R_{data}): *Collect*, *Update*, and *Access*, such that *Collect*(d, v), *Update*(d, v) and *Access*(d, v) hold at a given time point if and only if data at id d is collected, updated, and accessed with value v at this time point, respectively. The MFOTL formulas for $P1$, req_0, req_1 and req_2 are shown in Fig. 3. For instance, the formula req_0 specifies that if a data value stored at id d is accessed, then some data must have been collected and stored at id d before, and the collection must have occurred prior to 360 hours ago ($\blacklozenge_{[360,)}$]). Suppose IBS is invoked to find a counterexample for property $P1$ (shown in Fig. 3) subject to requirements $Reqs = \{req_1, req_2\}$ with the bound $b_{vol} = 4$. IBS translates the requirements and the property to FOL, and initializes $Reqs_{\downarrow}$ and $D_{AU\downarrow}$ to empty sets. For each iteration, we use ϕ_g and $\phi_g^{\perp}$ to represent the over- and under-approximation queries computed on LL:8-9, respectively.

1st iteration: $D_{AU\downarrow} = \emptyset$ and $Reqs_{\downarrow} = \emptyset$. ϕ_g introduces three new relational objects (from $\neg P1$): $access_1$, $collect_1$ and $update_1$ such that: (C1) $access_1$ occurs after $collect_1$ and $update_1$; (C2) $access_1.arg1 = collect_1.arg1 = update_1.arg1$; (C3) $access_1.arg2 \neq collect_1.arg2 \wedge access_1.arg2 \neq update_1.arg2$; and (C4) either $collect_1$ or $update_1$ must be present in the counterexample. ϕ_g is satisfiable, but $\phi_g^{\perp}$ is UNSAT since $D_{AU\downarrow}$ is an empty set. WLOG, we assume $D_{AU\downarrow}$ is expanded by adding $access_1$ and $update_1$.

2nd iteration: $D_{AU\downarrow} = \{access_1, update_1\}$ and $Reqs_{\downarrow} = \emptyset$. ϕ_g stays the same, and $\phi_g^{\perp}$ is now satisfiable since $access_1$ and $update_1$ are in $D_{AU\downarrow}$. Suppose the solution is σ_3 (see Fig. 4). However, σ_3 violates req_2, so req_2 is added to $Reqs_{\downarrow}$.

3rd iteration: $D_{AU\downarrow} = \{access_1, update_1\}$ and $Reqs_{\downarrow} = \{req_2\}$. ϕ_g introduces two new relational objects (from req_2): $collect_2$ and $update_2$ such that (C5) $collect_2.time \leq access_1.time \leq collect2.time + 168$; (C6) $update_2.time \leq access_1.time \leq update_2.time + 168$; (C7) $access_1.arg1 = collect_2.arg1 = update_2.arg1$; (C8) $access_1.arg2 = collect_2.arg2 = update_2.arg2$; and (C9) $collect_2$ or $update_2$ exists. ϕ_g is satisfiable, but $\phi_g^{\perp}$ is UNSAT because $update_2 \notin D_{AU\downarrow}$ and $update_1 \neq update_2$ (C8 conflicts with C3). Therefore, $D_{AU\downarrow}$ needs to be expanded. Assume $collect_2$ is added.

4th iteration: $D_{AU\downarrow} = \{access_1, update_1, collect_2\}$ and $Reqs_{\downarrow} = \{req_2\}$. ϕ_g stays the same, and $\phi_g^{\perp}$ is now satisfiable since $collect_2$ is in $D_{AU\downarrow}$. Suppose the solution is σ_4 (see Fig. 4). Since σ_4 violates req_1, req_1 is added to $Reqs_{\downarrow}$.

5th iteration: $D_{AU\downarrow} = \{access_1, update_1, collect_2\}$ and $Reqs_{\downarrow} = \{req_1, req_2\}$. Since req_1 is in $Reqs_{\downarrow}$, ϕ_g adds the following constraints (from req_1): (C9) $\neg(update_2.time - 168 \leq collect_1.time \leq update_2.time)$. Since (C9) conflicts with (C8),(C7) and (C1), $update_2$ cannot be in the solution of ϕ_g. ϕ_g is satisfiable if $collect_1$ (introduced in the 1st iteration) or $update_2$ (3rd iteration)

$P1 = \Box\ \forall d, v, v' (Access(d, v) \wedge v' \neq v) \implies \neg Update(d, v')\ \mathcal{S}\ (Update(d, v) \vee Collect(d, v)))$
If a personal health information is not accurate or not up-to-date, it should not be accessed.

$\neg P1 = \Diamond\ \exists d, v, v' (Access(d, v) \implies (v' \neq v \wedge \neg Update(d, v)\ \mathcal{S}\ (Update(d, v') \vee Collect(d, v'))))$

$req_0 = \Box\ \forall d, v (Access(d, v) \implies \blacklozenge_{[360,)}\ \exists v' \cdot Collect(d,\ v'))$
No data is allowed to be accessed before having been collected for at least 15 days (360 hours).

$req_1 = \Box\ \forall d, v (Update(d, v) \implies \neg(\exists v'' \cdot ((Update(d, v'') \wedge v'' \neq v) \vee Collect(d,\ v'')) \vee \blacklozenge_{[1,168]}\ \exists v' \cdot (Collect(d, v') \vee Update(d, v'))))$
Data can only be updated after having been collected or last updated for more than a week (168 hours).

$req_2 = \Box\ \forall d, v (Access(d, v) \implies \blacklozenge_{[0,168]}\ Collect(d, v) \vee Update(d, v))$
Data can only be accessed if has been collected or updated within a week (168 hours).

$req_3 = \Box\ \forall d, v (Collect(d, v) \implies \neg(\exists v'' \cdot (Collect(d, v'') \wedge v \neq v'') \vee \blacklozenge_{[1,)}\ \exists v' \cdot Collect(d, v')))$ Data cannot be re-collected.

Figure 3: Example requirements and the legal property $P1$ of DCC, with the signature $S_{data} = (\emptyset, \{Collect,\ Update,\ Access\}, \iota_{data})$, where $\iota_{data}(Collect) = \iota_{data}(Update) = \iota_{data}(Access) = 2$.

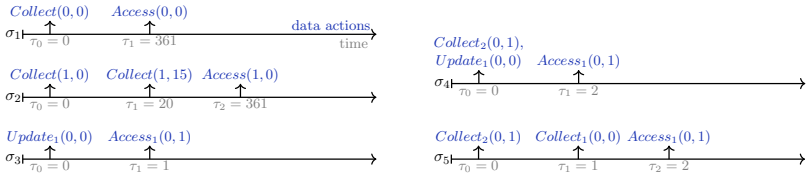

Figure 4: Several traces from the DCC example.

are in the solution. But $\phi_g^{\perp}$ is UNSAT since $D_{AU\downarrow}$ does not contain $collect_1$ or $update_2$. Thus, $D_{AU\downarrow}$ is refined. Assume $update_2$ is added to $D_{AU\downarrow}$.

6th iteration: $D_{AU\downarrow} = \{access_1, update_1, collect_2, update_2\}$ and $Reqs_{\downarrow} = \{req_1, req_2\}$. ϕ_g adds the constraints (C10) $update_2.time \geq update_1.time + 168$. ϕ_g (from req_1) and (C11) $update_2.time \leq update_1.time$ (from $\neg P$). Since (C10) conflicts with (C11), $update_2$ cannot exist in the solution of ϕ_g. Thus, ϕ_g is satisfiable only if $collect_1$ is in the solution. $\phi_g^{\perp}$ is UNSAT because $collect_1 \notin D_{AU\downarrow}$. Therefore, $D_{AU\downarrow}$ is expanded by adding $collect_1$.

final iteration: $D_{AU\downarrow} = \{access_1, update_1, collect_2, update_2,\ collect_1\}$ and $Reqs_{\downarrow} = \{req_1, req_2\}$. $\phi_g^{\perp}$ becomes satisfiable, and yields the solution σ_5 in Fig. 4. σ_5 satisfies req_1 and req_2, and is returned.

C.2 Correctness, Termination, Optimality of IBS

In this section, we state correctness of the IBS algorithm, then show that IBS always terminates when $b_{vol} \neq \infty$ and finally show that IBS always finds a solution with a minimum volume. Proofs of theorems in this section are available in supplementary material.

Theorem 1 (Soundness). *If the algorithm* IBS *terminates on input ϕ, Reqs and b_{vol}, then it returns the correct result, i.e., a counter-example σ, "UNSAT" or "bounded-UNSAT", when they apply.*

Proof. Let ϕ_f be the FOL formula $T(\phi) \bigwedge_{\psi \in Reqs} T(\psi)$. We consider correctness of IBS for three possible outputs: the satisfying solution σ to ϕ_f (L:24), the UNSAT determination of ϕ_f (L:11), and the bounded-UNSAT determination of ϕ_f (L:20). IBS returns a satisfying solution σ only if (1) σ is a solution $\phi_g^{\perp}$ (L:25) and (2) $\sigma \models T(\psi)$ for every $\psi \in Reqs$ (L:23). By (1) and Lemma 2, σ is a solution to $T(\phi) \bigwedge_{\psi \in Reqs_{\downarrow}} T(\psi)$. Together with (2), σ is a solution to

ϕ_f. IBS returns UNSAT iff ϕ_g is UNSAT (L:10). By Lemma 1, we show $T(\phi) \bigwedge_{\psi \in Reqs_\downarrow} T(\psi)$ is UNSAT. Since $Reqs_\downarrow \subseteq Reqs$, the original formula ϕ_f is also UNSAT. IBS returns bounded-UNSAT iff the volume of the minimum solution σ_{min} to the over-approximated query ϕ_g is larger than b_{vol} (L:19). Since ϕ_g is an over-approximation of the original formula ϕ_f, any solution σ to the ϕ_f has volume at least $vol(\sigma_{min})$. Therefore, when $vol(\sigma_{min}) > b_{vol}$, $vol(\sigma) > b_{vol}$ for every solution. □

Theorem 2 (Termination). *For an input property ϕ, requirements Reqs, and a bound $b_{vol} \neq \infty$,* IBS *eventually terminates.*

Proof. To prove that IBS always terminates when the input $b_{vol} \neq \infty$, we need to show that IBS does not get stuck at solving the SMT query via SOLVE (LL:13-10), nor refining $Reqs_\downarrow$ (LL:23-28), nor expanding $D_{AU\downarrow}$ (LL:18-22).

A call to SOLVE (LL:13-10) always terminates. By Prop. 1, $T(\phi)$ may contain quantifiers exclusively over relational objects. By Prop. 2 both the under- and the over-approximated queries ϕ_g and $\phi_g^\perp$ are quantifier-free. Since the background theory for ϕ is LIA, then ϕ_g and $\phi_g^\perp$ are a quantifier-free LIA formula whose satisfiability is decidable.

If the requirement checking fails on L: 23, a violating requirement *lesson* is added to $Reqs_\downarrow$ (LL:26-27) which ensures that any future solution σ' satisfies *lesson*. Therefore, *lesson* is never added to $Reqs_\downarrow$ more than once. Given that *Reqs* is a finite set of MFOTL formulas, at most $|Reqs|$ lessons can be learned before the algorithm terminates.

The under-approximated domain $D_{AU\downarrow}$ can be expanded a finite number of times because the size of the minimum solution $vol(\sigma_{min})$ to ϕ_g (computed on L:16) is monotonically non-decreasing between each iteration of the loop (LL:5-30). The size will eventually increase since each relational object in $D_{AU\downarrow}$ can introduce a finite number of options for adding a new relational object through the grounded encoding of ϕ_g on L:9, e.g., $r \Rightarrow \bigvee_{i=0}^{n} \exists r_i$. After exploring all options to $D_{AU\downarrow}$, $vol(\sigma_{min})$ must increase if the algorithm has not already terminated. Therefore, if $b_{vol} \neq \infty$, then eventually $vol(\sigma_{min}) > b_{vol}$, and the algorithm will return bounded-UNSAT instead of expanding $D_{AU\downarrow}$ indefinitely (LL:14-22). □

Optimality of the solution. The following theorem proves that the solution found by IBS has the minimum volume. the optimality of the solution found by IBS (i.e., the solution has the minimum volume).

Theorem 3 (Solution optimality). *For a property ϕ and requirements Reqs, let ϕ_f be the FOL formula $T(\phi) \bigwedge_{\psi \in Reqs} T(\psi)$. If* IBS *finds a solution σ for ϕ_f, then for every $\sigma' \models \phi_f$, $vol(\sigma) \leq vol(\sigma')$.*

Proof. IBS returns a solution σ on L:24 only if σ is a solution to the under-approximation query $\phi_g^\perp$ (computed on L:9) for some domain $D_{AU\downarrow} \neq \emptyset$. $D_{AU\downarrow}$ is last expanded in some previous iterations by adding relational objects to the minimum solution σ_{min} (L:16) of the over-approximation query ϕ'_g (L:18). Therefore, the returned σ has the same number of relational objects as σ_{min}

($vol(\sigma_{min}) = vol(\sigma)$). Since ϕ_g is an over-approximation of the original formula ϕ_f, any solution σ' to ϕ_f has volume that is at least $vol(\sigma_{min})$. Therefore, $vol(\sigma) \leq vol(\sigma')$. □