Supplementary Material

Anonymous Author(s)

March 2022

In this document, we provide the supplementary material for our TACAS submission. Specifically, it includes: (1) Our transformation rules for MFOTL to FOL (Fig. 2); (2) the proof of our over-approximation query lemma (Sec. 2); (3) an example illustrating our IBSC algorithm (Sec. 3); and (4) the study of Alg. 2's correctness (Th. 1), termination (Th,2) and optimality (Th.3).

1 Translation of MFOTL to First-Order Logic

In this section, we provide our translation rules together with the MFOTL semantics, as well as the proposition on the translation function.

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

Figure 1: MFOTL semantics

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

Figure 2: Translation rules from MFOTL to FOL

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

2 Over- and Under- Approximation

In this section, to have a self-contained document, we provide the proof of the Lemma 1 and Lemma 2 together with our grounding algorithm.

Algorithm 1 G: ground a quantified FOL formula.

Input an FOL formula ϕ_f.
Input a domain of relational objects $D_{AU\downarrow}$.
Output a grounded quantifier-free formula ϕ_g over relational objects.

if IS_ATOM(ϕ_f) **then return** ϕ_f **end if**
if $\phi_f.op = \exists$ **then** *//process the existential operator*
  $Cls \leftarrow \phi_f.class$
  $newR \leftarrow$ NEWACT(Cls) *//creates a new relational object of class Cls*
  return G (ϕ_f.body[$\phi_f.headAct \leftarrow newR$], $D_{AU\downarrow}$)
end if
if $\phi_f.op = \forall$ **then** *//process the universal operator*
  $Cls \leftarrow \phi_f.class$
  return $\bigwedge_{[r:Cls] \in D_{AU\downarrow}} r \Rightarrow G$ (ϕ_f.body[$\phi_f.head \leftarrow r$], $D_{AU\downarrow}$)
end if
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. 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., $v_g(r^-) = \bot$). 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. $\square$

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 $\phi_g^\perp = \textsc{UnderApprox}(\ \phi_f, D_{AU\downarrow})$. If σ is a solution to $\phi_g^\perp$, then there exists a solution to ϕ_f.*

Proof. By construction, $\textsc{NoNewR}(NewRs, D_{AU\downarrow})$ 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. $\square$

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. $\square$

3 Incremental Search for Bounded Counterexamples

In this section, we provide an example illustrating IBSC iterations with our IBSC algorithm, as well as the study of Alg. 2's correctness, termination and optimality.

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

Input an MFOTL formula ϕ.

Input a set of MFOTL requirements $Reqs = \{\psi_1, \psi_2, ...\}$.

Optional Input b_{vol}, the volume bound of the counterexample.

Optional Input data constraints T_{data}, default $\top$.

Output a counterexample σ, UNSAT or bounded-UNSAT.

1: $Reqs_f \leftarrow \{ \ \psi_f = T(\psi) \mid \psi \in Reqs\}$
2: $\phi_f \leftarrow T(\phi)$
3: $Reqs_{\downarrow} \leftarrow \emptyset$ *//initially do not consider any requirement*
4: $D_{AU\downarrow} \leftarrow \emptyset$ *//start with empty set of relational objects*
5: **while** $\top$ **do**
6:     $\phi \leftarrow \phi_f \wedge Reqs_{\downarrow}$
7:     $\phi_g \leftarrow G(\phi, D_{AU\downarrow})$ *//over-approximation query*
8:     $\phi_g^{\perp} \leftarrow$ UNDERAPPROX$(\phi, D_{AU\downarrow})$ *//under-approximation query*
9:     **if** SOLVE$(\phi_g \wedge T_{data})$ = UNSAT **then**
10:         **return** UNSAT
11:     **end if**
12:     $\sigma \leftarrow$ SOLVE$(\phi_g^{\perp} \wedge T_{data})$
13:     **if** σ = UNSAT **then** *//expand* $D_{AU\downarrow}$
14:         $\sigma_{min} \leftarrow$ MINIMIZE(ϕ_g) *//add relational object from the minimal solution*
15:         $D_{AU\downarrow}$ += $\{act \mid act \in \sigma_{min}\}$
16:         **if** $vol(\sigma_{min}) > b_{vol}$ **then**
17:             **return** bounded-UNSAT
18:         **end if**
19:     **else** *//checks all requirements*
20:         **if** $\sigma \models \psi$ for every $\psi \in Reqs_f$ **then**
21:             **return** σ
22:         **else**
23:             $lesson \leftarrow$ some $\psi \in Reqs_f$ such that $\sigma \not\models \psi$
24:             $Reqs_{\downarrow}$.add$(lesson)$ *//add violating requirement to* $Reqs_{\downarrow}$
25:         **end if**
26:     **end if**
27: **end while**

Example: 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 IBSC is invoked to find a counterexample for property $P1$ (shown in Fig. 3) subject to requirements $Reqs = \{req_1, req_2\}$ with the bound $b_{vol} = 4$. IBSC 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:7-8, 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) 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.

3.1 Correctness, Termination, Optimality of IBSC

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

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

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

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

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

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

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

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

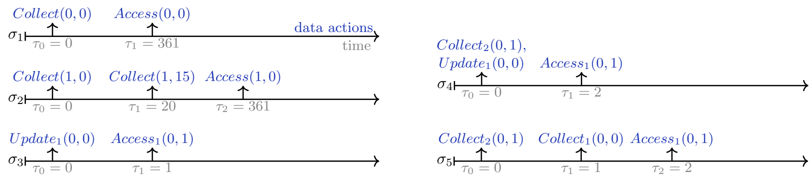

Figure 4: Several traces from the DCC example.

Theorem 1 (Soundness). *If the algorithm* IBSC *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 IBSC for three possible outputs: the satisfying solution σ to ϕ_f (L:21), the UNSAT determination of ϕ_f (L:10), and the bounded-UNSAT determination of ϕ_f (L:17). IBSC returns a satisfying solution σ only if (1) σ is a solution $\phi_g^\perp$ (L:22) and (2) $\sigma \models T(\psi)$ for every $\psi \in Reqs$ (L:20). 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. IBSC returns UNSAT iff ϕ_g is UNSAT (L:9). 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. IBSC 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:16). 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. $\square$

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

Proof. To prove that IBSC always terminates when the input $b_{vol} \neq \infty$, we need to show that IBSC does not get stuck at solving the SMT query via SOLVE (LL:12-9), nor refining $Reqs_\downarrow$ (LL:20-25), nor expanding $D_{AU\downarrow}$ (LL:15-19).

A call to SOLVE (LL:12-9) 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: 20, a violating requirement *lesson* is added to $Reqs_{\downarrow}$ (LL:23-24) 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:14) is monotonically non-decreasing between each iteration of the loop (LL:5-27). 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:8, 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:13-19). □

Optimality of the solution. The following theorem proves that the solution found by IBSC has the minimum volume. the optimality of the solution found by IBSC (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* IBSC *finds a solution σ for ϕ_f, then for every $\sigma' \models \phi_f$, $vol(\sigma) \leq vol(\sigma')$.*

Proof. IBSC returns a solution σ on L:21 only if σ is a solution to the under-approximation query $\phi_g^{\perp}$ (computed on L:8) 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:14) of the over-approximation query ϕ'_g (L:15). 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, $vol(\sigma) \leq vol(\sigma')$. □