A SUPPLEMENTARY MATERIALS

A.1 Theorems 3.2 and 3.3 and their proofs

Theorem 3.2 (Tuple and Empty Set Simplifiers)**.** *The following **tuple and empty set simplifiers***

1. $(r_1, \ldots, r_n) = (s_1, \ldots, s_n) \rightarrow$
   $r_1 = s_1 \wedge \cdots \wedge r_m = s_m$ *(or its negation for $\neq$).*
2. $t \in \{t_1, \ldots, t_n\} \rightarrow (t = t_1) \vee t \in \{t_2, \ldots, t_n\}$.
3. $t \in \bigcup_{i \in D:C} \phi \rightarrow \exists i \in D : (C \wedge t \in \phi)$.
4. $\left(\bigcup_{i \in D:C} \phi\right) = \emptyset \rightarrow \forall i \in D : (\neg C \vee \phi = \emptyset)$.
5. $S \cap \emptyset = \emptyset \rightarrow$ TRUE.
6. $S \cup \emptyset = \emptyset \rightarrow S = \emptyset$.
7. $S \cap \{t_1, \ldots, t_n\} = \emptyset \rightarrow$
   $(t_1 \notin S) \wedge (S \cap \{t_2, \ldots, t_n\} = \emptyset)$.
8. $\left(\bigcup_{i \in D:C} \phi\right) \cap \left(\bigcup_{i' \in D':C'} \phi'\right) = \emptyset \rightarrow$
   $\forall i \in D : C \Rightarrow \forall i' \in D' : C' \Rightarrow (\phi \cap \phi' = \emptyset)$.
9. $(S_1 \cup S_2) \cap S_3 = \emptyset \rightarrow (S_1 \cap S_3) \cup (S_2 \cap S_3) = \emptyset$.
10. $S_1 \cup S_2 = \emptyset \rightarrow S_1 = \emptyset \wedge S_2 = \emptyset$.
11. $\{t_1, \ldots, t_n\} = \emptyset \rightarrow$ FALSE *if* $n > 0$, TRUE *otherwise,*

when included in SEMT, rewrite $oc_f[E] = \emptyset$ expressions to equivalent formulas free of tuple and set expressions.

Proof. Intuitively, this theorem is analogous to an algorithm that converts propositional formulas into an equivalent disjunctive normal form (DNF), that is, a disjunction of conjunctive clauses (that is, conjunctions of literals). Once a DNF is reached, contradictory conjunctive clauses are eliminated, and therefore the formula is satisfiable if and only if there is at least one conjunctive clause with at least one literal. In this analogy, conjunctions and disjunctions correspond to intersection and union, and sets correspond to conjunctive clauses. Empty sets are eliminated, and if at the end we have a union of sets, and the empty ones have been eliminated, this means that the resulting set is not empty.

Formally, the theorem is proven by induction on the *distance vector*, a tuple that measures how far an expression is from being solved. Before we define the distance vector, we need to inductive define, for any expression E, the **intersection-union nesting** $N_{\cap\cup}(E)$:

$$N_{\cap\cup}(E) = \begin{cases} \sum_i N_{\cap\cup}(E_i), \text{ if } E = E_1 \cap \cdots \cap E_n \\ 1 + \max_i N_{\cap\cup}(E_i), \text{ if } E = E_1 \cup \cdots \cup E_n \\ 0, \text{ if } E \text{ is any other expression.} \end{cases}$$

Intuitively, $N_{\cap\cup}$ measures how far we are from a "flat" union of intersections.

The **distance vector** of an expression E is a vector of non-negative integers that is lexicographically ordered, with the most significant component listed first:

1. $N_{\cap\cup}(E)$;
2. number of intensional unions ($\bigcup$);
3. number of existential and universal quantifications;
4. number of $\cap$ applications;
5. sum of lengths of extensionally defined sets ($\{\ldots\}$);
6. number of $\in$ applications;
7. number of comparisons to $\emptyset$;
8. number of tuples.

In the base case, the distance vector is a tuple of zeros, and therefore the expression is a formula without any tuple or set operators, satisfying the theorem.

Otherwise, the distance vector contains at least one non-zero component. If we can show that there is always at least one applicable simplifier, and that every simplifier application results in an expression with a smaller distance vector with respect to the lexicographical order, then the theorem will have been proven by induction.

There is always an applicable simplifier to expressions with non-zero distance vector, because in that case there is at least one tuple operator or a comparison between a set expression and $\emptyset$:

- if there is a set expression, it must be one of $\cap$, $\cup$ applications or $\bigcup$, and there is at least one simplifier for each of these;
- If there is a tuple anywhere, it is either inside a tuple comparison, or inside a set; if it is in a comparison, simplifier 1 applies; if it is in a set, one of the set simplifiers applies.

Once it is established that there is always an applicable simplifier, the next step is whether the distance vector is always decreased according to its lexicographical order. Simplifiers 1, 2, 4, 5, 6, 10, and 11 strictly decrease one or more of the distance vector components without increasing any other, so for expressions for which any of them apply, the theorem is proven by induction on the distance vector.

The remaining simplifiers decrease a distance vector component while increasing others, but the ones increased are always less significant in the lexicographical order than the one decreased:

- Simplifier 3 decreases the number of intensional unions at the cost of the less significant number of existential quantifications;

- Simplifier 7 decreases the sum of lengths of extensionally defined sets at the cost of the less significant number of $\in$ applications;

- Simplifier 8 decreases the number of intensional unions at the cost of the less significant number of universal quantifications;

- Simplifier 9 duplicates S_3 and therefore *doubles* all distance vector components in S_3, with the exception the most significant one, $N_{\cap\cup}$, which is decreased:

$$\begin{aligned}
&N_{\cap\cup}((S_1 \cup S_2) \cap S_3 = \emptyset)\\
&= \max(N_{\cap\cup}((S_1 \cup S_2) \cap S_3), 0)\\
&= N_{\cap\cup}((S_1 \cup S_2) \cap S_3)\\
&= N_{\cap\cup}(S_1 \cup S_2) + N_{\cap\cup}(S_3)\\
&= 1 + \max(N_{\cap\cup}(S_1), N_{\cap\cup}(S_2)) + N_{\cap\cup}(S_3)\\
&= 1 + \max(N_{\cap\cup}(S_1) + N_{\cap\cup}(S_3),\\
&\qquad\qquad N_{\cap\cup}(S_2) + N_{\cap\cup}(S_3))\\
&= 1 + \max(N_{\cap\cup}(S_1 \cap S_3),\\
&\qquad\qquad N_{\cap\cup}(S_2 \cap S_3))\\
&= 1 + N_{\cap\cup}((S_1 \cap S_3) \cup (S_2 \cap S_3))\\
&> \quad N_{\cap\cup}((S_1 \cap S_3) \cup (S_2 \cap S_3))\\
&= \quad N_{\cap\cup}((S_1 \cap S_3) \cup (S_2 \cap S_3) = \emptyset).
\end{aligned}$$

To summarize, we have shown that, for every expression in the language of interest, there is always an applicable simplifier, and that all simplifiers decrease the distance vector until it reaches the base case all-zero distance vector, which is free of tuple and set operators. □

Theorem 3.3 (Inversion Modulo Theories)**.** *Let E be an expression and T_i, C_i be type and constraint, respectively, in a theory for which we have a satisfiability solver. Then,*

$$\bigoplus_{f \in A \to B} \quad \bigotimes_{x_1 \in T_1 : C_1} \cdots \bigotimes_{x_k \in T_k : C_k} E,$$

where $A = oc_f[\bigotimes_{x_1 \in T_1 : C_1} \cdots \bigotimes_{x_k \in T_k : C_k} E]$ (this is relaxed in Section 3.4), is equivalent to , and therefore can be rewritten as,

$$\bigotimes_{x_1 \in T_1 : C_1} \cdots \bigotimes_{x_k \in T_k : C_k} \quad \bigoplus_{f \,\in\, oc_f[E] \to B} E,$$

if $\bigotimes$ distributes over $\bigoplus$ and

$$\begin{aligned}
&\forall x_1' \in T_1 \ldots \forall x_k' \in T_k \quad \forall x_1'' \in T_1 \ldots \forall x_k'' \in T_k\\
&\begin{pmatrix} C_1[x_1/x_1'] \wedge \cdots \wedge C_k[x_k/x_k'] \\ \wedge\, C_1[x_1/x_1''] \wedge \cdots \wedge C_k[x_k/x_k''] \\ \wedge\, (x_1', \ldots, x_k') \neq (x_1'', \ldots, x_k'') \end{pmatrix}\\
&\Rightarrow \begin{pmatrix} oc_f[E][x_1/x_1', \ldots, x_k/x_k'] \\ \cap \\ oc_f[E][x_1/x_1'', \ldots, x_k/x_k''] \end{pmatrix} = \emptyset \qquad (2)
\end{aligned}$$

(that is, the set of argument tuples in applications of f in E for different value assignments to $x_1, \ldots, x_k$ are always disjoint — we refer to this as ***Condition (2)*** *from now on). Moreover, the rewritten expression is exponentially ($O(2^{\prod_i |\{x_i \in T_i : C_i\}|})$) cheaper to evaluate than the original.*

Proof. Let m be the number of possible assignments to $x_1, \ldots, x_m$. We prove the theorem by induction on m. If m is 0,

$$\begin{aligned}
&\bigoplus_{f \in A \to B} \quad \bigotimes_{x_1 \in T_1 : C_1} \cdots \bigotimes_{x_k \in T_k : C_k} E\\
&= \bigoplus_{f \in \emptyset \to B} 1\\
&= \bigotimes_{x_1 \in T_1 : C_1} \cdots \bigotimes_{x_k \in T_k : C_k} \quad \bigoplus_{f \,\in\, \emptyset \to B} 1\\
&= \bigotimes_{x_1 \in T_1 : C_1} \cdots \bigotimes_{x_k \in T_k : C_k} \quad \bigoplus_{f \,\in\, \emptyset \to B} E\\
&= \bigotimes_{x_1 \in T_1 : C_1} \cdots \bigotimes_{x_k \in T_k : C_k} \quad \bigoplus_{f \,\in\, oc_f[E] \to B} E,
\end{aligned}$$

because the empty products allow the substitution of 1 by E and $\emptyset$ by $oc_f[E]$ without change.

If $m > 0$, let $\bar{x}$ be the first possible assignment to $x_1, \ldots, x_k$ satisfying $C_1 \wedge \cdots \wedge C_k$. Then

$$\begin{aligned}
&\bigoplus_{f \in A \to B} \quad \bigotimes_{x_1 \in T_1 : C_1} \cdots \bigotimes_{x_k \in T_k : C_k} E\\
&= \text{(separating } \bar{x} \text{ from other assignments)}\\
&\bigoplus_{f_1 \in oc_f[E][x_1, \ldots, x_k/\bar{x}] \to B}\\
&\bigoplus_{f \in (A \setminus oc_f[E][x_1, \ldots, x_k/\bar{x}]) \to B}\\
&\qquad E_1 \quad \otimes\\
&\qquad\qquad \bigotimes_{(x_1, \ldots, x_k) \in T_1 \times \cdots \times T_k : C_1 \wedge \cdots \wedge C_k \wedge (x_1, \ldots, x_k) \neq \bar{x}} E,\\
&\qquad \text{where } E_1 = E[f/f_1]
\end{aligned}$$

$= (E_1$ has no occurrences of $f)$

$$\Big(\bigoplus_{f_1 \in oc_f[E][x_1,\ldots,x_k/\bar{x}] \to B} E_1\Big) \quad \otimes$$

$$\bigoplus_{f \in (A \setminus oc_f[E][x_1,\ldots,x_k/\bar{x}]) \to B}$$

$$\bigotimes_{(x_1,\ldots,x_k) \in T_1 \times \cdots \times T_k : C_1 \wedge \cdots \wedge C_k \wedge (x_1,\ldots,x_k) \neq \bar{x}} E$$

$=$ (by induction on m)

$$\Big(\bigoplus_{f_1 \in oc_f[E][x_1,\ldots,x_k/\bar{x}] \to B} E_1\Big) \quad \otimes$$

$$\bigotimes_{(x_1,\ldots,x_k) \in T_1 \times \cdots \times T_k : C_1 \wedge \cdots \wedge C_k \wedge (x_1,\ldots,x_k) \neq \bar{x}} \bigoplus_{f \in oc_f[E] \to B} E$$

$=$ (renaming f_1 to f and using the fact that $E_1 = E[f/f_1]$)

$$\Big(\bigoplus_{f \in oc_f[E][x_1,\ldots,x_k/\bar{x}] \to B} E\Big) \quad \otimes$$

$$\bigotimes_{(x_1,\ldots,x_k) \in T_1 \times \cdots \times T_k : C_1 \wedge \cdots \wedge C_k \wedge (x_1,\ldots,x_k) \neq \bar{x}} \bigoplus_{f \in oc_f[E] \to B} E$$

$=$ (introducing intensional products on $x_1, \ldots, x_k$ bound to $\bar{x}$)

$$\bigotimes_{(x_1,\ldots,x_k) \in T_1 \times \cdots \times T_k : C_1 \wedge \cdots \wedge C_k \wedge (x_1,\ldots,x_k) = \bar{x}}$$

$$\Big(\bigoplus_{f \in oc_f[E][x_1,\ldots,x_k/\bar{x}] \to B} E\Big) \quad \otimes$$

$$\bigotimes_{(x_1,\ldots,x_k) \in T_1 \times \cdots \times T_k : C_1 \wedge \cdots \wedge C_k \wedge (x_1,\ldots,x_k) \neq \bar{x}} \bigoplus_{f \in oc_f[E] \to B} E$$

$=$ (merging $\bigotimes$ by disjuncting their constraints)

$$\bigotimes_{(x_1,\ldots,x_k) \in T_1 \times \cdots \times T_k : C_1 \wedge \cdots \wedge C_k \wedge \Big((x_1,\ldots,x_k) = \bar{x} \vee (x_1,\ldots,x_k) \neq \bar{x}\Big)}$$

$$\bigoplus_{f \in oc_f[E] \to B} E$$

$=$ (eliminating tautology on $\bar{x}$ and separating $\bigotimes$ per index)

$$\bigotimes_{x_1 \in T_1 : C_1} \cdots \bigotimes_{x_k \in T_k : C_k} \bigoplus_{f \,\in\, oc_f[E] \to B} E$$

The final expression is $O(2^{\prod_i |\{x_i \in T_i : C_i\}|})$ cheaper to evaluate because the final f has all x_i bound to a single value, mapping each of their $\prod_i |\{x_i \in T_i : C_i\}|$ assignments to a single one and thus dividing the total size of the domain over which one must iterate. $\square$