1. Unions of Theories have the Pushout Property

Suppose $T_1$ and $T_2$ have signatures $\Sigma_1 = \text{sig}(T_1)$ and $\Sigma_2 = \text{sig}(T_2)$ that are nicely interpretable as $\Sigma_0 = \Sigma_1 \cap \Sigma_2$.

Let $T_0 = (\Sigma_0, \emptyset)$. We then have,

**Theorem (Pushout Property)** Let $T_i, 0 \leq i \leq 2$ be as above, and let $H_i : T_i \rightarrow T, 1 \leq i \leq 2$ be theory interpretations such that $H_1 | T_0 = H_2 | T_0$, i.e., such that $J_1 ; H_1 = J_2 ; H_2$ in the diagram below.

Then $H_1 \cup H_2$ is a theory interpretation $H_1 \cup H_2 : T_1 \cup T_2 \rightarrow T$, and is the only theory interpretation such that $H_1 = J_2 ; (H_1 \cup H_2)$, and $H_2 = J_1 ; (H_1 \cup H_2)$.

Proof. Uniqueness of $H_1 \cup H_2$ is guaranteed by the pushout property for $\Sigma_1 \cup \Sigma_2$ for the signature map $H_i : \Sigma_i \rightarrow \text{sig}(T)$, $1 \leq i \leq n$. We just need to show that $H_1 \cup H_2$ is a theory interpretation, that is, that

$$\text{mod}(T) \mid_{H_1 \cup H_2} \subseteq \text{mod}(T_1 \cup T_2).$$
But this follows immediately from the commutative diagram:

\[
\begin{array}{c}
\text{mod}(T_0) \leftarrow \text{mod}(T_1) \leftarrow \text{mod}(T_2) \\
\downarrow \downarrow \downarrow \\
\text{mod}(T_0) \leftarrow \text{mod}(T_1) \leftarrow \text{mod}(T_2) \\
\end{array}
\]

(where in fact \( -1_{H_1} \) and \( -1_{H_2} \) are well-defined functions, since \( \text{mod}(T_0)_H \subseteq \text{mod}(T_1)_H \), \( 1 \leq i \leq 2 \) by \( H_i \) theory interpretation

Hence \( \text{mod}(T_0)_H \subseteq \text{mod}(T_1)_H \), \( 1 \leq i \leq 2 \), thanks to the fact that \( \text{mod}(T_1U_2) = \text{mod}(T_1) \times \text{mod}(T_2) \), \( 1 \leq i \leq 2 \), is a pullback in the category of sets. \( \Box \)

**Corollary**. Let \( H_i : T_i \rightarrow T_i' \), \( 1 \leq i \leq 2 \) be theory interpretations such that \( \Sigma_i = \text{sig}(T_i) \), \( 1 \leq i \leq 2 \) are nicely intersectable as \( \Sigma_0 = \Sigma_1 \cap \Sigma_2 \), and \( \Sigma_i' = \text{sig}(T_i') \), \( 1 \leq i \leq 2 \) are likewise nicely intersectable as \( \Sigma_0' = \Sigma_1' \cap \Sigma_2' \), and \( H_2|_{\Sigma_0} = H_2|_{\Sigma_0} = H_0 \), so that \( H_1 \) and \( H_2 \) have a common restriction to a signature map \( H_0 : \Sigma_0 \rightarrow \Sigma_0' \). Then \( H_1 \cup H_2 : T_1U_2 \rightarrow T_1'U_2' \) in a theory interpretation, and the only one commutes the diagram:

\[
\begin{array}{c}
H_1 \rightarrow T_1 \longrightarrow T_1U_2' \rightarrow T_1'U_2' \\
\downarrow \downarrow \downarrow \\
T_1 \leftarrow T_1 \leftarrow T_1' \leftarrow T_1' \\
\end{array}
\]
2. Trivial and Expansive Ascent Maps are Closed Under Unions

To increase extensibility of SMT solving, ascent maps should be composable out of smaller pieces as much as possible. The fact that Ascent is a category already gives us a compositional way of obtaining new ascent maps from simpler ones by sequential composition (we exploited, for example, this fact in Lecture 23, pg. 14, last two compositions). Is there, furthermore, some way of composing ascent maps in parallel? The answer is yes! (under some conditions).

**Lemma** Let \((H_i, \xi_i): (T_i, \mathcal{F}_i) \rightarrow (T'_i, \mathcal{G}_i)\) \(1 \leq i \leq 2\) be trivial ascent maps such that \(H_i: T_i \rightarrow T'_i, 1 \leq i \leq 2\) are expansive and satisfy the conditions in the Corollary of the Pushout Property. Then (as in Lecture 23 above), suppose that \(\Sigma_0\) (and also \(\Sigma_0'\)) has no operations (only predicate symbols) and \(H_0: \Sigma_0 \rightarrow \Sigma_0'\) is surjective. Then, defining \(\mathcal{F}_1 \times \mathcal{F}_2 = \{ (\mathcal{F}_{1,i}, \mathcal{F}_{2,i}) | 1 \leq i \leq 2 \}\), and likewise \(\mathcal{G}_1 \times \mathcal{G}_2\),

\((H_{1 \uplus H_2}, H_{1 \uplus H_2}): (T_1 \cup T_2, \mathcal{F}_1 \times \mathcal{F}_2) \rightarrow (T'_1 \cup T'_2, \mathcal{G}_1 \times \mathcal{G}_2)\)

is an ascent map, called the **parallel composition** of the two original ascent maps.

**Proof.** \(H_{1 \uplus H_2}: T_1 \cup T_2 \rightarrow T'_1 \cup T'_2\) is well-defined by the Corollary, and \(H_{1 \uplus H_2}: \mathcal{F}_1 \times \mathcal{F}_2 \rightarrow \mathcal{G}_1 \times \mathcal{G}_2\) is a well-defined recursive function defined by \(H_1\) and \(H_2\). We just need to
show that: (i) $H_1UH_2 : T_1UT_2 \rightarrow T'_1UT'_2$ is expansive.

Let $[M_1', M_2'] \in \text{mod}(T_1UT_2)$, so that $M_1|_{\Sigma_0} = M_2|_{\Sigma_0}$.

By the Hi expansivity, $1 \leq i \leq 2$, we have $M_i' \in \text{mod}(T'_i)$, $1 \leq i \leq 2$ such that $M_i'|_{H_i} = M_i$, $1 \leq i \leq 2$.

We will be done if we show that $M_1'|_{\Sigma_0} = M_2'|_{\Sigma_0}$, since then $[M_1', M_2']|_{H_1UH_2} = [M_1', M_2']$. But since $\Sigma_0$ has no operations, this amounts to showing that for each sent $s_0'$ in $\Sigma_0$ we have $M_1|_{s_0'} = M_2|_{s_0'}$.

But by the assumption that $H_0 : \Sigma_0 \rightarrow \Sigma_0$ is surjective, there is a sent $s_0$ in $\Sigma_0$ such that $s_0' = H(s_0)$ and we have:

$$M_1'|_{H(s_0)} = M_1|_{s_0} = M_2|_{s_0} = M_2'|_{H(s_0)}$$

as desired. $\Box$

Of course, since we have just shown that $H_1UH_2 : T_1UT_2 \rightarrow T'_1UT'_2$

is expansive under the conditions in the lemma, we have trivial ascent maps $(H_1UH_2, H_1UH_2) : (T_1UT_2, K) \rightarrow (T'_1UT'_2, K')$ for any $K \subseteq FO(\text{sig}(T_1UT_2))$ such that $(H_1UH_2)(K) \subseteq K'$. What is the point of restricting to $F_1 \wedge F_2$? The point is, of course, that we want to use ascent maps for satisfiability purposes.

Suppose, then, that we have decision procedures for $T'_i$-satisfiability of $G_i$ formulas, $1 \leq i \leq 2$. Then, under reasonable assumption on $T'_1$ and $T'_2$, the Nelson-Oppen combination procedure (more on this in upcoming lectures) will give us
a decision procedure on formulas of the form $H_1(q_1) \land H_2(q_2)$
by invoking the decision procedure for $T_1'$ with $H_1(q_1)$, and that of $T_2'$ with $H_2(q_2)$.

3. Useful Formula Transformations as Generalized Ascent Maps

Regarding the just-discussed notion of parallel composition

$$(H_1 \lor H_2, H_1 \lor H_2) : (T_1 \lor T_2, F_1 \lor F_2) \rightarrow (T_1 \lor T_2, G_1 \lor G_2)$$

an obvious question to ask is: are the sets of formulas $F_1 \lor F_2$ and $G_1 \lor G_2$ too particular (and therefore the parallel composition not so useful)? The answer is that it is not, provided we see it within the broader ecosystem of other ascent maps performing various useful formula transformations.

Before discussing several of these transformations, let us generalize the notion of an ascent map $(H, \alpha)$ so that $\alpha$ is generalized from a recursive function to a recursive relation:

**Definition.** An ascent map $(H, \alpha) : (T, F) \rightarrow (T', G)$

where $H : T \rightarrow T'$ is an interpretation and $F \subseteq \mathcal{P}(\mathcal{O}(g(T)))$,

$G \subseteq \mathcal{P}(\mathcal{O}(g(T')))$. Recursive sets of formulas has a second component $\alpha$ a recursive function

$$\alpha : F \rightarrow \mathcal{O}(g(F))$$

where $\mathcal{O}(g(F))$ denotes the recursive data type of non-empty finite sets of formulas in $F$. Furthermore, the following formula equi-satisfiability property is satisfied:
\((\forall \varphi \in \mathcal{F}) \ \varphi \text{ is } T\text{-satisfiable } \iff \exists \varphi' \in \mathcal{F} \text{ s.t. } \varphi' \text{ is } T\text{-satisfiable.}\)

\((H, \alpha)\) is called effective if \(\exists \varphi' \in \mathcal{F} \text{ s.t. } M' \models (T')\),

and \(\alpha \in [\text{from}(\varphi) \rightarrow M']\). There is an effective construction of

a model \(M \in \text{mod}(T)\) and \(\alpha \in [\text{from}(\varphi) \rightarrow M]\) s.t., \(M, \alpha \models \varphi'.\)

Note that we recover the more restrictive notion of ascent map in Lecture 23 by viewing a recursive function \(\alpha : \mathcal{F} \rightarrow \mathcal{G}\) as a recursive function \(\hat{\alpha} : \mathcal{F} \ni \varphi \mapsto \text{def}(\varphi) \in \mathcal{P}_{\text{fin}T}(\mathcal{G}).\)

Note also that given ascent maps

\(T, F \xrightarrow{(H, \alpha)} T', G \xrightarrow{(G, \beta)} (T', K)\)

their composition \((H, \alpha); (G, \beta)\) is defined as the ascent map

\((H; G), \hat{\alpha} : \mathcal{P}_{\text{fin}T}(\beta)), \text{ where}\)

\(\mathcal{P}_{\text{fin}T}(\beta) : \mathcal{P}_{\text{fin}T}(\mathcal{G}) \ni \psi_1, \ldots, \psi_n \mapsto \beta(\psi_1) \cup \ldots \cup \beta(\psi_n) \subseteq \mathcal{P}_{\text{fin}T}(K).\)

From now on, \underline{Ascent} will denote the category of ascent maps in the more general sense just defined.

Here are some useful ascent maps, all of which are of the form:

\((1_\Sigma, \alpha) : (T, F) \rightarrow (T, G).\) That is, \(T\) does not change,

and \(1_\Sigma\) is the identity theory inclusion \(T \subseteq T\) viewed as the identity signature map \(1_\Sigma\), where \(\Sigma = \text{sig}(T).\)

Therefore, all the action in such maps resides in \(\alpha\), which is some kind of \underline{satisfiability-preserving formula}.\)
transformation. Here are several useful examples very heavily used in SMT solving. For some of them I leave the proof that they are effective ascent maps as an exercise.

1. \((1^\Sigma, \text{dnt}): (T, \text{QF}(\Sigma)) \to (T, \text{DNF}(\Sigma)), \) where \(\Sigma = \text{Sig}(T),\)

\(\text{DNF}(\Sigma) \subseteq \text{QF}(\Sigma)\) are the formulas in disjunctive normal form, and \(\text{dnt}(\phi)\) is the disjunctive normal form of \(\phi\), which can be determined either by using \(\land\) and \(\lor\) associative-commutative or by some total order on formulas.

2. \((1^\Sigma, \text{cnj}^\phi): (T, \text{DNF}(\Sigma)) \to (T, \text{ALit}(\Sigma)), \) where \(\text{ALit}(\Sigma)\) are the conjunction of literals of \(\Sigma\), and \(\text{cnj}^\phi: \text{DNF}(\Sigma) \to \bigcap_{\text{all } \phi} \text{ALit}(\Sigma)\)

maps each \(\phi = \bigvee_{i} C_i\) to \(\{C_1, \ldots, C_n\}\).

3. \((1^\Sigma, \text{val}^{\phi}) : (T, \text{ALit}(\Sigma)) \to (T, \text{SALit}(\Sigma)), \) where \(\text{SALit}(\Sigma) \subseteq \text{ALit}(\Sigma)\) are called simple conjunctions, and \(\text{val}^{\phi}\) is defined by

\[\text{val}^{\phi}(\phi) = \begin{cases} T & \text{if } \phi \in \text{DNF}(\Sigma) \\ \bigwedge \phi & \text{if } \phi \in \text{ALit}(\Sigma) \end{cases}\]

The mapping \(\text{val}^{\phi}\) is called variable substitution and is defined by the following rewrite rules, rewriting conjunctions where \(\land\) is treated as an associative-commutative symbol with \(T\) as its identity element, and where the symbols \(-\) and \(\neq\) are considered commutative (so \(u = v\) in the same equation as \(v = u\)), so this is ACU + C - rewriting:
\( f(x_1, \ldots, x_n) = g(y_1, y_n) \land C \rightarrow x : i = f(x_1, \ldots, x_n) \land x : i = g(y_1, y_n) \land (C) \upharpoonright \{x_i \rightarrow x : i\} \)

(1) \( f(u_1, \ldots, u_i, \ldots, u_n) = v \land C \rightarrow x : i = u_i \land (f(u_1, \ldots, x : i, \ldots, u_n) = v \land C) \upharpoonright \{u_i \rightarrow x : i\} \)

(2) \( f(u_1, \ldots, u_i, \ldots, u_n) \neq v \land C \rightarrow x : i = u_i \land (f(u_1, \ldots, x : i, \ldots, u_n) = v \land C) \upharpoonright \{u_i \rightarrow x : i\} \)

(3) \( f(u_1, \ldots, u_i, \ldots, u_n) \land C \rightarrow x : i = u_i \land (f(u_1, \ldots, x : i, \ldots, u_n) \land C) \upharpoonright \{u_i \rightarrow x : i\} \)

(4) \( f(u_1, \ldots, u_i, \ldots, u_n) \land C \rightarrow x : i = u_i \land (f(u_1, \ldots, x : i, \ldots, u_n) \land C) \upharpoonright \{u_i \rightarrow x : i\} \)

\[ \text{where } u_i \notin X \text{ is a non-variable subterm, } S = \text{ls}(u_i) \text{ is the least sort of } u_i, \]

\( x : i \) is a fresh variable not appearing anywhere in the conjunction, and \( (C') \upharpoonright \{u_i \rightarrow x : i\} \)

denote the canonical form obtained by exhaustively rewriting a conjunction \( C' \) by replacing each occurrence of the term \( u_i \) by the variable \( x : i \) (note that this happens at the metalevel, so that \( u_i \rightarrow x : i \) is a ground rewrite rule). The process \( (C') \upharpoonright \{u_i \rightarrow x : i\} \) is not strictly required for correctness, but can greatly improve efficiency by yielding formulas with considerably fewer fresh variables.

That this is an actual way can be shown by proving that:

(a) each step of rewriteup (before the \( \{u_i \rightarrow x : i\} \) optimization)

with rules (0)-(4), and

(b) each step of rewriteup with the ground rewrite rule \( u_i \rightarrow x : i \)

the right-hand side of rules (0)-(4)

preserves \( T \)-satisfiability and is effective. To illustrate the general idea, let us focus on the rule

\( f(u_1, \ldots, u_i, \ldots, u_n) = v \land C \rightarrow x : i = f(u_1, x : i, \ldots, u_n) \neq v \land C \)
Let \( Y = \text{vars} (f(u_1, \ldots, u_n) \neq v \land C) \), and let \( M = (M, \psi) \in \text{mod}(T) \), and \( a \in [Y \to M] \) be such that \( M, a \models f(u_1, \ldots, u_n) \neq v \land C \), i.e., \( M, a \models C \) and if \( M, a \models f(u_1, \ldots, u_n) \neq v \land C \),

\( \tilde{a} = a \cup \{(x:3, u_1: a_1)\} \), then \( M, \tilde{a} \models x:3 = u_1 \land f(u_1, x:3, \ldots, u_n) \).

Conversely, if for some \( \tilde{a} \in [Y \cup \{x:3\}] \) \( \bullet \) holds,

then defining \( a = \tilde{a} \setminus \{(x:3, \tilde{a}(x:3))\} \) we get

\( M, a \models f(u_1, \ldots, u_n) = v \), and, of course, \( M, \tilde{a} \models C \).

Therefore \( M, a \models C \).

4. Let \( \text{sig}(T) = \Sigma_1 \cup \Sigma_2 \), with \( \Sigma_1, \Sigma_2 \) nicely intersectable and

\( \Sigma_0 = \Sigma_1 \cap \Sigma_2 \) consisting only of sorts (unshared function or

predicate symbols), let

\[ (1, 2_{\Sigma_1, \Sigma_2}^{\text{pur}}) : (T, \text{QF}(\Sigma_1 \cup \Sigma_2)) \to (T, \text{Lit}(\Sigma_1) \land \text{Lit}(\Sigma_2)) \]

where \( \text{Lit}(\Sigma_1) \land \text{Lit}(\Sigma_2) = \{ C_i \land C_j \mid C_i \in \text{Lit}(\Sigma_1), C_j \in \text{Lit}(\Sigma_2), 1 \leq i \leq 2 \} \),

and \( \text{pur} \) is the formula purification transformation defined by first applying exhaustively the rules:

1. \( u[v]_p = w \land C \to (u[x:3] = w \land C)_! \{v \mapsto x:3\} \)

2. \( u[v]_p = w \land C \to (x:3 = v \land u[x:3] \neq w \land C)_! \{v \mapsto x:3\} \)

3. \( f(u_1, \ldots, u_n) \land C \to (x:3 = u_1 \land f(u_1, x:3, \ldots, u_n) \land C)_! \{u_i \mapsto x:3\} \)

4. \( \top \land C \to (x:3 = u_1 \land \top \land (u_1, x:3, \ldots, u_n) \land C)_! \{u_i \mapsto x:3\} \)

if \( u_i \notin X \)

5. \( f(u_1, \ldots, u_n) \land C \to (x:3 = u_1 \land f(u_1, x:3, \ldots, u_n) \land C)_! \{u_i \mapsto x:3\} \)
where in rules (1)-(2) \( p = q \cdot i \in \text{pos}_2(u) \) is such that for each decomposition \( q = q_1 \cdot q_2 \), including \( q_1 = \varepsilon \) and \( q_2 = q \), \( \text{top}(u \mid q_2) \in \text{fun}(\Sigma_1) \) (resp. \( \text{fun}(\Sigma_2) \)), and \( \text{top}(u \mid p) \in \text{fun}(\Sigma_2) \) (resp. \( \text{fun}(\Sigma_1) \)), where \( \text{top}(t) \) returns the top function symbol of a non-variable term, and where if \( \llbracket ls(u \mid p) \rrbracket_1 = \llbracket ls(u \mid p) \rrbracket_2 \) the sort \( s \) of the fresh variable \( x : s \) is just \( ls(u \mid p) \), and otherwise in the unique sort \( s \) such that \( [s_1] \cap \llbracket ls(u \mid p) \rrbracket_2 = \{ s \} \) for some \( [s_1] \in S_1 / \equiv_{x_1} \) (resp. \( [s_2] \cap \llbracket ls(u \mid p) \rrbracket_1 = \{ s \} \) for some \( [s_2] \in S_2 / \equiv_{x_2} \)); and for rules (3) - (4), if \( p = x_{i_1} \cdots x_{i_n} \) in \( \Sigma_j \) \( 1 \leq j \leq 2 \) and \( \text{top}(u_i) \in \text{fun}(\Sigma_j) \) then \( s = \llbracket ls(u_i) \rrbracket_j \). Otherwise, if \( \text{top}(u_i) \in \text{fun}(\Sigma_j) \) with \( \{ j, j' \} = \{ 1, 2 \} \), if \( [s_i]_1 = [s_i]_2 \) we choose again \( s = \llbracket ls(u_i) \rrbracket_i \) in the fresh variable \( x : s \). Finally, if \( \{ s_0 \} = [s_i]_j \cap [s_i]_j \) with \( \{ j, j' \} = \{ 1, 2 \} \), then we choose \( s = s_0 \) in the fresh variable \( x : s \).

After a canonical form has been reached with rules (1)-(4) we perform a second process of rewriting to canonical form with the rule:

\[ u = \exists x : s \left[ C \mid \text{if } x \in \mathcal{T}_2(x) - X \text{ and } \forall x \in X \right. \]

\[ \text{if } x = \exists y \text{ then } \exists y : s \text{ if } s \subseteq \text{sub}(y) \]
where is (5) as follows:

(i) \( [l_3(u^*)]_2 = [l_3(u)]_2 \),

we choose \( \alpha = l_3(u^*) \),

and (ii) otherwise, \( \{s_0\} = [l_3(u)]_2 \land [l_3(u^*)]_2 \),

and we choose \( \alpha = s_0 \). Note that (ii) is conditioned on \( \alpha = l_3(u) \).

The proof that \((1_\Sigma, \text{pur})\) is an effective ascent map is quite similar to that for \((1_\Sigma, \text{vals})\) and is left as an exercise.

We can now come back to the question of whether the definition of \((H_1 \lor H_2, H_2 \lor H_2) : (T_1 \lor T_2, F \land F) \rightarrow (T_1' \lor T_2', G \land G)\)

was or not too restrictive. It is not in the following ecosystem of ascent maps:

\[
\begin{align*}
(T_1 \lor T_2, \text{QF}(\Sigma_1 \lor \Sigma_2)) & \rightarrow (T_1 \lor T_2, \text{DNF}(\Sigma_1 \lor \Sigma_2)) & (1_\Sigma, \text{val}), \text{dnf} \\
(T_1 \lor T_2, \text{DNF}(\Sigma_1 \lor \Sigma_2)) & \rightarrow (T_1 \lor T_2, \text{LA}(\Sigma_1 \lor \Sigma_2)) & (1_\Sigma, \text{pur})
\end{align*}
\]

where, recall that \(H_1, H_2\) must be expansive for the whole composition to be an effective ascent map.

4. Descent Maps

Intuitively, an ascent map \((H, \alpha) : (T, F) \rightarrow (T', G)\) is used to reduce the \(T\)-satisfiability of \(F\)-formulas to that of the \(T'\)-satisfiability of \(G\)-formulas for which we hopefully have (or can get by further composition with
additional ascent maps) a descent procedure. We do so by "climbing up" using $H : T \rightarrow T'$ from an, in principle simpler, theory $T$ to an, in principle, more complex theory $T'$. I say, "in principle", because in the case of $(\Sigma, \mu) : (\Sigma, \phi) QF(\Sigma) \rightarrow (\Sigma, \mu) QF(\Sigma')$ used to reduce order-sorted congruence closure for $(\Sigma, \phi)$ (or $(\Sigma, \alpha_{\Delta})$) to unsorted congruence closure for $(\Sigma', \phi)$ (or $(\Sigma', \alpha_{\Delta'})$) the theory $(\Sigma', \phi)$ is actually simpler!

But there are many other cases (we have seen many in the VanSant paper) where what we want to do is not to climb up from $T$ to $T'$ but to descend from $T'$ to $T$, for example when $T' \supset T$ is an extension. Therefore, there is also a very useful notion of descent map as follows:

**Definition:** Let $H : T \rightarrow T'$ be a theory interpretation, and let $F \subseteq \text{FO}(\text{sig}(T))$, $G \subseteq \text{FO}(\text{sig}(T'))$ be recursive sets of formulas. A descent map from $(T', G)$ to $(T, F)$ is then a pair $(H, \delta)$, written $(H, \delta) : (T', G) \rightarrow (T, F)$ where $\delta$ is a recursive function $\delta : F \rightarrow \bigcup_{\phi \in \text{sig}(G)} G_{\phi}$ such that for each $\psi \in G$, $\psi$ is $T'$-satisfiable iff $\psi$ is $T$-satisfiable for some $\psi \in \delta(\psi)$. $(H, \delta)$ is called effective if given $\psi \in G$, $\psi \in \delta(\psi)$, $M \in \text{mod}(T)$, and $\bar{x} \in \text{Fvar}(\psi)$ there is an effective construction of an $M' \in \text{mod}(T')$, $x \in M'$ s.t. $M', \bar{x} \models \psi$. 
