Theorem 8.3.3 (SUP(T) Completeness)**.** Let $\mathcal{H} = (\mathcal{T}^H, \mathcal{T}^B)$ be sufficiently complete and $\mathcal{T}^B$ be compact and term-generated. Then N is unsatisfiable with respect to hierarchic algebras of $\mathcal{H}$ iff $N \Rightarrow^*_{\text{SUPT}} N' \cup \{\bot\}$.

ToDo: Proof

The concrete redundancy notions from superposition, Section 5.2, carry over to the extension with theories, however, implication of the respective constraints has to be checked in addition.

Subsumption $(N \uplus \{\Lambda_1 \parallel C_1, \Lambda_2 \parallel C_2\}) \Rightarrow_{\text{SUPT}} (N \cup \{\Lambda_1 \parallel C_1\})$
provided $C_1\sigma \subset C_2$ for some matcher σ and $\models_B \forall\vec{x}.\exists\vec{y}.(\Lambda_2 \rightarrow \Lambda_1\sigma$ where $\vec{x} = \text{vars}(\Lambda_2 \parallel C_2)$ and $\vec{y} = \text{vars}((\Lambda_1 \parallel C_1)\sigma)) \setminus \text{vars}(\Lambda_2 \parallel C_2)$

8.4 SUP(T) Decides the Ground Case

If the clause set N, not yet abstracted, of a hierarchic specification $\mathcal{H}$ is ground, then an instance of SUP(T) decides unsatisfiability of N, provided $\mathcal{T}^B$ enables a decision procedure for the applicability of Constraint Refutation. An immediate application of SUP(T) to a ground clause set N does not yield a decision procedure, because N may not be sufficiently complete and hence SUP(T) may not be complete, and, SUP(T) does not necessarily terminate on N without further refinements. So this section is mainly about solving these two issues.

Sufficient comepleteness, Definition 8.3.1, requires equality of any unpure background theory sort ground term t to a pure background theory ground term t'. There are only finitely many ground terms in a ground clause set N. If N can be "completed", i.e., extended by further defining unit equations, such that N becomes sufficiently complete for all its ground terms and if then SUP(T) does not derive any new unpure ground terms of a background theory sort, this guarantees sufficient completeness for all ground terms occurring in a SUP(T) derivation of N. The below rule transforms N into a sufficiently complete clause set by the introduction of new parameters, preserving satisfiability.

Sufficient Completion $N[f(t_1, \ldots, t_n)]_{p_1,\ldots,p_n} \Rightarrow_{\text{SC}} N[b]_{p_1,\ldots,p_n} \cup \{f(t_1, \ldots, t_n) \approx b\}$
provided f is a Σ^F function symbol ranging into a background theory sort, no t_i contains a Σ^F function symbol ranging into a background theory sort, and b is a fresh parameter from Σ^B

So $\Rightarrow_{\mathrm{SC}}$ replaces unpure ground terms with parameters in a bottom-up way. For example, the ground clause

$$\neg P(f(h(1+g(a)))) \vee f(h(1)) + g(a) \geq 0 \vee P(g(a)) \vee Q(g(a))$$

is replaced by the clauses

$$\begin{array}{l} \neg P(b_3) \vee b_2 + b_1 \geq 0 \vee P(b_1) \vee Q(b_1) \\ g(a) \approx b_1 \\ f(h(1)) \approx b_2 \\ f(h(1+b_1)) \approx b_3 \end{array}$$

where $\mathcal{T}^B$ is LRA, a is not of sort LA, h does not range into LA, and the b_i are fresh parameters from LRA.

Next the clauses are abstracted resulting in the clause set

$$\begin{array}{l} y = b_1, x = b_3, b_2 + b_1 < 0 \parallel \neg P(x) \vee P(y) \vee Q(y) \\ y = b_1 \parallel g(a) \approx y \\ z = b_2, w = 1 \parallel f(h(w)) \approx z \\ u = 1 + b_1, v = b_3 \parallel f(h(u)) \approx v \end{array}$$

where now all introduced variables are equal in the constraint to a background theory ground term. The resulting clauses now have the property that all variables are variables of a background theory sort and that for all background variables $x \in C$ of some clause $\Lambda \parallel C$ there is a an atom $x = t \in \Lambda$ where t is a ground base term. In addtion, the only variable occurrences in Λ are equations $x = t$ for some ground term t.

The additional parameters b_i destroy compactness of LRA, see the introduction, Section 8.1. However, compactness is not needed here, because I will eventually show termination of superposition on completed and abstracted clause sets. In order to show termination, clauses must not become arbitrarily long and terms must not become aribtrarily deep in the generated clauses. The length of clauses can always be limited if variable chains in clauses are prevented. For example, the transitivity clause $\neg R(x,y) \vee \neg R(y,z) \vee R(x,z)$ constiutes such a variable chain preventing the clause to be split into variable disjoint parts.

The next step is to prevent variable chains. If in some clause $\Lambda \parallel C$ a variable x occurs several times in C, then in the context of completed and abstracted ground clauses a fresh variables y is introduced and the assignment $x = t \in \Lambda$ is copied for y and added to Λ. For the first clause of the running example the result is

$$z = b_1, y = b_1, x = b_3, b_2 + b_1 < 0 \parallel \neg P(x) \vee P(y) \vee Q(z).$$

Now this clause can actually be split into three variable disjoint parts

$$\begin{array}{l} x = b_3, b_2 + b_1 < 0 \parallel \neg P(x) \\ y = b_1, b_2 + b_1 < 0 \parallel P(y) \\ z = b_1, b_2 + b_1 < 0 \parallel Q(z) \end{array}$$

where the inital clause set containing $z = b_1, y = b_1, x = b_3, b_2 + b_1 < 0 \parallel \neg P(x) \vee P(y) \vee Q(z)$ is unsatisfiable iff the three clause sets obtained by replacing the clause with one of the three split clauses $x = b_3, b_2 + b_1 < 0 \parallel \neg P(x)$, $y = b_1, b_2 + b_1 < 0 \parallel P(y)$, $z = b_1, b_2 + b_1 < 0 \parallel Q(z)$, respectively, is unsatisfiable. So for the overall inference process it can be assumed that every clause contains at most one background theory variable.

Proposition 8.4.1 (Clause Variations)**.** Let M be a finite set of ground literals of the background theory and $k \in \mathbb{N}$ fixed. Then there are only finitely many non-redundat clauses $\Lambda \parallel C$ where

1. $\mathrm{vars}(\Lambda \parallel C) = \{x\}$ for some variable x
2. $\Lambda = \{x = t\} \uplus \Lambda'$ where $\Lambda' \subseteq M$
3. $|L| \leq k$ for all $L \in C$.

Practically, the number of clause variations is kept finite in a SUP(T) run by exhaustive application of subsumption.

So far I have explained how the conditions 8.4.1.1 and 8.4.1.2 can be obtained and preserved during derivations of the SUP(T) calculus. It remains to guarantee 8.4.1.3 via a suitable reduction ordering. Note that any reduction ordering is total on the clauses considered here, because every variable is from a background theory sort and mapped to a background theory ground term in the constraint. So in order to compare literals in a clause $\{x = t\} \uplus \Lambda' \parallel C[x]$ it is sufficient to consider the simple ground instance $(\{x = t\} \uplus \Lambda' \parallel C[x])\{x \mapsto t\}$ because all other different simple ground instances are tautologies.

Definition 8.4.2 ($\succ^F_{lpo}$)**.** For some term t let $|t|^F$ be the number of function symbols from Σ^F contained in t. Then $\succ^F_{lpo}$ is defined by $t \succ^F_{lpo} s$ iff

1. $|t|^F > |s|^F$ or
2. $|t|^F = |s|^F$ and $t \succ_{lpo} s$

Note that for any clause $\Lambda \parallel C$ considered here, all function symbols occurring in C are from Σ^F and if C contains a variable x, it is of a background theory sort, hence instantiated only by Σ^B (ground) terms via simple substitutions. Recall that any pure Σ^B ground term is by assumption strictly smaller than any ground term containing symbols from Σ^F. Therefore, $\succ^F_{lpo}$ is stable under substitutions in the current context and hence a reduction ordering total on ground terms.

There is final complication to be considered. It might happen that SUP(T) generates a variable equation in a clause $\Lambda_1, \Lambda_2 \parallel x \circ y \vee C' \vee D'$, $\circ \in \{\approx, \not\approx\}$, as a result of a superposition inference from clauses $\Lambda_1 \parallel L_1 \vee C'$ and $\Lambda_2 \parallel L_2 \vee D'$ where actually the clauses have the form $\{x = t_1\} \uplus \Lambda'_1 \parallel L_1 \vee C'$ and $\{y = t_2\} \uplus \Lambda_2 \parallel L_2 \vee D'$. In this case I simply replace the clause $\Lambda_1, \Lambda_2 \parallel x \circ y \vee C' \vee D'$ with $\Lambda_1, \Lambda_2, t_1 = t_2 \parallel C' \vee D'$.

Lemma 8.4.3 (SUP(T) Termination)**.** SUP(T) terminates on any completed and abstracted ground clause set with respect to the ordering $\succ^F_{lpo}$, exhaustive splitting, and simplifications on background theory variables and equations.

Theorem 8.4.4 (Decision Procedure)**.** SUP(T) is a decision procedure for a hierarchic specification $\mathcal{H} = (\mathcal{T}^H, \mathcal{T}^B)$ where N is ground and $\mathcal{T}^B$ provides a decision procedure for ground formulas with parameters.