5 Completeness of Saturation

In the previous chapter we stated some calculi that have been used for the implementation of the automated theorem provers we listed at the end of the chapter. We have seen in examples that in some cases the set of clauses saturates, i.e. the procedure cannot generate any new clause from the set of clauses using the procedure-specific calculus. A procedure is complete when it does not saturate for unsatisfiable set of clause without proofing unsatisfiability, e.g. by demonstrating unsatisfiability with an expensive transformation of the set of clauses into a disjunctive normal form.

It was easy to see that Gilmore's prover is theoretically complete, but practically very inefficient. Additionally it does not stop for clearly satisfiable and simple sets of clauses. It seems intuitive that resolution is complete since we search for all clashing literals between clauses and resolve them. But it is puzzling that ordered resolution and even superposition are indeed complete. At first sight we could expect that all these restrictions would ignore important inferences at least in some special cases.

In general a smaller set of sound rules and stricter conditions for a rule's application can improve the efficiency of a procedure, but undermine the completeness of the procedure. An empty calculus, i.e. a calculus without any rules at all, is obviously sound and very efficient, but of course not complete.

In other words we have to prove for every saturation-based calculus that we can decide satisfiability of any saturated set of clauses. That's even true for Gilmore's prover, but in most cases, i.e. where the signature contains at least one non-constant predicate symbol (or the equality symbol) and one non-constant function symbol, sets of clauses will not saturate.

In the first Section 5.1 we state unit paramodulation for literal closures. In Section 5.2 on page 42 we present a less nested and more sequential version of the proof of satisfiability of saturated sets of closures from the literature [9]. After that we look at saturation strategies in Section 5.3 on page 46 and finally we apply the completeness result to `Inst-Gen-Eq` in Section 5.5 on page 48.

5.1 Equational Reasoning on Equation Closures

Before we can approach the completeness proof we have to define our terminology.

Definition 5.1. A closure is a pair of a clause C and a substitution σ, conveniently written as $C \cdot \sigma$, where $\mathrm{dom}(\sigma) \subseteq \mathcal{V}ars(C)$. Two closures $C \cdot \sigma = D \cdot \tau$ are the same if C is a variant of D and $C\sigma$ is a variant of $D\tau$. A closure $C \cdot \sigma$ represents a clause $C\sigma$, i.e. the result of applying substitution σ to C. A ground closure represents a ground clause.

For the following definitions we assume $\succ_{\mathtt{gr}}$ as a a total, well-founded and monotone extension from a total simplification ordering on ground terms to ground literals and clauses [23], which always exists.

Definition 5.2. We define an order $\succ_{\mathtt{L}}$ on ground closures of literals as an arbitrary total well-founded extension of $\succ_{\mathtt{gr}}$ such that $L \cdot \sigma \succ_{\mathtt{L}} L' \cdot \sigma'$ whenever $L\sigma \succ_{\mathtt{gr}} L'\sigma'$. We define an order $\succ_{\mathtt{C}}$ on ground closures as an arbitrary total well-founded extension of $\succ_{\mathtt{gr}}$ such that $C \cdot \tau \succ_{\mathtt{C}} D \cdot \rho$ whenever $C\tau \succ_{\mathtt{gr}} D\rho$ or $C\theta = D$ for a proper instantiator θ.

In summary we assume $\succ_{\mathtt{gr}}$, $\succ_{\mathtt{L}}$, $\succ_{\mathtt{C}}$ as total, well-founded, and monotone extensions of a total simplification order $\succ$ over ground terms to ground (literal) clauses and ground (literal) closures such that the following properties hold

$$s\sigma \not\approx t\sigma \quad \succ_{\mathtt{gr}} \quad s\sigma \approx t\sigma \qquad\qquad L\sigma \succ_{\mathtt{gr}} L'\sigma' \quad \Rightarrow \quad L \cdot \sigma \succ_{\mathtt{L}} L' \cdot \sigma'$$

$$L\sigma \vee C\sigma \quad \succ_{\mathtt{gr}} \quad L\sigma \qquad\qquad \left.\begin{array}{r}(C\tau = D\rho \text{ and } C\theta = D)\\ \text{or } C\tau \succ_{\mathtt{gr}} D\rho\end{array}\right\} \quad \Rightarrow \quad C \cdot \tau \succ_{\mathtt{C}} D \cdot \rho$$

for arbitrary terms s, t, literals L, L', clauses C, D, ground substitutions σ, τ, ρ, and proper instantiator θ. Note that the order on unit closures is slightly different than on literal closures, e.g.:

$$(\mathsf{f}(x) \approx x) \cdot \{x \mapsto \mathsf{a}\} \succ_{\mathtt{C}} (\mathsf{f}(\mathsf{a}) \approx \mathsf{a}) \cdot \emptyset \qquad\qquad \theta = \{x \mapsto \mathsf{a}\}$$
$$(\mathsf{f}(x) \approx x) \cdot \{x \mapsto \mathsf{a}\} \not\succ_{\mathtt{L}} (\mathsf{f}(\mathsf{a}) \approx \mathsf{a}) \cdot \emptyset$$

In the following definition we write $\underset{\approx}{L}[l']$ for an arbitrary equational literal, i.e. either $s[l'] \not\approx t$ or $s[l'] \approx t$ for some first order terms s and t.

Definition 5.3 (Unit superposition and resolution on literal closures [29], UPC)**.**

$$\frac{(l \approx r) \cdot \theta_1 \quad (\underset{\approx}{L}[l']) \cdot \theta_2}{(\underset{\approx}{L}[r])\sigma \cdot \rho}\ (\sigma) \qquad\qquad \frac{(s' \not\approx t') \cdot \theta'}{\square}\ (\sigma')$$

whenever

- $\sigma = \mathrm{mgu}(l, l')$, $l' \notin \mathcal{V}$, $l\theta_1 \succ_{\mathtt{gr}} r\theta_1$, $s[l']\theta_2 \succ_{\mathtt{gr}} t\theta_2$ $\qquad\qquad \sigma' = \mathrm{mgu}(s', t')$
- $l\theta_1 = l'\theta_2$, $\mathit{Vars}(\{l, r\}) \cap \mathit{Vars}(\{s[l'], t\}) = \emptyset$ $\qquad\qquad s'\theta' = r'\theta'$
- there exists ρ such that $(\theta_1 \cup \theta_2) = \sigma\rho$, $\mathrm{dom}(\rho) \subseteq \mathit{Vars}(\{s[r], t\})$

Remark. The corresponding inference rule in an earlier paper [9] omits $s[l']\theta_2 \succ_{\mathtt{gr}} t\theta_2$ and is sound. We expect a calculus that is based on the inference rules above to draw less inferences with this restriction in place than without. However, it still has to yield a complete procedure with some additional considerations.

In the following example we demonstrate an entire unit superposition step where all necessary conditions are checked and fulfilled.

Example 5.4. Let $\mathsf{f}(s,t,u) \succ_{\mathtt{gr}} \mathsf{g}(s)$ for all ground terms r, s, t.

$$\frac{\left(\overbrace{\mathsf{f}(x,y,\mathsf{c})}^{l} \approx \overbrace{\mathsf{g}(x)}^{r}\right)\cdot\theta_1 \qquad \left(\mathsf{h}(\overbrace{\boxed{\mathsf{f}(x',\mathsf{h}(y'),z')}}^{l'}) \not\approx \mathsf{g}(x')\right)\cdot\theta_2}{\left(\mathsf{h}(\underbrace{\boxed{\mathsf{g}(x')}}_{r\sigma}) \not\approx \mathsf{g}(x')\right)\cdot\rho}\ (\sigma)$$

$$\theta_1 = \{x \mapsto \mathsf{a}, y \mapsto \mathsf{h}(\mathsf{b})\} \qquad \theta_2 = \{x' \mapsto \mathsf{a}, y' \mapsto \mathsf{b}, z' \mapsto \mathsf{c}\}$$
$$\sigma = \{x \mapsto x', y \mapsto \mathsf{h}(y'), z' \mapsto \mathsf{c}\} \qquad \rho = \{x' \mapsto \mathsf{a}, y' \mapsto \mathsf{b}\}$$

$$\begin{aligned}\sigma\rho &= \{x \mapsto x'\rho, y \mapsto \mathsf{h}(y')\rho, z' \mapsto \mathsf{c}, x' \mapsto \mathsf{a}, y' \mapsto \mathsf{b}\}\\ &= \{x \mapsto \mathsf{a}, y \mapsto \mathsf{h}(\mathsf{b}), z' \mapsto \mathsf{c}, x' \mapsto \mathsf{a}, y' \mapsto \mathsf{b}\} = \theta_1 \cup \theta_2\end{aligned}$$

Definition 5.5. A substitution σ is irreducible w.r.t. TRS R if $\mathrm{img}(\sigma) \subseteq \mathsf{NF}(\mathcal{R})$, i.e. all terms $\sigma(x)$ with $x \in \mathrm{dom}(\sigma)$ are irreducible (normal forms) w.r.t. TRS R.

Lemma 5.6. *Let R be a ground rewrite system and suppose USC is applicable to $(l \approx r)\cdot\theta_1, L[l']\cdot\theta_2$ with conclusion $L[r]\sigma\cdot\rho$. If θ_1, θ_2 are irreducible w.r.t. R then ρ is irreducible w.r.t. R.*

Note that we can easily demonstrate that UPC on its own is not complete on sets of unit closures, i.e. we might or might not be able to derive the empty clause from inconsistent set of literal closures. First we look at a working example then at an "equivalent" counterexample.

Example 5.7. With $\mathsf{a} \succ_{\mathtt{gr}} \mathsf{b}$ we can derive the empty clause from the set of unit closures $\{(\mathsf{f}(\mathsf{a}) \approx \mathsf{b})\cdot\emptyset, \mathsf{a} \approx \mathsf{b}\cdot\emptyset, \mathsf{f}(\mathsf{b}) \not\approx \mathsf{b}\cdot\emptyset\}$ in two steps. We start with a superposition step and finish with a resolution step, as follows.

$$\dfrac{\dfrac{\mathsf{a} \approx \mathsf{b}\cdot\emptyset \quad \mathsf{f}(\mathsf{a}) \approx \mathsf{b}\cdot\emptyset}{\mathsf{f}(\mathsf{b}) \approx \mathsf{b}\cdot\emptyset}\,\emptyset \qquad \mathsf{f}(\mathsf{b}) \not\approx \mathsf{b}\cdot\emptyset}{\square}\,\emptyset$$

Example 5.8. With $\mathsf{a} \succ_{\mathtt{gr}} \mathsf{b}$ we cannot derive any unit closure from the set of unit closures $\{\mathsf{f}(x) \approx \mathsf{b}\cdot\{x \mapsto \mathsf{a}\}, \mathsf{a} \approx \mathsf{b}\cdot\emptyset, \mathsf{f}(\mathsf{b}) \not\approx \mathsf{b}\cdot\emptyset\}$. Neither the resolution rule nor the superposition rule are applicable to any of the unit closures.

$$\dfrac{\mathsf{a} \approx \mathsf{b}\cdot\emptyset \quad \mathsf{f}(x) \approx \mathsf{b}\cdot\{x \mapsto \mathsf{a}\}}{\varepsilon}\,\emptyset \qquad x \in \mathcal{V}$$

$$\dfrac{\mathsf{f}(x) \approx \mathsf{b}\cdot\{x \mapsto \mathsf{a}\} \quad \mathsf{f}(\mathsf{b}) \not\approx \mathsf{b}}{\varepsilon}\,\{x \mapsto \mathsf{b}\} \qquad \mathsf{f}(x)\{x \mapsto \mathsf{a}\} \neq \mathsf{f}(\mathsf{b})\emptyset$$

$$\dfrac{\mathsf{a} \approx \mathsf{b}\cdot\emptyset \quad \mathsf{f}(\mathsf{b}) \not\approx \mathsf{b}}{\varepsilon} \qquad \mathrm{mgu}(\mathsf{a}, \ell') \text{ does not exist}$$

$$\dfrac{\mathsf{f}(x) \approx \mathsf{b}\cdot\{x \mapsto \mathsf{a}\} \quad \mathsf{a} \approx \mathsf{b}\cdot\emptyset}{\varepsilon} \qquad \mathrm{mgu}(\mathsf{f}(x), \ell') \text{ does not exist}$$

5.2 Satisfiability of Saturated Sets

In this section we will define saturated sets of literal closures with respect to UPC. Additionally we will show that a saturated set of literal closures is satisfiable if and only if its grounding is satisfiable.

Definition 5.9 (UP-Redundancy)**.** Let $\mathcal{L}$ be a set of literal closures. We define

- $\text{irred}_R(\mathcal{L}) = \{\, L \cdot \sigma \in \mathcal{L} \mid \sigma \text{ is irreducible w.r.t. } R \,\}$
  for an arbitrary ground rewrite system R
- $\mathcal{L}_{L\cdot\sigma\succ_{\mathtt{L}}} = \{\, L' \cdot \sigma' \in \mathcal{L} \mid L \cdot \sigma \succ_{\mathtt{L}} L' \cdot \sigma' \,\}$.
- Literal closure $L \cdot \sigma$ is *UP-redundant* in $\mathcal{L}$ if

$$R \cup \text{irred}_R(\mathcal{L}_{L\cdot\sigma\succ_{\mathtt{L}}}) \vDash L\sigma$$

  for every ground rewrite system R oriented by $\succ_{\mathtt{gr}}$ where σ is irreducible w.r.t. R.
- $\mathcal{R}_{UP}(\mathcal{L})$ denotes the set of all UP-redundant closures in $\mathcal{L}$.

Definition 5.10 (UP-Saturation)**.** A UP-saturation process is a sequence $\{\mathcal{L}_i\}_{i=0}^{\infty}$ where $\mathcal{L}_{i+1}$ is constructed from $\mathcal{L}_i$ by removing redundant literal closures in $\mathcal{L}_i$ or by adding inferences of $\mathcal{L}_i$ until saturation.

$$\mathcal{L}_{i+1} = \begin{cases} \mathcal{L}_i \backslash L \cdot \sigma & \text{if} \quad R \cup \text{irred}_R(\mathcal{L}_{i,L\cdot\sigma\succ_{\mathtt{L}}}) \vDash L\sigma \\ \mathcal{L}_i \cup \square & \text{if} \quad \begin{cases} (s \not\approx t) \cdot \tau \in \mathcal{L}_i \\ s\tau = t\tau,\ \mu = \text{mgu}(s,t) \end{cases} \\ \mathcal{L}_i \cup\ L[r]\theta \cdot \rho & \text{if} \quad \begin{cases} (\ell \approx r) \cdot \sigma,\ L[\ell'] \cdot \sigma' \in \mathcal{L}_i \\ \ell\sigma \succ_{\mathtt{gr}} r\sigma,\ \theta = \text{mgu}(\ell,\ell'), \\ \ell' \notin \mathcal{V},\ \ell\sigma = \ell'\sigma' = \ell'\theta\rho, \end{cases} \\ \mathcal{L}_i & \text{if and only if no other step is applicable} \end{cases}$$

The set of persistent closures $\mathcal{L}^{\infty}$ is the lower limit of $\mathcal{L}_i$.

Definition 5.11 (UP-Fairness)**.** A UP-saturation process is *UP-fair* if for every UP-inference with premises in $\mathcal{L}^{\infty}$ the conclusion is UP-redundant w.r.t. $\mathcal{L}_j$ for some j.

Definition 5.12. For a set of literals $\mathcal{L}$ we define the saturated set of literal closures $\mathcal{L}^{sat} = \mathcal{L}^{\infty} \backslash \mathcal{R}_{UP}(\mathcal{L}^{\infty})$ for some UP-saturation process $\{\mathcal{L}_i\}_{i=0}^{\infty}$ with $\mathcal{L}_0 = \mathcal{L}$.

Lemma 5.13. *[9] The set $\mathcal{L}^{sat}$ is unique because for any two UP-fair saturation processes $\{\mathcal{L}_i\}_{i=0}^{\infty}$ and $\{\mathcal{L}'_i\}_{i=0}^{\infty}$ with $\mathcal{L}_0 = \mathcal{L}'_0$ we have*

$$\mathcal{L}^{\infty} \backslash \mathcal{R}_{UP}(\mathcal{L}^{\infty}) = \mathcal{L}'^{\infty} \backslash \mathcal{R}_{UP}(\mathcal{L}'^{\infty})$$

Definition 5.14 (Inst-Redundancy)**.** Let S be a set of clauses.

- A ground closure C is *Inst-redundant* in S if for some k
  - $C'_i \in S$, $C_i = C'_i \cdot \sigma'_i$, $C \succ_{\mathtt{C}} C_i$ for $i \in 1 \dots k$
  - such that $C_1, \dots, C_k \models C$
- A (possible non-ground) clause C is called Inst-redundant in S if each ground closure $C \cdot \sigma$ is Inst-redundant in S.
- $R_{Inst}(S)$ denotes the set of all Inst-redundant clauses in S.

Example 5.15. $S = \{ \mathsf{f}(x) \approx x, \mathsf{f}(\mathsf{a}) \approx \mathsf{a}, \mathsf{f}(\mathsf{f}(x)) \approx \mathsf{f}(x)$
$R_{Inst}(S) = \{ \mathsf{f}(\mathsf{f}(x)) \approx \mathsf{f}(x) \}$

Definition 5.16 (S-Relevance)**.** Let S be a set of clauses, let $I_\bot$ be a model of $S\bot$.

- A selection function sel based on $I_\bot$ maps clauses to literals such that

$$\mathrm{sel}(C) \in C \qquad\qquad I_\bot \models \mathrm{sel}(C)\bot$$

- The set of S-relevant literal closures

$$\mathcal{L}_S = \left\{ L \cdot \sigma \mid \begin{array}{l} L \vee C \in S,\ L = \mathrm{sel}(L \vee C) \\ (L \vee C) \cdot \sigma \text{ is not Inst-redundant in } S, \end{array} \right\}$$

Definition 5.17 (Inst-Saturation)**.** Let $\mathcal{L}_S^{sat}$ denote the result of the saturation process of $\mathcal{L}_S$. Then a set of clauses S is *Inst-saturated* w.r.t. a selection function sel, if $\mathcal{L}_S^{sat}$ does not contain the empty literal clause.

Theorem 5.18. *If a set of clauses S is Inst-saturated, and $S\bot$ is satisfiable, then S is also satisfiable.*

Proof. We assume that S is not satisfiable.

1. We construct a candidate model $\mathcal{I}$ in Definition 5.19.
2. We can show that $\mathcal{I}$ is a model by Lemma 5.24 on page 45.
   That contradicts our assumption.

We discard our assumption and conclude that S is satisfiable. □

Definition 5.19 (Candidate Model Construction)**.** Let S be an Inst-saturated set of clauses, i.e. $\square \notin \mathcal{L}_S^{sat}$ and $(S\bot)$ is satisfiable.
Let $L = L' \cdot \sigma \in \mathcal{L}_S^{sat}$. We define inductively:

- $I_L = \{ \epsilon_M \mid L \succ_{\mathtt{L}} M \}$ IH: ϵ_M is defined for any $M \mid L \succ_{\mathtt{L}} M$

- $R_L = \{s \to t \mid s \approx t \in I_L, s \succ_{\mathtt{gr}} t\}$
- $\epsilon_L = \begin{cases} \emptyset & \text{if } L'\sigma \text{ reducible by } R_L \\ \emptyset & \text{if } I_L \vDash L'\sigma \text{ or } I_L \vDash \overline{L'}\sigma \quad \text{(defined)} \\ \{L'\sigma\} & \text{otherwise} \quad \text{(productive)} \end{cases}$
- $R_S = \bigcup_{L \in \mathcal{L}_S^{sat}} R_L$ $\qquad R_S$ is convergent and interreduced
- $I_S = \bigcup_{L \in \mathcal{L}_S^{sat}} \epsilon_L$ $\qquad I_S$ is consistent, $L\sigma \in I_S$ is irreducible by R_S

Let $\mathcal{I}$ be an arbitrary consistent extension of I_S in all the following lemmata.

Lemma 5.20. *If any $L \cdot \sigma \in \mathcal{L}_S$, irreducible by R_S exists with $\mathcal{I} \not\models L\sigma$ then there is a $L' \cdot \sigma' \in \mathrm{irred}_{R_S}(\mathcal{L}_S^{sat})$ with $\mathcal{I} \not\models L'\sigma'$.*

Proof. We have two cases:

- If $L \cdot \sigma$ is not UP-redundant in $\mathcal{L}_S^{sat}$, then $L' \cdot \sigma' = L \cdot \sigma$.
- Let $L \cdot \sigma$ be UP-redundant in $\mathcal{L}_S^{sat}$. By construction σ is irreducible by R_S. Then we have

  $$R_S \cup \mathrm{irred}_{R_S}(\{L' \cdot \sigma' \in \mathcal{L}_S^{sat} \mid L \cdot \sigma \succ_{\mathtt{L}} L' \cdot \sigma'\}) \models L\sigma$$

  with $\mathcal{I} \not\models L'\sigma'$. ✓

□

Lemma 5.21. *Whenever*

$$M \cdot \tau = \min_{\succ_{\mathtt{L}}} \Big\{ L' \cdot \tau' \mid L' \cdot \tau' \in \mathrm{irred}_{R_S}(\mathcal{L}_S^{sat}),\ L'\tau' \textit{ false in } \mathcal{I} \Big\}$$

is defined, then $M \cdot \tau$ is irreducible by R_S.

Proof. Assume $M \cdot \tau$ is reducible by $(\ell \to r) \in R_S$ and $(\ell \to r)$ is produced by $(\ell' \approx r') \cdot \rho \in \mathcal{L}_S^{sat}$

Now a UPC-inference is applicable because τ is irreducible by R_S:

$$\frac{(\ell' \approx r') \cdot \rho \quad M[\ell''] \cdot \tau}{M[r']\theta \cdot \mu}\ UPC$$

Then μ is irreducible by R_S, and $M[r']\theta\mu$ is false in $\mathcal{I}$ and we have two cases:

- If $M[r']\theta \cdot \mu$ is not UP-redundant in $\mathcal{L}_S^{sat}$ then $M[r']\theta \cdot \mu \in \mathcal{L}_S^{sat}$.

  Now $M \cdot \tau \succ_{\mathtt{L}} M[r']\theta \cdot \mu \in \mathrm{irred}_{R_S}(\mathcal{L}_S^{sat})$

  contradicts minimality of $M \cdot \tau$.

- If $M[r']\theta \cdot \mu$ is UP-redundant in $\mathcal{L}_S^{sat}$ then

$$R_S \cup \text{irred}_{R_S}\{M' \cdot \tau' \in \mathcal{L}_S^{sat} \mid M[r']\theta \cdot \mu \succ_{\mathtt{L}} M'\tau'\} \models M[r']\theta\mu$$

  Hence there is $M' \cdot \tau' \in \mathcal{L}_S^{sat}$ false in $\mathcal{I}$ such that $M \cdot \tau \succ_{\mathtt{L}} M[r']\theta \cdot \mu \succ_{\mathtt{L}} M' \cdot \tau'$, and $M' \cdot \tau'$ contradicts minimality of $M \cdot \tau$.

Hence $M \cdot \tau$ is irreducible by R_S. □ □

Lemma 5.22. *Let $M \cdot \tau \in \mathcal{L}_S^{sat}$, irreducible by R_S, and defined (not productive). From $\mathcal{I} \not\models M\tau$ follows that M is not an equation $(s \approx t)$.*

Proof. Assume $M = (s \approx t)$. Then we have

- $I_{M\cdot\tau} \models (s \not\approx t)\tau$
- All literals in $I_{M\cdot\tau}$ are irreducible by $R_{M\cdot\tau}$
- $s\tau$ and $t\tau$ are irreducible by $R_{M\cdot\tau}$
- $R_{M\cdot\tau}$ is a convergent term rewrite system

Hence $(s \not\approx t)\tau \in I_{M\cdot\tau}$ is produced to $I_{M\cdot\tau}$ by some $(s' \not\approx t')\cdot\tau'$, but $(s' \not\approx t')\tau' \succ_{\mathtt{gr}} (s \approx t)\tau$ and $(s' \not\approx t') \cdot \tau' \succ_{\mathtt{L}} M \cdot \tau$.

□

Lemma 5.23. *Let $M \cdot \tau \in \mathcal{L}_S^{sat}$ be irreducible by R_S, and defined (not productive). From $\mathcal{I} \not\models M\tau$ it follows that M is not an inequation $(s \not\approx t)$.*

Proof. Assume $M \cdot \tau$ is inequation $(s \not\approx t) \cdot \tau$. We have

- $I_{M\cdot\tau} \models (s \approx t)\tau$
- $s\tau$ and $t\tau$ are irreducible by $R_{M\cdot\tau}$

Hence $s\tau = t\tau$ and equality resolution is applicable.
This contradicts $\square \notin \mathcal{L}_S^{sat}$.

□

Lemma 5.24. *$\mathcal{I}$ is a model for all ground instances of S.*

Proof. Assume $\mathcal{I}$ is not a model.
Hence a minimal ground closure $D = \min_{\succ_{\mathtt{C}}}\{ C' \cdot \sigma \mid C' \in S, \mathcal{I} \not\models C'\sigma \}$, an instance of a clause in S, false in $\mathcal{I}$, must exist. Furthermore

- $D = D' \cdot \sigma$ is not Inst-redundant.

  Otherwise by Definition 5.14 there are $D_1, \ldots, D_n \models D$, $D \succ_{\mathtt{C}} D_i$ for all i,

  and D_j is false in $\mathcal{I}$ for some j, which contradicts minimality.

- $x\sigma$ is irreducible by R_S for every variable x in D'.

  Otherwise let $(\ell \to r)\tau \in R_L$ and $x\sigma = x\sigma[l\tau]_p$ for some variable x in D'. We define the substitution σ' such that $x\sigma' = x\sigma[r\tau]_p$ and $y\sigma' = y\sigma$ for $y \neq x$. $D'\sigma'$ is false in $\mathcal{I}$ and $D \succ_{\mathtt{C}} D' \cdot \sigma'$, which contradicts minimality.

Since D is not Inst-redundant in S, we have for some literal L, that $D' = L \vee D''$, $\mathrm{sel}(D') = L$, $L \cdot \sigma \in \mathcal{L}_S$, $L\sigma$ is false in $\mathcal{I}$.

Hence the following literal closure

$$M \cdot \tau = \min_{\succ_{\mathtt{L}}} \Big\{ L' \cdot \tau' \mid L' \cdot \sigma' \in \mathrm{irred}_{R_S}(\mathcal{L}_S^{sat}),\, \mathcal{I} \not\models L' \cdot \sigma' \Big\}$$

exists by Lemma 5.20, is irreducible by Lemma 5.21, and is not productive. Since $\mathcal{I} \not\models M \cdot \tau$ the literal M cannot be an equation by Lemma 5.22 or an inequation by Lemma 5.23. We have derived a contradiction from our only assumption.

Therefore $\mathcal{I}$ is a model for all instances of S, hence Theorem 5.18 on page 43 holds. □

5.3 Saturation Strategies

So far we have only shown that a Inst-saturated set of clauses S is satisfiable if $S\bot$ is satisfiable. Now we take a look at how Inst-saturation can be achieved starting from a set of clauses S^1 with satisfiable set $S^1\bot$.

Definition 5.25. An Inst-*saturation process* is a sequence of triples $\left(\langle S^i, I_\bot^i, \mathrm{sel}^i\rangle\right)_{i=1}^{\infty}$ where S^i is a set of clauses, $I_\bot^i$ a model for $S^i\bot$, and sel^i a selection function based on that model. A valid successor state $\langle S^{i+1}, I_\bot^{i+1}, \mathrm{sel}^{i+1}\rangle$ satisfies one of the following conditions:

- $S^{i+1} = S^i \cup N$ where N is a set of clauses such that $S^i \vDash N$
- $S^{i+1} = S^i \setminus \{C\}$ where clause C is Inst-redundant in S^i.

If $S^{i+1}\bot$ is unsatisfiable the process terminates with result "unsatisfiable". Otherwise we build $I_\bot^{i+1}$ and sel^{i+1} from $S^{i+1}\bot$. The set of persistent clauses S^∞ denotes the lower limit of $\{S^i\}_{i=1}^{\infty}$.

Definition 5.26. Let $K = \{(L_1 \vee C_1) \cdot \sigma, \ldots, (L_n \vee C_n) \cdot \sigma\}$ be a finite set of closures of clauses from S^∞. Let $\mathcal{L} = \{L_1 \cdot \sigma, \ldots, L_n \cdot \sigma\}$. The pair (K, L) is a *permanent conflict* if $\mathcal{L}^{sat}$ contains the empty clause and for infinitely many i we have $\mathrm{sel}^i(L_j \vee C_j) = L_j$ for $1 \leq j < n$. An Inst-saturation process is *Inst-fair* if for every persistent conflict (K, L) at least one of the closures in K is Inst-redundant in S^i for some i.

Lemma 5.27. *Let S^∞ be a set of persistent clauses of an Inst-fair saturation process $\{\langle S^i, {I_\bot}^i, \mathrm{sel}^i\rangle\}_{i=1}^{\infty}$, and let $S^\infty\bot$ be satisfiable. Then there exists a model $I\bot$ of $S^\infty\bot$ and a selection function* sel *based on $I\bot$ such that S^∞ is Inst-saturated w.r.t.* sel.

Proof. [9]] ... □

In an *Inst-fair* saturation process the set of ground instances $S^i\bot$ is either satisfiable for all i or unsatisfiable for some i. In the first case an Inst-saturated S^i with satisfiable $S^i\bot$ proves satisfiability of S^1, while in the second case the first unsatisfiable $S^i\bot$ provides evidence of the unsatisfiability of S^1. It remains to ensure that the Inst-saturation process is Inst-fair.

We represent UP-proofs as binary trees. We label the nodes by closures and substitutions from the corresponding inferences. At each node we assume that the left proof branch is variable disjoint from the right proof branch (which can be achieved by variable renaming). We construct a *P-relevant instantiator* of literal $L \in \mathcal{L}$ as composition $\theta = \theta_1 \cdots \theta_k$ of the sequence of substitutions $(\theta_i)_{i=1}^k$ along the path of length k from leaf $L \cdot \sigma$ to the root of a proof P. Further we construct the *P-relevant instance* $L\theta \cdot \tau$ from $L \cdot \sigma$ with θ with $L\theta\tau = L\sigma$.

Lemma 5.28. *Let $\mathcal{P}$ be the set of all P-relevant instances of all leaves of a proof P of the empty clause. Then $\mathcal{P}\bot$ is unsatisfiable.*

Corollary 5.29. *Let $(K, \mathcal{L})$ be a persistent conflict and P a proof of the empty clause from $\mathcal{L}$ in UP, then at least one P-relevant instantiator is proper.*

We make closures in K of a persistent conflict $(K, \mathcal{L})$ Inst-redundant by adding their P-relevant proper instances. We make an Inst-saturation process fair by UP-saturating literal closures from persisting conflicts and adding proper instantiations of clauses with substitutions gained from proofs of the empty clause.

5.4 Equational Reasoning on Equation Literals

We now lift unit superposition and resolution form closures to literals.

Definition 5.30 (Unit superposition and resolution on literals [9, 29], UPL)**.**

$$\frac{(\ell \approx r) \quad (\underset{\approx}{L}[\ell'])}{(\underset{\approx}{L}[r])\sigma}\ (\sigma) \qquad\qquad \frac{(s' \not\approx t')}{\square}\ (\sigma') \qquad\qquad \sigma' = \mathrm{mgu}(s', t')$$

with $\underset{\approx}{L}[\ell'] \in \{s[\ell'] \not\approx t, s[\ell'] \approx t\}$ where for some grounding substitution θ

- $\sigma = \mathrm{mgu}(\ell, \ell')$, $\ell' \notin \mathcal{V}$, $l\sigma\theta \succ_{\mathtt{gr}} r\sigma\theta$, $s[\ell']\sigma\theta \succ_{\mathtt{gr}} t\sigma\theta$
- $\mathit{Vars}(\{\ell, r\}) \cap \mathit{Vars}(\{s[\ell'], t\}) = \emptyset$
- $\mathrm{dom}(\theta) \subseteq \mathit{Vars}(\{\ell\sigma, r\sigma, s[\ell']\sigma, t\sigma\})$

We construct P-relevant instantiators and P-relevant instances of literals $L\sigma$ from proofs in UPL of the empty clause in the same way as for UPC since proof in UPL can be represented as proofs in UPC by the following lemma.

Lemma 5.31. *[9] Let Lit be a set of literals such that Lit$\bot$ is satisfiable. Further let $\mathcal{L}$ be a set of ground closures from Lit such that the empty clause is derivable in* UPC *from $\mathcal{L}$. Then there is a proof of the empty clause in* UPL *from Lit such that for at least one closure $L \cdot \sigma \in \mathcal{L}$, P-relevant instance of L is $L\theta$ where θ is a proper instantiator and $L\sigma = L\theta\tau$ for some ground substitution τ.*

5.5 Equational Reasoning on Predicate Literals

So far the calculi in Definitions 5.3 (UPC) and 5.30 (UPL) dealt with equations only. In this section we translate first order logic into purely equational logic and justify the inference rules in Definition 4.32 on page 34. Intuitively we just replace predicate symbols with Boolean function symbols and demand that functional predicates evaluate to true or not true in a model.

Definition 5.32. Let $\mathcal{F} = \mathcal{F}_\mathsf{f} \dot\cup \mathcal{F}_\mathsf{P} \dot\cup \{\approx\}$ be a first order signature. We construct a set of "Boolean" function symbols

$$\mathcal{F}_\mathbb{B} = \{\, \mathsf{P}_\mathbb{B} \mid \mathsf{P} \in \mathcal{F}_\mathsf{P} \,\}$$

from the set of predicate symbols $\mathcal{F}_\mathsf{P}$. We construct an extended set of function symbols

$$\mathcal{F}_\mathsf{f}' = \mathcal{F}_\mathsf{f} \dot\cup \mathcal{F}_\mathbb{B} \dot\cup \{\mathsf{c}_\mathbb{B}\}$$

from *uninterpreted* function symbols $\mathcal{F}_\mathsf{f}$, Boolean function symbols $\mathcal{F}_\mathbb{B}$, and fresh Boolean constant $\mathsf{c}_\mathbb{B}$, such that $\mathrm{arity}(\mathsf{P}_\mathbb{B}) = \mathrm{arity}(\mathsf{P})$ for all $\mathsf{P}_\mathbb{B} \in \mathcal{F}_\mathbb{B}$. We extend our term order $\succ_{\mathtt{gr}}$ to the new symbols such that $\mathsf{c}_\mathbb{B}$ is the smallest Boolean ground term, i.e. the smallest ground term with a Boolean function symbol at its root position (and no other position).

Definition 5.33. We translate a set of clauses over signature $\mathcal{F}$ into an equisatisfiable and purely equational set of clauses over the purely equational signature $\mathcal{F}'_\approx = \mathcal{F}'_\mathsf{f} \dot\cup \{\approx\}$ by replacing predicates with equations.

$$T_\approx(S) = \bigcup_{C \in S} T_\approx(C) \qquad T_\approx(C) = \bigcup_{L \in C} T_\approx(L)$$

$$T_\approx(s \approx t) = s \approx t \qquad T_\approx(s \not\approx t) = s \not\approx t$$

$$T_\approx(\mathsf{P}(t_1, \ldots, t_n)) = \mathsf{P}_\mathbb{B}(t_1, \ldots, t_n) \approx \mathsf{c}_\mathbb{B} \qquad T_\approx(\neg\mathsf{P}(t_1, \ldots, t_n)) = \mathsf{P}_\mathbb{B}(t_1, \ldots, t_n) \not\approx \mathsf{c}_\mathbb{B}$$

The original equations of terms stay unchanged while we replace each predicate with an equation, and each negated predicate with an inequation, where we have a Boolean term on the left side and the unique Boolean constant $\mathsf{c}_\mathbb{B}$ on the right side.

Corollary 5.34. *We can represent proofs in `Inst-Gen-Eq` as proofs in UPL.*

Proof. Unit superposition and unit equality resolution work the same. We simulate derivation steps with predicates in `Inst-Gen-Eq` as proofs with translated predicates in UPL for

- unit paramodulation

$$\frac{s \approx t \quad A[s'] \not\approx \mathsf{c}_{\mathbb{B}}}{A[t]\sigma \not\approx \mathsf{c}_{\mathbb{B}}}\ (\sigma) \qquad\qquad \frac{s \approx t \quad A[s'] \approx \mathsf{c}_{\mathbb{B}}}{A[t]\sigma \approx \mathsf{c}_{\mathbb{B}}}\ (\sigma)$$

  where $\sigma = \mathrm{mgu}(s, s')$ is defined, $s' \notin \mathcal{V}$, $s\sigma\theta \succ_{\mathtt{gr}} t\sigma\theta$, $(A[s']\sigma\theta \succ_{\mathtt{gr}} \mathsf{c}_{\mathbb{B}})$ for some grounding substitution θ;

- unit resolution

$$\frac{\dfrac{A \approx \mathsf{c}_{\mathbb{B}} \quad \neg B \not\approx \mathsf{c}_{\mathbb{B}}}{\mathsf{c}_{\mathbb{B}} \not\approx \mathsf{c}_{\mathbb{B}}}\ (\sigma)}{\square}\ (\emptyset)$$

  where $\sigma = \mathrm{mgu}(A, B)$ is defined, $(A\sigma\theta \succ \mathsf{c}_{\mathbb{B}}$ for any grounding substitution $\theta)$

□

Remark. The transformation from predicate logic to purely equational logic and the application of the unit paramodulation to translated literals are well defined, e.g. we will not end up with literals like $x \approx \mathsf{c}_{\mathbb{B}}$, $\mathsf{P}_{\mathsf{f}}(x) \approx x$ or $\mathsf{f}(\mathsf{P}_{\mathsf{f}}(x)) \approx x$.

Implicitly we define two sorts of function symbols in this transformation, uninterpreted and Boolean function symbols. Boolean function symbols only appear at the root positions of terms. The root symbols of the two terms of an equation are always of the same sort.

Due to the extended order $\succ_{\mathtt{gr}}$, where $\mathsf{c}_{\mathbb{B}}$ is the smallest term, even $\mathsf{P}_{\mathsf{f}}(x) \approx \mathsf{Q}_{\mathsf{f}}(x)$ cannot be derived from literals $\mathsf{P}_{\mathsf{f}}(x) \approx \mathsf{c}_{\mathbb{B}}$ and $\mathsf{Q}_{\mathsf{f}}(x) \approx \mathsf{c}_{\mathbb{B}}$. So we can always transform derived literals back to predicate logic. Further we can easily transform predicate models to equational models and vice versa.

$$\begin{array}{lcl} \mathcal{M} \models \mathsf{P}(s) & \iff & \mathcal{M}' \models \mathsf{P}(s) \approx \mathsf{c}_{\mathbb{B}} \\ \mathcal{M} \models \neg\mathsf{P}(s) & \iff & \mathcal{M}' \models \mathsf{P}(s) \not\approx \mathsf{c}_{\mathbb{B}} \end{array}$$

i.e. $s \in P_{\mathcal{M}}$ if and only if the evaluations $v'_{\mathcal{M}'}(\mathsf{P}(s)) = v'_{\mathcal{M}'}(\mathsf{c}_{\mathbb{B}})$ are equal.

List of Figures

List of Tables

Bibliography

[1] L. Albert, R. Casas, and F. Fages. Average-case analysis of unification algorithms. *Theoretical Computer Science*, 113(1):3 – 34, 1993.

[2] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, New York, NY, USA, 1998.

[3] D. W. Bennett. An elementary completeness proof for a system of natural deduction. *Notre Dame J. Formal Logic*, 14(3):430–432, 07 1973.

[4] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 2009.

[5] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problem*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1997. With an appendix by Cyril Allauzen and Bruno Durand.

[6] A. Church. A note on the entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):40–41, 1936.

[7] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, July 1962.

[8] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, July 1960.

[9] H. Ganzinger and K. Korovin. Integrating Equational Reasoning into Instantiation-Based Theorem Proving. In *18th CSL 2004. Proceedings*, volume 3210 of *LNCS*, pages 71–84, 2004.

[10] P. C. Gilmore. A proof method for quantification theory: Its justification and realization. *IBM Journal of Research and Development*, 4(1):28–35, Jan 1960.

[11] K. Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37(1):349–360, Dec 1930.

[12] P. Graf and D. Fehrer. *Term Indexing*, pages 125–147. Springer Netherlands, Dordrecht, 1998.

[13] R. Hähnle. Tableaux and related methods. In Robinson and Voronkov [27], pages 100–178.

[14] J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[15] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA, 2004.

[16] K. Korovin. Inst-Gen – a Modular Approch. In *IJCAR 2008. Proceedings*, pages 292–298.

[17] K. Korovin and C. Sticksel. iProver-Eq: An Instantiation-Based Theorem Prover with Equality. In *IJCAR 2010. Proceedings*, volume 6173 of *LNAI*, pages 196–202, 2010.

[18] D. Kroening and O. Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[19] T. A. M. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265.

[20] A. Middeldorp. Lecture Notes – Term Rewriting, 2015.

[21] A. Middeldorp. Lecture Notes – Logic, 2016.

[22] G. Moser. Lecture Notes – Module Automated Reasoning, 2013.

[23] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 371–443. Elsevier, 2001.

[24] N. Olivetti and A. Tiwari, editors. *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*. Springer, 2016.

[25] D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2(3):293 – 304, 1986.

[26] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, Jan. 1965.

[27] J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 2001.

[28] S. Schulz and M. Möhrmann. Performance of clause selection heuristics for saturation-based theorem proving. In N. Olivetti and A. Tiwari, editors, *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*, pages 330–345. Springer, 2016.

[29] C. Sticksel. *Efficient Equational Reasoning for the Inst-Gen framework*. PhD thesis, School of Computer Science, University of Manchester, 2011.

[30] G. Sutcliffe. The CADE ATP System Competition - CASC. *AI Magazine*, 37(2):99–101, 2016.

[31] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[32] G. S. Tseitin. On the complexity of derivations in the propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic*, Part II:115–125, 1970.

[33] L. Wos, G. A. Robinson, and D. F. Carson. Efficiency and completeness of the set of support strategy in theorem proving. *J. ACM*, 12(4):536–541, Oct. 1965.