Tree grammars for induction on inductive data types modulo equational theories—supplemental material*

Gabriel Ebner and Stefan Hetzl

TU Wien

A Finding covering induction grammars via MaxSAT

We will now reduce the Paramerized Indexed Termset Cover Problem to a variant of the MaxSAT optimization problem, for which efficient off-the-shelf solvers exist (with yearly competitions [1]). The approach is similar to the one used in [3], but is based on the more recent compression of VTRATGs in [2] which allows us to generate grammars for $\overline{\gamma}$-vectors of arbitrary length. We also extend the algorithm to many-sorted terms and grammars. The reduction itself will be polynomial-time, but MaxSAT is well-known to be NP-complete.

Computational Problem 1 (Unweighted, partial MaxSAT, MaxSAT*).*
Input: Two sets of propositional clauses H and S (called "hard" and "soft" clauses, resp.).
Output: Interpretation I such that $I \models H$ and I maximizes the number of satisfied clauses in S.

The main idea is that we do not need to search for all possible grammars, it is sufficient to only consider grammars where all productions are "stable". In fact, the grammar $S((L_i)_{i\in I})$ consisting of all stable productions is of polynomial size and we can transform every covering grammar into a subgrammar (a subset of the productions) without increase in size.

Definition 1. *Let L be a set of ground terms, t, s by any terms, and define $t \sqsubseteq_L s$ if and only if $t\sigma \trianglelefteq r$ for some $r \in L$ implies $t\sigma = s\sigma$ for all substitutions σ. We say that t is* stable *(for L) if and only if $t \sqsubseteq_L s$ implies $s \sqsubseteq_L t$ for all s.*

Definition 2. *Let $(L_i)_{i\in I}$ be a ρ-indexed term set, and $\alpha, (\overline{\nu_c})_c, \overline{\gamma}$ as in* **??**. *Then the* stable grammar *$S((L_i)_{i\in I}) = (\tau, \alpha, (\overline{\nu_c})_c, \overline{\gamma}, P)$ contains all possible productions where the right-hand side is a stable term, that is:*

$$P = \{\overline{\kappa} \to \overline{t}[\alpha, \overline{\nu_i}, \overline{\gamma}] \mid \overline{\kappa} \in \{\tau, \overline{\gamma}\} \wedge \overline{t}[\alpha, \overline{\nu_i}, \overline{\gamma}] \in S\big(\bigcup_i L_i\big)\}$$

Let us first show that we can transform grammars into subgrammars of the stable grammar without increase in size.

* Supported by the Vienna Science and Technology Fund (WWTF) project VRG12-004

Definition 3. *Let $G = (\tau, \alpha, (\overline{\nu_c})_c, \overline{\gamma}, P)$ and $G' = (\tau, \alpha, (\overline{\nu_c})_c, \overline{\gamma}, P')$ be induction grammars. We say that G' is a subgrammar of G, written $G' \subseteq G$, iff $P' \subseteq P$.*

Theorem 1. *Let $(L_i)_{i\in I}$ be a ρ-indexed term set, α of type ρ, and $G = (\tau, \alpha, (\overline{\nu_c})_c, \overline{\gamma}, P)$ an induction grammar that covers $(L_i)_{i\in I}$. Then there exists a grammar $G' = (\tau, \alpha, (\overline{\nu_c})_c, \overline{\gamma}, P')$ such that:*

- *G' covers $(L_i)_{i\in I}$*
- *$|G'| \leq |G|$*
- *$G' \subseteq S((L_i)_{i\in I})$*

Proof (sketch). Analogous to Theorem 5.3 in [2]. We replace the right-hand side of every production in G by one of its $\sqsubset_L$-normal forms, which are stable. The resulting grammar still covers $(L_i)_{i\in I}$ since we can commute language generation and rewriting by $\sqsubseteq_L$.

The following theorem implies that the stable grammar is polynomial-time computable for fixed $\overline{\gamma}$ and bounded I.

Theorem 2. *Let X be a fixed finite set of variables. Then the set of stable terms with variables from X is polynomial-time computable from a set of terms L.*

Proof (sketch). This theorem is shown for single-sorted terms as Theorem 6.13 in [2], but we can extend it to the many-sorted case via type erasure: let E denote the type erasure function that maps many-sorted terms to single-sorted terms. If $E(t) \sqsubseteq_{E(L)} s$ and t subsumes a subterm of L, then s is well-typed and $t \sqsubseteq_L s'$ for some s' such that $E(s') = s$. This observation implies that if t is stable for L, then $E(t)$ is stable for $E(L)$. Let $S(L)$ be the set of terms that are stable for L. Since E is injective, we can compute $S(L)$ from $S(E(L))$ by inferring the types. This shows polynomial-time computability of $S(L)$.

Now we can assume without loss of generality that we want to find a subgrammar of $S((L_i)_{i\in I})$, i.e., a subset of its productions. Given a VTRATG $G = (\tau, N, P)$, Definition 7.3 in [2] constructs a propositional formula $\mathrm{GenTerm}(G, t)$ such that for every subset $P' \subseteq P$ of the VTRATG, the formula $\mathrm{GenTerm}(G, t) \wedge \bigwedge_{p\in P\setminus P'} \neg(p \in P'(G))$ is satisfiable if and only if $t \in L((\tau, N, P'))$. We will also define the minimization of induction grammar in terms of their instance grammars:

Definition 4. *Let $(L_i)_{i\in I}$ be an indexed term set, and $G = (\tau, \alpha, (\overline{\nu_c})_c, \overline{\gamma}, P)$ an inductive grammar. We define the propositional formula $\mathrm{Gen}(G, (L_i)_{i\in I})$ as the following formula:*

$$\bigwedge_{i\in I} \bigwedge_{t\in L_i} \mathrm{GenTerm}(I(G,i), t) \quad \wedge \quad \bigwedge_{i\in I} \bigwedge_{p'} p' \in P'(I(G,i)) \to \bigvee_{p \rightsquigarrow p'} p \in P$$

Theorem 3. *Let $(L_i)_{i\in I}$ be an indexed term set, G an inductive grammar, and $G' \subseteq G$ a subgrammar. Then the formula $\mathrm{Gen}(G, (L_i)_{i\in I}) \wedge \bigwedge\{\neg p \in P' \mid p \in G \setminus G'\}$ is satisfiable if and only if G' covers $(L_i)_{i\in I}$.*

Proof (sketch). By Lemma 7.5 of [2], the formula is satisfiable iff for all $i \in I$, the instance grammar $I(G', i)$ covers L_i. The second conjunct of Definition 4 then ensures that whenever an instantiated production is used to cover a term in the instance grammar, it is already included in the induction grammar.

Algorithm 1 Cover an indexed term set by an induction grammar

procedure FINDGRAMMAR($\sigma, \overline{\tau}, (L_i)_i$)
  $\mathtt{S} \leftarrow S((L_i)_i)$
  $\mathtt{hard} \leftarrow \mathrm{Gen}(\mathtt{S}, (L_i)_i)$
  $\mathtt{soft} \leftarrow \{\neg p \in P' \mid p \in \mathtt{S}\}$
  if $\mathrm{Sat}(\mathtt{I}) \leftarrow$ MAXSAT($\mathtt{hard}, \mathtt{soft}$) **then**
    return $\{p \mid \mathtt{I} \models p \in P'\}$
  else
    fail
  end if
end procedure

Definition 5. *Let $G = (\tau, \alpha, (\overline{\nu}_c)_c, \overline{\gamma}, P)$ be an induction grammar. Its* size $|G|$ *is the number of productions $|P|$.*

Lemma 1. *Let $(L_i)_{i \in I}$ be an indexed term set, then Algorithm 1 produces an induction grammar covering $(L_i)_{i \in I}$ of minimal size, or fails if there is no covering grammar.*

Proof. First, assume that there exists a covering induction grammar; by Theorem 1 there is a grammar G covering $(L_i)_{i \in I}$ of minimal size such that $G \subseteq S((L_i)_{i \in I})$. By Theorem 3, the formula $\mathtt{hard} \wedge \bigwedge_{p \in \mathtt{S} \setminus \mathrm{G}} \neg(p \in P')$ is satisfiable. Hence the interpretation I returned by the MaxSAT solver sets at most $|G|$ many $p \in P'$ atoms to true, corresponding to a grammar with at most $|G|$ productions. If there is no covering grammar, then MAXSAT will return unsatisfiable as return status, and Algorithm 1 will fail.

References

1. Argelich, J., Li, C.M., Manya, F., Planes, J.: The first and second Max-SAT evaluations. Journal on Satisfiability, Boolean Modeling and Computation 4(2-4), 251–278 (2008)
2. Eberhard, S., Ebner, G., Hetzl, S.: Algorithmic compression of finite tree languages by rigid acyclic grammars. ACM Transactions on Computational Logic 18(4), 26:1–26:20 (Sep 2017)
3. Eberhard, S., Hetzl, S.: Inductive theorem proving based on tree grammars. Annals of Pure and Applied Logic 166(6), 665–700 (2015)