Kalpa Publications in Computing

Volume 1, 2017, Pages 119–123

LPAR-21S: IWIL Workshop
and LPAR Short Presentations

Towards an Abstraction-Refinement Framework for Reasoning with Large Theories

Julio Cesar Lopez Hernandez and Konstantin Korovin

The University of Manchester, School of Computer Science
{lopezhej,korovin}@cs.man.ac.uk

Abstract

In this paper we present an approach to reasoning with large theories which is based on the abstraction-refinement framework. The proposed approach consists of the following approximations: the over-approximation, the under-approximation, and their combination.

1 Introduction

Efficient reasoning with large theories is one of the main challenges in automated theorem proving arising in many applications ranging from reasoning with ontologies to proof assistants for mathematics. Current methods for reasoning with large theories are based on different axiom selection methods. Some of them are based on the syntactic or semantic structure of the axioms and conjecture formulas [11, 6]. Other methods for axiom selection use machine learning to take advantage of previous knowledge about proving conjectures [13, 14]. What those methods have in common are two phases of the whole process for proving a conjecture: one is the axiom selection phase, and the other one is reasoning phase. Those phases are performed in a sequential way. First, the axiom selection takes place, then using the selected axioms the reasoning process starts.

Our proposed approach based on abstraction-refinement framework [5] has the purpose of interleaving the axioms selection and reasoning phases, having a more dynamic interaction between them. This proposed approach encompasses two ways for approximating axioms: one is called over-approximation and the other one under-approximation. Those approximations are combined to converge more rapidly to a proof if it exists or to a model otherwise. There are a number of related works which consider different specific types of under and/or over approximations in different contexts [9, 3, 12, 7, 10, 4].

2 Preliminaries

We consider a theory A which is a collection of axioms which we call *concrete axioms* and two sets of formulas: $\hat{A}^s$ called *stronger abstract axioms* and $\hat{A}^w$ called *weaker abstract axioms*.

A *strengthening abstraction function* α_s is a mapping $\alpha_s : A \to \hat{A}^s$, which maps concrete axioms in A to stronger abstract axioms in $\hat{A}^s$. A *stronger abstract axiom* $\hat{a}^s$ is an element

T.Eiter, D.Sands, G.Sutcliffe and A.Voronkov (eds.), LPAR-21S (Kalpa Publications in Computing, vol. 1), pp. 119–123

of $\hat{A}^s$ such that $\hat{a}^s = \alpha_s(a)$ and $\hat{a}^s \models a$, where a is a concrete axiom; i.e., α_s strengthens the axioms in the sense that they are a more general representation of the concrete axioms. A *concretisation function* γ_s is a mapping $\gamma_s : \hat{A}^s \to 2^A$ such that $a \in \gamma_s(\alpha_s(a))$.

A *weakening abstraction function* α_w is a mapping $\alpha_w : A \to \hat{A}^w$, which maps concrete axioms in A to weaker abstract axioms in $\hat{A}^w$. A *weaker abstract axiom* $\hat{a}^w$ is an element of $\hat{A}^w$ such that $\hat{a}^w = \alpha_w(a)$ and $a \models \hat{a}^w$ where a is a concrete axiom.

Abstraction refinement is a process to approximate an abstract representation of axioms $\hat{A}$ to their concrete representation A. *Weakening abstraction refinement* is a process to construct $\hat{A}^{s\prime}$, which is a closer representation of A from $\hat{A}^s$ such that $\hat{A}^s \models \hat{A}^{s\prime}$ and $\hat{A}^{s\prime} \models A$; i.e., this refinement weakens the abstract axioms in $\hat{A}^s$. However those weak abstract axioms are still stronger than the concrete axioms. *Strengthening abstraction refinement* constructs $\hat{A}^{w\prime}$ which is an approximation to the set of concrete axioms A, such that $A \models \hat{A}^{w\prime}$ and $\hat{A}^{w\prime} \models \hat{A}^w$.

We use ATP_S to denote an instance of automated theorem prover which is sound and could be complete or not. On the other hand, we use ATP_C to make reference to an automated theorem prover which is complete but not necessary sound [9, 3]. The purpose of these ATPs is to prove or disprove conjectures more efficiently than a sound and complete ATP but with a possible loss of precision.

3 Over-Approximation

This procedure starts by applying the strengthening abstract function α_s to A, to obtain an abstract representation of axioms $\hat{A}^s$, $\hat{A}^s = \alpha_s(A)$. Utilising the set $\hat{A}^s$, the procedure tries to prove the conjecture C using an ATP_C. If the ATP_C disproves the conjecture, the process finishes and responds that the conjecture has been disproved. If ATP_C proves the conjecture the procedure uses the abstract axioms involved in the proof $\hat{A}^s_p$ to retrieve their concrete axioms in A by applying the function γ_s over $\hat{A}^s_p$. The retrieved concrete axioms form a new subset A_p, where $A_p = \gamma_s(\hat{A}^s_p)$. With the new set A_p, the procedure tries again to prove the conjecture using this time an ATP_S. If the ATP_S proves the conjecture, the process stops and provides the proof. Otherwise, if the time limit is reached or the conjecture is disproved, the set of axioms $\hat{A}^s$ is refined using the weakening abstraction refinement. The procedure is repeated utilising the refined set of abstract axioms. This loop finishes when the conjecture is proved or disproved or the time limit of the whole procedure is reached. The diagram of this approximation is shown in Figure 1.

3.1 Strengthening Abstraction Function

In the over-approximation approach, one candidate for the abstraction function α_s is a function that generalises concrete axioms, for example, by replacing certain non-variable terms with variables or by removing literals. This transformation will satisfy the required property that the obtained abstract axioms entail the concrete axioms, $\hat{a}^s \models a$. We propose to define such an abstraction function, as follows. First we group concrete axioms based on their syntactic structure obtaining a partitioning of $A = \cup A_i$. Then for each group of concrete axioms A_i we associate an abstract axiom $\hat{a}^s_i$ which logically implies all concrete axioms from the group, i.e., $\hat{a}^s_i \models A_i$. The resulting abstraction function is defined as $\alpha_s(a) = \hat{a}^s_i$ for $a \in A_i$. For example, if A is a set of clauses, then we can group clauses based on their joint literal occurrences and define $\hat{a}^s_i$ to be a single clause that subsumes all clauses in A_i. The abstract axioms can be also restricted to a decidable fragment. For example, by replacing complex terms with variables we can over-approximate concrete axioms with abstract axioms in the EPR fragment. The

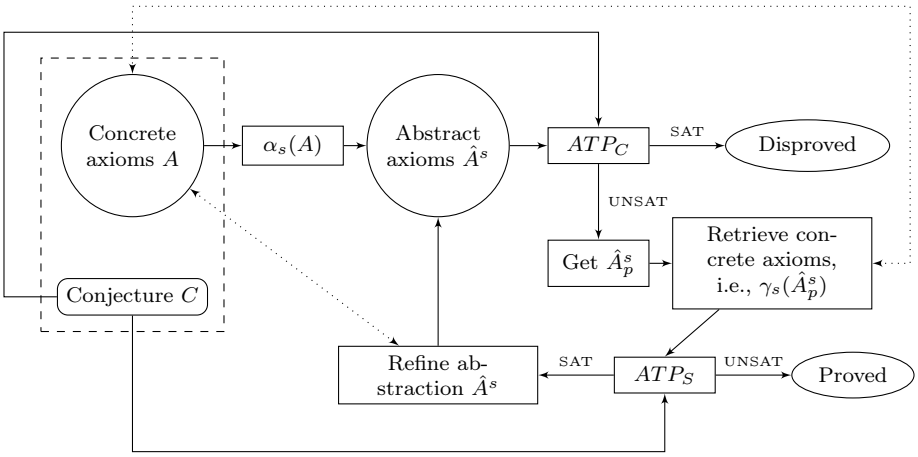

Figure 1: Over-approximation

EPR fragment consists of formulas which in the clausal form do not contain function symbols other than constants. Formulas in this fragment can be efficiently solved by instantiation-based methods like Inst-Gen [7] or Model Evolution [2]. Other over-approximation abstractions can be based on *argument filtering* (which is also used for proving termination of term rewriting systems [1, 8]), and *signature grouping* where we group signature symbols of the same type and replace the whole group by a single representative.

3.2 Weakening Abstraction Refinement

Weakening abstraction refinement is applied when a proof is obtained using abstract axioms (Figure 1). The abstraction refinement weakens the abstract axioms in $\hat{A}^s$, involved in the proof. For example, we can sub-partition groups of concrete axioms which were used to define α_s into smaller subgroups. The refined abstraction function is the abstraction function which assigns abstract axioms to the new partitioning, as defined in Section 3.1. The refinement process is goal directed in the sense that only groups corresponding to abstract axioms involved in the proof are partitioned further. When an axiom group consists of a single axiom then we can concretise the corresponding abstract axiom to coincide with the concrete axiom. The refinement process stops if each abstract axiom involved in the proof is concrete and in this case we obtain a concrete proof of the conjecture. A simple instance of abstraction refinement would be to concretise all abstract axioms involved in the proof: $\hat{A}^s := (\hat{A}^s \setminus \hat{A}^s_p) \cup \gamma_s(\hat{A}^s_p)$.

4 Under-Approximation

The process starts by applying the weakening abstraction function to the set of concrete axioms A, $\hat{A}^w = \alpha_w(A)$. This set $\hat{A}^w$ of weaker axioms is used to prove the conjecture, using an ATP_S. If the conjecture is proved the procedure stops and provides the proof. Otherwise, a model I of $\hat{A}^w$ and the negated conjecture is obtained. This model is used to refine the set of weaker axioms $\hat{A}^w$. During this refinement (strengthening abstraction refinement), the procedure tries to find a set of axioms $\breve{A}$ that turns the model into a countermodel but are still implied by A, i.e., $I \not\models \breve{A}$ and $A \models \breve{A}$. If the set of axioms $\breve{A}$ is empty, $\breve{A} = \emptyset$, the procedure stops and

disproves the conjecture. Otherwise, the obtained set of axioms is added to the set of weaker axioms, $\hat{A}^w := \hat{A}^w \cup \breve{A}$. Using this new set of abstract axioms $\hat{A}^w$, another round for proving the conjecture starts. The process finishes when the conjecture is proved or disproved or the time limit for the quest of a proof is reached. The diagram of this procedure is shown in Figure 2.

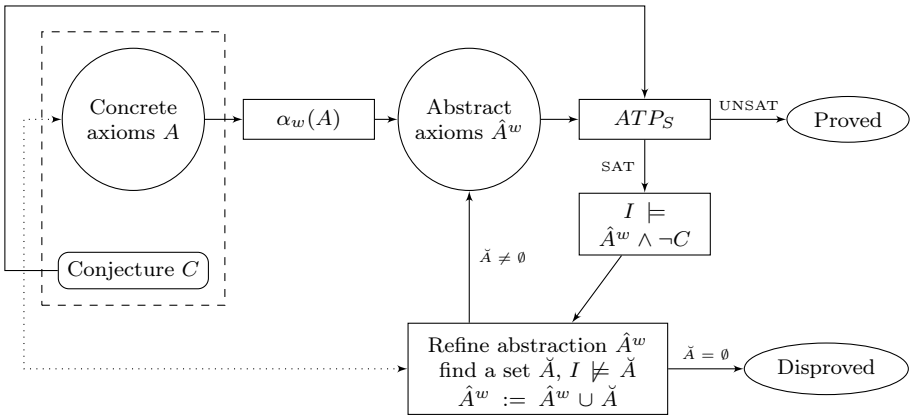

Figure 2: Under-approximation

4.1 Weakening Abstraction Function

In the case of under-approximation, we propose two weakening abstractions: *instantiation abstraction* and *deletion abstraction*. In the case of instantiation abstraction, abstraction function generates ground instances of the concrete axioms as it is done in the Inst-Gen framework [7]. In the case of deletion abstraction we delete certain concrete axioms from the theory. This abstraction can be used to incorporate other axioms selection methods into this framework, which are based on removing irrelevant axioms. In practice, these abstractions can be recombined.

4.2 Strengthening Abstraction Refinement

In the case of deletion abstraction, refinement can be done by adding concrete axioms $\breve{A}$ that turn the model I, which is obtained form ATP_S, into a countermodel, $\breve{A} \subseteq \{\breve{a} \mid \breve{a} \in A, I \not\models \breve{a}\}$. In the case of instantiation abstraction, refinement can be done by generating a set of ground instances of axioms $A\sigma$ such that $I \not\models A\sigma$, $\breve{A} := A\sigma$.

5 Combined-Approximation

Currently we are working on combining the two approximations discussed above. This combination has the purpose of converging to a proof more rapidly by over and under approximating. One approach to combine them is using under-approximation in the outer-loop and over-approximation in the place of ATP_S. Let us note that this allows us incorporate other axiom selection methods [11, 6, 13, 14] as part of the under-approximation abstraction.

References

[1] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theor. Comput. Sci.*, 236(1-2):133–178, 2000.

[2] Peter Baumgartner and Cesare Tinelli. The model evolution calculus as a first-order DPLL method. *Artif. Intell.*, 172(4-5):591–632, 2008.

[3] Maria Paola Bonacina, Christopher Lynch, and Leonardo Mendonça de Moura. On deciding satisfiability by theorem proving with speculative inferences. *J. Autom. Reasoning*, 47(2):161–189, 2011.

[4] Randal E Bryant, Daniel Kroening, Joël Ouaknine, Sanjit A Seshia, Ofer Strichman, and Bryan Brady. Deciding Bit-Vector Arithmetic with Abstraction. *Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2007. Lecture Notes in Computer Science*, 4424:358–372, 2007.

[5] Edmund M Clarke, Orna Grumberg, and David E Long. Model checking and abstraction. *ACM transactions on Programming Languages and Systems (TOPLAS)*, 16(5):1512–1542, 1994.

[6] Krystof Hoder and Andrei Voronkov. Sine qua non for large theory reasoning. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings*, volume 6803 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2011.

[7] Konstantin Korovin. Inst-Gen – A Modular Approach to Instantiation-Based Automated Reasoning. In Andrei Voronkov and Christoph Weidenbach, editors, *Programming Logics: Essays in Memory of Harald Ganzinger*, volume 7797 of *Lecture Notes in Computer Science*, pages 239–270. Springer Berlin Heidelberg, 2013.

[8] Keiichirou Kusakari, Masaki Nakamura, and Yoshihito Toyama. Argument filtering transformation. In *Principles and Practice of Declarative Programming, International Conference PPDP'99, Paris, France, September 29 - October 1, 1999, Proceedings*, volume 1702 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 1999.

[9] Christopher Lynch. Unsound Theorem Proving. *Computer Science Logic. CSL 2004. Lecture Notes in Computer Science*, 3210:473–487, 2004.

[10] David A. Plaisted. Theorem proving with abstraction. *Artif. Intell.*, 16(1):47–108, 1981.

[11] Geoff Sutcliffe and Yury Puzis. SRASS - A Semantic Relevance Axiom Selection System. In *International Conference on Automated Deduction, CADE-21*, volume 4603 of *LNCS*, pages 295–310, 2007.

[12] Andreas Teucke and Christoph Weidenbach. First-order logic theorem proving and model building via approximation and instantiation. In Carsten Lutz and Silvio Ranise, editors, *Frontiers of Combining Systems - 10th International Symposium, FroCoS 2015, Wroclaw, Poland, September 21-24, 2015. Proceedings*, volume 9322 of *Lecture Notes in Computer Science*, pages 85–100. Springer, 2015.

[13] Josef Urban. MaLARea: A metasystem for automated reasoning in large theories. *CEUR Workshop Proceedings*, 257:45–58, 2007.

[14] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jirí Vyskocil. Malarea SG1- machine learner for automated reasoning with semantic guidance. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2008.