TOWARDS AN INCREMENTAL REENTRANT ASP-BASED SOLVER FOR ARGUMENTATION FRAMEWORKS

KAMILLO HUGH FERRY

Abstract. We devise a incremental ASP-based solving system for Dung's abstract argumentation frameworks. This means that we allow for dynamic changes to an input AF such as adding or removing arguments and attacks while retaining the state of the underlying grounder and solver.

This paper documents the design process and current state of the incremental solver system. Also, we details some of the problems that were encountered and open issues of the cosntructed system.

1. Introduction

The need for incremental algorithms arises when dealing with dynamically changing inputs where it is desirable to process each such change relatively quickly and where it is not desirable to process the input every time from scratch. One example for such a setting are knowledge bases, and in particular, abstract argumentation frameworks (AFs) that model a knowledge base.

Here, given an AF, we might have computed a set of possible extensions satisfying a certain semantic. But when we edit this AF, e. g. to react to a change in our knowledge base, and ask for the extensions of the resulting framework, we can try to keep information from previous solve iterations so we do not have to construct all extensions from scratch.

This approach to algorithmic problems trying to retain information between the runs of an algorithm and incrementally solving a dynamically changing instance has been studied in literature before [1, 2, 3].

An example taken from Patnaik and Immerman [3] (Example 3.2), when asking for the parity of a string of n bits, if we retain the parity bit between calculations, we can actually express the answer using propositional formulas when the allowed edits are flipping bits.

That is, suppose we have a data structure S representing the state of a string of n bits called S_x, and the parity bit b of this string. We

$$\mathsf{ins}(S,a): \quad \begin{aligned} S'_x &:= S_x \vee x = a \\ b' &:= (b \wedge S_a) \vee (\neg b \wedge \neg S_a) \end{aligned}$$

$$\mathsf{del}(S,a): \quad \begin{aligned} S'_x &:= S_x \wedge x \neq a \\ b' &:= (b \wedge \neg S_a) \vee (\neg b \wedge S_a) \end{aligned}$$

FIGURE (1). Operations ins and del for calculation of parity bits

can define two operations $\mathsf{ins}(S,i)$ and $\mathsf{del}(S,i)$ that represent 'setting the i-th bit to 0 resp. 1' that edit S accordingly to the changes that were requested and those operations can be expressed by propositional formulae as in fig. 1.

The main idea behind this example is that given a specific problem, we might be able to identify how a change to the input affects the solution and what kind of intermediate information has to be kept during each computation.

Applying this to abstract argumentation frameworks, we build upon the ASP-based argumentation system ASPARTIX [**aspartix**] which already provides encodings for most common semantics of argumentation frameworks, albeit static ones.

ASPARTIX itself is implemented in the ASP system *clingo*, whose API for partial grounding and externally provided atoms is also heavily used to enable editing the input AF.

This is done by breaking apart the given encodings into parts that are affected by the insertion of an argument and parts that are affected by the insertion of an attack. Whenever an argument or attack is to be added, the respective partial encoding is applied.

The detailed process of modifying the encodings and

References

[1] Bernard Mans and Luke Mathieson. "Incremental Problems in the Parameterized Complexity Setting". In: *Theory of Computing Systems* 60.1 (), pp. 3–19. ISSN: 1432-4350, 1433-0490. DOI: 10.1007/s00224-016-9729-6 (cit. on p. 1).

[2] Peter Bro Miltersen et al. "Complexity models for incremental computation". In: *Theoretical Computer Science* 130.1 (), pp. 203–236. ISSN: 03043975. DOI: 10.1016/0304-3975(94)90159-7 (cit. on p. 1).

[3] Sushant Patnaik and Neil Immerman. "Dyn-FO: A Parallel, Dynamic Complexity Class". In: *Journal of Computer and System Sciences* 55.2 (), pp. 199–209. ISSN: 00220000. DOI: 10.1006/jcss.1997.1520 (cit. on p. 1).