GRIS: Computing traditional argumentation semantics through numerical iterations.

Odinaldo Rodrigues

Department of Informatics,
King's College London,
The Strand, London, WC2R 2LS, UK
odinaldo.rodrigues@kcl.ac.uk

Abstract. This paper provides an outline of the Gabbay-Rodrigues Iterative Solver (GRIS). The solver can be used in decision and enumeration problems of the grounded and preferred semantics.

1 Introduction

The Gabbay-Rodrigues Iterative Solver (GRIS) is implemented in C++ and uses the Gabbay-Rodrigues Iteration Schema (see below) in the computation of the solution to the following classes of problems. Given an argumentation network $\langle S, R \rangle$: to produce one or all of the extensions of the network under the grounded and preferred semantics; and given an argument $X \in S$ to decide whether X is accepted credulously or sceptically according to one of these two semantics. Problems to GRIS must be submitted according to `probo`'s syntax (see [2]).

GRIS is not currently able to handle the complete and stable semantics although their implementation would not pose any additional technical difficulties on top of what is already available.

2 Theoretical underpinnings

GRIS works with numerical argumentation networks where arguments are given initial values from $[0, 1]$ from which equilibrium values are calculated iteratively yielding traditional extensions via the following correspondence.

Definition 1 (Caminada/Gabbay-Rodrigues Translation). *A labelling function λ and a valuation function V can be inter-defined by identifying the value* 1 *with the label* **in**, *the value* 0 *with the label* **out** *and assigning any node with a value in* $(0, 1)$ *with the label* **und** *and assigning a node with the label* **und** *with a suitable value in* $(0, 1)$, *e.g.,* 0.5.

The Gabbay-Rodrigues Iteration Schema. Let $\langle S, R \rangle$ be an argumentation network and V_0 be an initial assignment of values from $[0, 1]$ to the nodes in S. Let for each $X \in S$, $MA_i(X) = \max_{Y \in Att(X)}\{V_i(Y)\}$ and the equation below define the value of X at the subsequent iterations (i.e., $V_1, V_2, \ldots$):

$$V_{i+1}(X) = (1 - V_i(X)) \cdot \min\left\{\frac{1}{2}, 1 - MA_i(X)\right\} + V_i(X) \cdot \max\left\{\frac{1}{2}, 1 - MA_i(X)\right\}$$

The set of all such equations constitutes the argumentation network's *GR system of equations* [3]. For each $X \in S$, the sequence of values $V_0(X)$, $V_1(X)$, $\ldots$, etc, converges and the value $V_e(X) = \lim_{i\to\infty} V_i(X)$ is defined as the *equilibrium value* of the node X. For any assignment (resp., labelling function) f, let $in(f) = \{X \in \text{dom } f \mid f(X) = 1 \text{ (resp., } \mathbf{in})\}$ and $out(f) = \{X \in \text{dom } f \mid f(X) = 0 \text{ (resp., } \mathbf{out})\}$. Caminada and Pigozzi have shown that any labelling function λ for an argumentation framework $\langle S, R\rangle$ can be turned into an admissible labelling function λ_{da} via a sequence of contraction operations that successively turn nodes illegally labelled **in** or **out** by λ into **und**. Furthermore, each admissible labelling function λ_{da} can be turned into a complete labelling λ' by a sequence of expansion operations that successively turn nodes illegally labelled **und** by λ_{da} into **in** or **out** as appropriate. Since an expansion sequence stops when no **und** nodes remain illegally labelled, it follows that λ' is the minimal complete labelling such that $in(\lambda_{da}) \subseteq in(\lambda')$ and $out(\lambda_{da}) \subseteq out(\lambda')$ [1].

The Gabbay-Rodrigues Iteration Schema defined according to the equation above produces an equivalent result in a numerical context, except that no contractions or expansions are needed and the labelling λ' can be obtained from the equilibrium values of the sequences (see [3] for details). A labelling function λ such that for all $X \in S$, $\lambda(X) = \mathbf{und}$ is, by definition, admissible, since it does not label any nodes **in** or **out**. When an expansion sequence is then applied to λ resulting in the labelling function λ',[1] as a result λ' will correspond to the smallest complete labelling for $\langle S, R\rangle$. The set $in(\lambda') = \{X \in S \mid \lambda'(X) = \mathbf{in}\}$ will then correspond to the grounded extension of $\langle S, R\rangle$. In the numerical case, all we have to do is to assign all nodes with a initial value V_0 in $(0, 1)$ (we use 0.5); compute their equilibrium values; and then define the grounded extension of $\langle S, R\rangle$ as the set of nodes with equilibrium value 1.

The above ideas constitute the basis of the GR Grounder module (see Fig. 1). In addition, the grounder can also be applied to a subnetwork of particular interest with optional given "conditioning" values during the course of a computation. The conditioning values are calculated in a previous step, and hence fixed, and simply fed into the equations. This allows the result of an individual strongly connected component (SCC) of the network to be computed provided the values of all of its conditioning nodes are known. Since non-trivial SCCs in a given layer are independent of each other, the computation of the results within a layer can *potentially* be done in parallel (a possible enhancement to the solver). Results of the SCCs are then carefully combined to provide answers to the decision and enumeration problems with respect to the required semantics.

One question of interest is of course how to determine the equilibrium values in a finite number of computations. This is done by *approximation.* Corollary 2.38 in [3] shows that all values converge to one of the values in $\{0, \frac{1}{2}, 1\}$. The grounder stops iterating as soon as the maximum variation in node values between two successive iterations is smaller than or equal to the upper bound of the relative error introduced due to rounding in the calculations of the target machine. In

[1] Caminada and Pigozzi have shown that the set of complete labellings that are "bigger or equal" to a given labelling function has a unique smallest element [1, Theorem 11].

our 64-bit computer, this is 10^{-19}. It is envisaged that the computation may be stopped much earlier if we can safely detect the convergence of all nodes to one of the values in $\{0, \frac{1}{2}, 1\}$. As it turns out, this is not the main bottleneck in the computation, as grounding can be computed relatively quickly even for larger networks. The computational complexitity (and GRIS' current inability to complete the calculation of solutions for some classes of problems) mainly arises because of the management of multiple candidate assignments, which increase in proportion to the number and size of the SCCs and the number of network layers. This aspect of GRIS can be improved in a number of different ways.

3 System Overview

GRIS initially reads the problem specification passed as command line arguments, validates it and then validates the input network, exiting in case of error.

The basic workflow for the computations involving the grounded and preferred semantics is depicted in Fig. 1, with the exception that in the grounded semantics, intermediate preferred solutions do not need to be generated and so the *preferred solutions generator* and *partial preferred solutions* datastore are not used. The solver starts by computing the strongly connected components of the network using a specially modified version of Tarjan's algorithm [6] and arranging them into layers that can be used in successive computation steps as described in [4]. Once the layers are computed, the solver can identify the deepest layer level of computation needed according to the layer depth of the input argument and this can be used to terminate the computation of decision problems as early as possible.

For the computation of the solutions to the problems in the grounded semantics, the decomposition into layers is not strictly necessary. The GR Grounder can in principle be applied to the entire network at once and the nodes with equilibrium value 1 will correspond to the grounded extension of the entire network. However, since the computation of the SCCs and their arrangement into layers can be performed very efficiently with our version of the algorithms, this extra cost is offset by gains obtained through the computation by layers in all but a few special cases. Our strategy is then to feed the result of each layer into the next layer's computation until either all layers are computed or we reach the maximum depth needed to establish the membership (or not) of the argument in the grounded extension (this strategy proves particularly efficient if the argument belongs to a layer of low depth).

Preferred Semantics. The solution of problems involving the preferred semantics involves the computation of partial network solutions to each layer. The key point in GRIS is that once the grounder is invoked for a particular SCC (using all required conditioning values), the resulting equilibrium values may contain undecided values, some of which could potentially be assigned the value 1 in a preferred extension. So our (naive) implementation assigns the value 1 to all such nodes and then corrects illegal values in a manner similar to that of the Modgil and Caminada's labelling-based algorithms [5]. The results of every candidate solution are then propagated to the next layer using the grounder again and the whole process repeats. There is ample scope for optimisation here since undecided nodes that are attacked by a conditioning undecided argument cannot possibly be **in** in any extension. Furthermore, in decision problems, a careful analysis of the argument involved may identify partial solutions of particular interest without the need to generate all partial solutions (the blind generation of partial solutions can quickly exhaust resources).

Solutions to the problems in the complete and stable semantics may also be computed with appropriate modifications, but these mechanisms have not been implemented in this version of the solver.

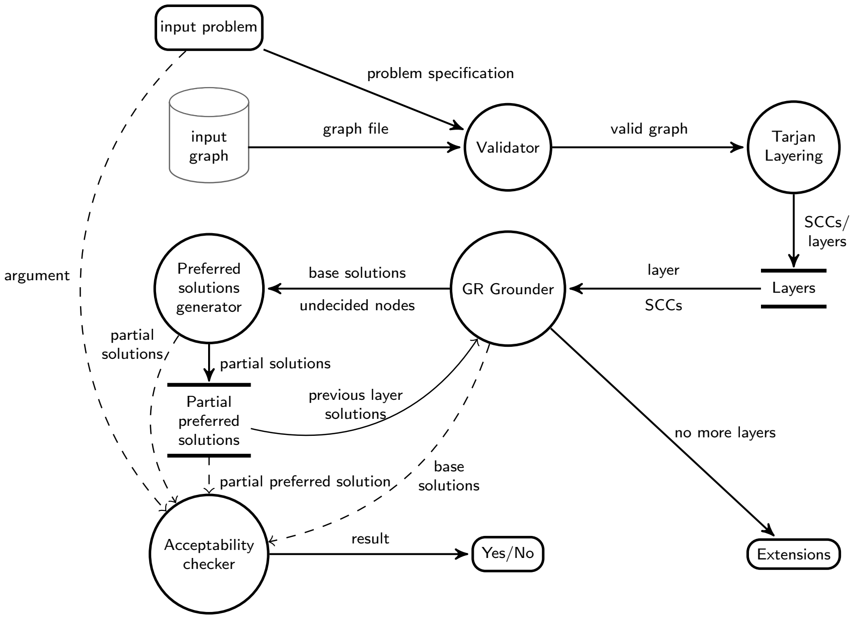

Fig. 1. Basic workflow of the preferred semantics calculator.

Obtaining GRIS: `http://www.inf.kcl.ac.uk/staff/odinaldo/gris`

References

1. M. Caminada and G. Pigozzi. On judgment aggregation in abstract argumentation. *Autonomous Agents and Multi-Agent Systems*, 22(1):64–102, 2011.
2. F. Cerutti, N. Oren, H. Strasse, M. Thimm, and M. Vallati. The first international competition on computational models of argumentation (ICCMA15). `http://argumentationcompetition.org/2015/index.html`, 2015.
3. D. M. Gabbay and O. Rodrigues. Equilibrium states in numerical argumentation networks. *Logica Universalis*, pages 1–63, 2015.
4. B. Liao. *Efficient Computation of Argumentation Semantics*. Elsevier, 2014.
5. S. Modgil and M. Caminada. Proof theories and algorithms for abstract argumentation frameworks. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 105–129. Springer US, 2009.
6. R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.