Proving Program Correctness Using the AG Semantics: An Example with n-Queens

Amelia Harrison

May 22, 2018

Abstract

We show an example of how the proposal for the semantics of the subset of the input language of GRINGO called AG can be used as the basis for a proof of correctness of a program encoding the n-queens problem.

Gebser et al. (2015) proposed a semantics for a large subset of the input language of the popular ASP grounder GRINGO. That subset is called AG, short for Abstract Gringo, because it uses an abstract syntax. The semantics is based on the definition of stable models for infinitary propositional formulas (Truszczynski, 2012), which are formulas with conjunctions and disjunctions over infinite sets. It is defined in terms of a translation function τ that transforms ASP programs into sets of infinitary propositional formulas.

One use of the semantics is as a tool for proving the correctness of programs written in the GRINGO input language. To illustrate such a proof, let's consider the program in Table 1. Here we give a broad strokes overview of the proof of correctness of that program presented by Gebser et al.(2015). This program (which we will call program K) is an encoding for the n-queens problem: how can n queens be placed on an $n \times n$ chess board such that no two queens threaten each other?

To formally state what it means to be a solution to this problem, we can represent squares by pairs of integers (i, j) where $1 \leq i, j \leq n$. Two squares (i_1, j_1) and (i_2, j_2) are said to be in the same row if $i_1 = i_2$, in the same column if $j_1 = j_2$, and in the same diagonal if $|i_1 - i_2| = |j_1 - j_2|$. Then a set Q of n squares is a *solution* to the n-queens problem if no two elements of Q are in the same row, in the same column, or in the same diagonal.

We identify an atom of the form $q(i, j)$ with the square (i, j). Now the claim of the correctness of program K can be stated formally and succinctly:

> A set of squares is a solution to the n-queens problem iff it is a stable model of K.

% place queens on the chess board	
$\{q(1..n, 1..n)\}$	R_1
% exactly 1 queen per row/column	
$\leftarrow X = 1..n \wedge \textit{not count}\{Y : q(X, Y)\} = 1$	R_2
$\leftarrow Y = 1..n \wedge \textit{not count}\{X : q(X, Y)\} = 1$	R_3
% at most one queen per diagonal	
$\leftarrow D = 1..\ n\times 2-1 \wedge \textit{not count}\{X, Y{:}q(X, Y), D = X - Y + n\} \leq 1$	R_4
$\leftarrow D = 1..\ n\times 2-1 \wedge \textit{not count}\{X, Y{:}q(X, Y), D = X + Y - 1\} \leq 1$	R_5

Table 1: Program K: An encoding of the n-queens problem.

To prove the claim using the AG semantics, the first step is to calculate the translation of program K as a set of infinitary formulas. For example, the set

$$\left\{ \bigwedge_{1 \leq i,j \leq n} (q(i, j) \vee \neg q(i, j)) \right\}$$

is the result of applying τ to rule R_1 from Table 1. It is relatively easy to verify, using the definition of stable models for infinitary formulas, that the stable models of this formula alone correspond to arbitrary subsets of the squares in an n by n board. On the other hand, the result of applying τ to rules $R2$–$R5$ is a set of *constraints*, that is to say, infinitary formulas of the form $\neg F$. Then the set τK can be partitioned into two parts: one which contains only constraints, and the other whose stable models we've already described.

The following proposition, then will help to prove the claim. It is a straightforward generalization of Proposition 4 from Ferraris and Lifschitz (2005). Its proof is a straightforward generalization of that proof.

Proposition 1 *Let $\mathcal{H}_1$ be an infinitary formula $\mathcal{H}_2$ be a set of constraints. A set I of atoms is a stable model of $\mathcal{H}_1 \cup \mathcal{H}_2$ iff I is a stable model of $\mathcal{H}_1$ and satisfies all formulas in $\mathcal{H}_2$.*

To prove the claim based on the proposition, we treat τR_1 as $\mathcal{H}_1$ and we treat the result of applying τ to all other rules in K as $\mathcal{H}_2$. We know that stable models of τR_1 correspond to arbitrary sets of squares on an $n \times n$ board. Then it suffices to show that an interpretation I satisfies τR_2 iff no two squares in I are in the same row, satisfies τR_3 iff no two squares in I are in the same column, and satisfies $\tau R_4 \cup \tau R_5$ iff no two squares in I are in the same diagonal. Each

of these facts is relatively easy to establish based on the form of each of these sets of constraints.

References

Ferraris, P. and Lifschitz, V. (2005). Mathematical foundations of answer set programming. In *We Will Show Them! Essays in Honour of Dov Gabbay*, pages 615–664. King's College Publications.

Gebser, M., Harrison, A., Kaminski, R., Lifschitz, V., and Schaub, T. (2015). Abstract Gringo. *Theory and Practice of Logic Programming*, 15:449–463.

Truszczynski, M. (2012). Connecting first-order ASP and the logic FO(ID) through reducts. In Erdem, E., Lee, J., Lierler, Y., and Pearce, D., editors, *Correct Reasoning: Essays on Logic-Based AI in Honor of Vladimir Lifschitz*, pages 543–559. Springer.