Assignment 7 (individual): Answer Set Programming

Fall 2012, CSCI 4980/8986

Due: November 7th

1 Answer Set Programming

Answer set programming (ASP) [7, 8] is a declarative programming formalism based on the answer set semantics of logic programs [3, 4]. The idea of ASP is to represent a given computational problem by a program whose answer sets correspond to solutions, and then use an answer set solver to generate answer sets for this program.

In this assignment we will use the answer set system CLINGO\(^1\) that incorporates answer set solver CLASP [2] with its front-end (grinder) GRINGO (user guide [1]). System CLINGO is currently one of the most widely used answer set solvers.

A common methodology to solve a problem in ASP is to design GENERATE, DEFINE, and TEST [6] parts of a program. The GENERATE part defines a large collection of answer sets that could be seen as potential solutions. The TEST part consists of rules that eliminate the answer sets of the GENERATE part that do not correspond to solutions. The DEFINE section expresses additional concepts and connects the GENERATE and TEST parts.

A typical logic programming rule has the form of a Prolog rule. For instance, program

\[
\begin{align*}
  p. \\
  q & \leftarrow p, \text{not } r.
\end{align*}
\]

is composed of such rules. This program has one answer set \(\{p, q\}\). In addition to Prolog rules, GRINGO also accepts rules of other kinds — “choice rules” and “constraints”. For example, rule

\[
\{p, q, r\}.
\]

is a choice rule. Answer sets of this one-rule program are arbitrary subsets of the atoms \(p, q, r\). Choice rules are typically the main members of the GENERATE part of the program. Constraints often form the TEST section of a program. Syntactically, a constraint is the rule with an empty head. It encodes the conditions on the answer sets that have to be met. For instance, the constraint

\[
\leftarrow p, \text{not } q.
\]

eliminates the answer sets of a program that include \(p\) and do not include \(q\).

System GRINGO allows the user to specify large programs in a compact way, using rules with schematic variables and other abbreviations. A detailed description of its input language can be found in the online manual [1]. Grounder GRINGO takes a program “with abbreviations” as an input and produces its propositional counterpart that is then processed by CLASP. The system CLINGO can be used as a shortcut for invoking both of these systems at once.

2 Example: \(N\)-Queens

The goal is to place \(n\) queens on an \(n \times n\) chessboard so that no two queens would be placed on the same row, column, and diagonal. A solution can be described by a set of atoms of the form \(q(i, j) (1 \leq i, j \leq n)\):

\(^1\text{http://potassco.sourceforge.net/}\)
including \( q(i, j) \) in the set indicates that there is a queen at position \((i, j)\). A solution is a set \( X \) satisfying the following conditions:

1. the cardinality of \( X \) is \( n \),
2. \( X \) does not contain a pair of different atoms of the form \( q(i, j), q(i', j) \) (two queens on the same row),
3. \( X \) does not contain a pair of different atoms of the form \( q(i, j), q(i, j') \) (two queens on the same column),
4. \( X \) does not contain a pair of different atoms of the form \( q(i, j), q(i', j') \) with \(|i' - i| = |j' - j|\) (two queens on the same diagonal).

Here is the representation of this program in the input language of CLINGO/GRINGO:

```
number(1..n).
#domain number(I).
#domain number(I1).
#domain number(J).
#domain number(J1).

%Condition 1 and 2
1{q(K,J): number(K)}1.

%Condition 3
:-q(I,J), q(I,J1), J<J1.

%Condition 4
:-q(I,J), q(I1,J1), J<J1, #abs(I1-I)==J1-J.
```

The command line

```
clingo -c n=4 queens.gr
```

instructs the answer set system CLINGO to find a single solution for 4-queens problem. Whereas the command line

```
clingo -c n=8 queens.gr 0
```

instructs CLINGO to find all solutions for 8-queens problem. An extract from the output of the last command line follows

```
... 
Answer: 92
number(1) number(2) number(3) number(4) number(5) number(6) number(7) number(8)
q(5,8) q(7,7) q(2,6) q(6,5) q(3,4) q(1,3) q(8,2) q(4,1)
SATISFIABLE
```

This ninety second solution found by the solver encodes the following valid configuration of queens on the board

```
1 2 3 4 5 6 7 8
1 Q
2 Q
3 Q
4Q
5 Q
6 Q
7 Q
8 Q
```
**Problem 1.** Use CLINGO to find all solutions to the 8 queens problem that (a) have a queen at (1,1); (b) have no queens in the $4 \times 4$ square in the middle of the board.

**Problem 2.** Report times used by GRINGO, CLASP individually to process a program to find solutions for 20, 40, 60, 80, 100, 120, 140 queens problem.

**Acknowledgments**

The introduction to answer set programming followed the lines of [5]. The statement of $N$-queens problem follows the lecture notes on *Methodology of Answer Set Programming; course Answer set programming: CS395T, Spring 2005* by Vladimir Lifschitz.

**References**


__http://www.cs.utexas.edu/~vl/teaching/asp.html__