Appendix A
ASP Solver Quick-Start

What follows is a very brief, operational introduction to two currently existing ASP solvers, `clingo` and DLV. Since the field is developing rapidly, we recommend that users of this information learn about the most current versions of these solvers. To find DLV, go to `http://www.dlvsystem.com`. To find `clingo`, go to `http://potassco.sourceforge.net/`. For quick access to the manuals, just search online for `DLV manual` or `clingo manual`.

To find all answer sets of a given program, type

```
clingo 0 program_name
```

or

```
dlv -n=0 program_name
```

The `0` tells the programs to return *all* answer sets. If you omit the parameter when calling `clingo`, the program will return only one answer set; DLV will return all answer sets. Changing the number will return the corresponding number of answer sets. Both systems use `-` and `:-` instead of $\neg$ and $\leftarrow$, respectively. Epistemic disjunction *or* is denoted by `|`.

Often we may want to limit what a solver will output when it prints answer sets because complete sets can be large and we may only be interested in a few predicates. When using `clingo`, it is useful to learn the `#show` commands. For example, if you had a program with predicate $mother(X, Y)$, and you included the following line in your program:

```
#show mother/2.
```

`clingo` would output only those atoms of the answer sets that are formed by predicate $mother$. Adding

```
#show -mother/2
```

will also yield all occurrences of negative literals formed by predicate $mother$.

When using DLV, a similar (but not identical) effect can be produced with command-line options:

```
dlv -filter=mother dlvfamily.lp
dlv -pfilter=mother dlvfamily.lp
```

The first will return all the literals formed by predicate $mother$; the second will only return atoms formed by this predicate.

There is also a useful program called `mkatoms`, which can be found at `http://www.mbal.tk/mkatoms/`. It formats the output of the solvers with one predicate per line. Simply pipe the output of the solver to it.

Here is a small example:

```
%% program basic_test.lp
p(a).
-q(a).
r(a) | -r(a).
```

Running this program with `clingo`

```
clingo 0 basic_test.lp | mkatoms
```

or with DLV

```
dlv basic_test.lp | mkatoms
```

will give two answer sets as follows:

```
p(a)
-q(a)
-r(a)
::endmodel
p(a)
-q(a)
r(a)
::endmodel
```

For programs with a comparatively small number of answer sets, DLV and `clingo` can be used to answer queries by simply computing the program's answer sets and using Definition 2.2.2 from Section 2.2.1. For instance, the answer to query $?p(a)$ to the above program should be $true$ because it is true in both answer sets, the answer to query $?q(a)$ should be $false$ because it is false in both answer sets, and the answer to query $?r(a)$ should be $unknown$ because it is true in one but not in the other.