Declarative Programming:
Lecture Notes and Exercises

Part 3

In theoretical papers, such as Abstract Gringo, programs are often written using symbols different from the ASCII characters that we see in clingo input files. For instance, instead of the string :- that separates the head of a rule from its body, in theoretical notation we use the left arrow (←). To separate members of the body of a rule, instead of commas we use the conjunction symbol (∧). And there is no need to put a period at the end of the rule when we use theoretical notation. For example, the expression

\[
\text{warm}(C) \leftarrow t(C, T1) \land t(austin, T2) \land T1 > T2
\]

is the rule

\[
\text{warm}(C) :- t(C,T1), t(austin,T2), T1>T2.
\]

as it may be written in a theoretical paper.

When a rule containing integers is written in theoretical notation, integers will be overlined, for instance:

\[
t(austin, \overline{88}).
\]

Overlined integers will be called numerals.

Many formulas can be thought of as rules written in theoretical notation. For instance, formulas (1)–(4) from Part 2 of these lecture notes correspond to the rules

\[
p.
\]
\[
q :- p, r.
\]
\[
r :- p.
\]
\[
s :- r, t.
\]

If we give these rules as input to CLINGO, its output will represent the minimal model of these formulas:

Solving...
Answer: 1
p r q
The formulas from Problem 2.1

\[ p \lor q, \ r \leftarrow p, \ s \leftarrow q \]

can be written as input for CLINGO in the following way:

\[
\begin{align*}
& p, \ q. \\
& r :- p. \\
& s :- q.
\end{align*}
\]

Note that the comma in the first line corresponds to disjunction (\(\lor\)), not conjunction. CLINGO interprets a comma as \(\land\) when it separates members of the body of a rule, and as \(\lor\) when it separates members of the head. Given this input, CLINGO will solve Problem 2.1:

Solving...
Answer: 1
s q
Answer: 2
r p

These examples illustrate the following general observation: a formula represents a CLINGO rule if it is an atom, or a disjunction of atoms

\[ A_1 \lor \cdots \lor A_n, \]  \hspace{1cm} (1)

or an implication of the form

\[ A_1 \lor \cdots \lor A_m \leftarrow A_{m+1} \land \cdots \land A_n \]  \hspace{1cm} (2)

\((n > m > 0)\) where each \(A_i\) is an atom. Formulas of the form

\[ \bot \leftarrow A_1 \land \cdots \land A_n \]  \hspace{1cm} (3)

represent CLINGO rules as well; \(\bot\) is the symbol \#false written in theoretical notation. Formula (3) is equivalent to

\[ \neg (A_1 \land \cdots \land A_n). \]

When the head of a rule is \(\bot\), it can be dropped without changing the meaning of the rule.

Formulas of the forms (1)–(3) will be called easy. CLINGO is designed in such a way that given a finite set of easy formulas, it generates all minimal models of that set.
3.1. Determine which of the given formulas are easy.

(a) \( p \land q \leftarrow r \land s \),
(b) \( p \lor q \leftarrow r \lor s \),
(c) \( p \lor \neg q \).

For each of the formulas that are not easy, find an equivalent easy formula or set of easy formulas.

The relationship between CLINGO and minimal models can be sometimes used to predict the output that CLINGO is going to produce.

3.2. For each of the given programs, predict what answers CLINGO is going to produce if this program is given to it as input. Justify your predictions.

(a)

\[ p. \]
\[ :- p, q. \]

(b)

\[ p, q, r. \]
\[ s :- p, q. \]

(c)

\[ p, q :- r. \]
\[ q, r :- s. \]
\[ r, s :- p. \]
\[ s, p :- q. \]

This method is applicable not only when the atoms \( A_i \) in (1)–(3) are symbolic constants, but also when they have the form \( p(t_1, \ldots, t_k) \), where each \( t_i \) is a precomputed term. For the definition of a precomputed term, see Section 2.1 of the paper Abstract Gringo.

3.3. Determine whether the terms

\[ f(5, -3) \] and \( f(5, -\overline{3}) \]

are precomputed.

3.4. Predict what answers CLINGO is going to produce for the program

\[ p(1). \]
\[ p(2) :- p(0), p(1). \]