4. Choice Formulas, Constraints, and Cardinality Expressions

The art of answer set programming is based on the possibility of representing the collection of sets that we are interested in as the collection of answer sets of a logic program. This is often achieved by combining rules of the following kinds, studied in this handout. A "choice formula" is a formula with many answer sets that provides an approximation from above for the collection of sets that we want to describe. The redundant answer sets can be removed by adding "constraints" to this program. A cardinality expression is a construct for representing the cardinality of a set, which essentially combines choice formulas and constraints.

Choice Formulas

For any finite set Z of atoms, by Z^c we denote the formula

$$\bigwedge_{A \in Z} (A \vee \neg A)$$

4.1 Prove the following statement: A set X is an answer set of Z^c iff X is a subset of Z.

Thus if Z consists of n atoms then Z^c has 2^n answer sets. Under the answer set semantics, Z^c says: choose for every element of Z arbitrarily whether to include it in the answer set. We will call formulas of the form Z^c *choice formulas*. (The superscript ^{c} is used in this notation because it is the first letter of the word "choice.")

Although LPARSE does not allow us to use conjunctions and disjunctions in the heads of rules, it does understand choice formulas as heads, with the superscript ^{c} dropped. For instance, we can write

```
{p,q}
```

for

$$(p \vee \neg p) \wedge (q \vee \neg q)$$

and

```
{p} :- q
```

for

$$p \vee \neg p \leftarrow q.$$

4.2 (a) Consider the program

$$\begin{array}{l} \{p, q, r\}^c \\ s \leftarrow p, \neg q. \end{array}$$

What do you think its answer sets are? Use SMODELS to verify your conjecture. (b) Do the same for the program

$$\begin{array}{l} \{p, q\}^c \\ \{r, s\}^c \leftarrow \neg p. \end{array}$$

If the head of a rule is a choice formula Z^c for a large set Z then it may be possible to represent the rule in the language of LPARSE concisely using variables (see Handout 2 on the use of variables in LPARSE). For instance, the rule

$$\{p_1, \dots, p_7\}^c \leftarrow q \tag{1}$$

can be encoded as follows:

```
index(1..7).
{p(I) : index(I)} :- q.
```

Note how the "local" use of `I` in this example differs from the "global" use of variables in Handout 2 and in the following example:

```
index(1..7).
{p(I)} :- q, index(I).
```

The last two lines correspond to the set of 7 rules:

$$\{p_i\}^c \leftarrow q \qquad (1 \leq i \leq 7). \tag{2}$$

In this example, the difference between local and global variables is not essential, however: we will see in a later handout that replacing (1) with (2) in any program does not affect the program's answer sets.

Constraints

4.3 A set of atoms is an answer set of $F \wedge \neg G$ iff it is an answer set of F and does not satisfy G.

We observed earlier that a conjunction $F \wedge G$ may have answer sets that are not found among the answer sets of F. The assertion of Problem 4.3 shows, however, that this cannot happen if the second conjunctive term begins with negation. Conjoining a formula with $\neg G$ leads only to the loss of answer sets—of all those that do not satisfy G.

Formulas beginning with negation are called *constraints*. In the language of LPARSE, constraints are written as rules with the empty head. For instance, constraint

$$\leftarrow p$$

is represented in LPARSE as

```
        :- p.
```

4.4 Find the answer sets of the program

$$\begin{array}{l} \{p, q, r\}^c \\ \leftarrow p, q, \neg r \end{array}$$

(a) using properties of choice formulas and constraints; (b) using SMODELS.

The combination of choice rules and constraints yields a way to embed propositional logic into the answer set semantics as follows.

4.5 For any propositional formula F, a set X of atoms is a model of F iff X is an answer set of $Z^c \wedge \neg\neg F$ where Z is the set of all atoms occurring in F.

Cardinality Expressions

Constraints used in ASP programs often involve conditions on the cardinality of a set of atoms. We will introduce special notation for such formulas.

For any finite set Z of literals and any nonnegative integer l ("lower bound"), by $l \leq Z$ we denote the disjunction of the formulas $\bigwedge_{L \in Y} L$ over all l-element subsets Y of Z. For instance,

$$2 \leq \{p, q, r\}$$

stands for

$$(p \wedge q) \vee (p \wedge r) \vee (q \wedge r).$$

Clearly, if the elements of Z are atoms then a subset X of Z satisfies $l \leq Z$ iff the cardinality of X is at least l.

By $Z \leq u$, where u is a nonnegative integer ("upper bound"), we denote the formula $\neg(u + 1 \leq Z)$. Finally, $l \leq Z \leq u$ stands for

$$(l \leq Z) \wedge (Z \leq u).$$

As an example of the use of cardinality expressions in constraints, consider the program

$$\begin{array}{l} \{p_1, \dots, p_n\}^c \\ \leftarrow \{p_1, \dots, p_n\} \leq 0 \\ \leftarrow 2 \leq \{p_1, \dots, p_n\}. \end{array} \tag{3}$$

The constraints eliminate two kinds of sets from the collection of answer sets of the choice rule: the empty set and the sets that have at least 2 elements. Consequently, the answer sets of (3) are the singletons $\{p_1\}, \dots, \{p_n\}$.

The language of LPARSE allows us to use cardinality expressions

$$l \leq Z,\ Z \leq u,\ l \leq Z \leq u$$

in the bodies of rules, with the sign $\leq$ dropped. For instance, program (3) can be written as

```
index(1..n).

{p(I) : index(I)}.
:- {p(I) : index(I)} 0.
:-  2 {p(I) : index(I)}.
```

If Z is a finite set of atoms, we will use

$$\begin{array}{rll} l \leq Z^c & \text{as shorthand for} & Z^c \wedge (l \leq Z), \\ Z^c \leq u & \text{as shorthand for} & Z^c \wedge (Z \leq u), \\ l \leq Z^c \leq u & \text{as shorthand for} & Z^c \wedge (l \leq Z \leq u). \end{array}$$

We will see in a later handout that for any finite set Z of atoms,

$$\begin{array}{rll} l \leq Z^c & \text{is intuitionistically equivalent to} & Z^c \wedge \neg(Z \leq l - 1), \\ Z^c \leq u & \text{is intuitionistically equivalent to} & Z^c \wedge \neg(u + 1 \leq Z), \\ l \leq Z^c \leq u & \text{is intuitionistically equivalent to} & Z^c \wedge \neg(Z \leq l - 1) \wedge \neg(u + 1 \leq Z) \end{array}$$

$(l > 0)$.

The theorem on intuitionistic equivalence (later handout) tells us that if two formulas are intuitionistically equivalent to each other, then they have the same answer sets. In view of this theorem, the last assertion shows that program (3) can be rewritten as

$$1 \leq \{p_1, \ldots, p_n\}^c \leq 1. \tag{4}$$

The following proposition explains why these are useful abbreviations.

Proposition on cardinality expression *For any pairwise distinct atoms $A_1, \ldots, A_n$, nonnegative integers l and u, and a set X of atoms,*

(i) X is an answer set of $l \leq \{A_1, \ldots, A_n\}^c$ iff $X \subseteq \{A_1, \ldots, A_n\}$ and $l \leq |X|$;

(ii) X is an answer set of $\{A_1, \ldots, A_n\}^c \leq u$ iff $X \subseteq \{A_1, \ldots, A_n\}$ and $|X| \leq u$;

(iii) X is an answer set of $l \leq \{A_1, \ldots, A_n\}^c \leq u$ iff $X \subseteq \{A_1, \ldots, A_n\}$ and $l \leq |X| \leq u$.

The language of LPARSE allows us to use expressions

$$l \leq Z^c,\ Z^c \leq u,\ l \leq Z^c \leq u$$

in heads of rules, with both $\leq$ and ^{c} dropped. For instance, program (4) can be written as

```
index(1..n).

1 {p(I) : index(I)} 1.
```

Note that LPARSE understands the expression

```
l {...} u
```

in different ways depending on whether it occurs in the body or in the head of a rule: it stands for

$$l \leq \{\cdots\} \leq u$$

in the body, and for

$$l \leq \{\cdots\}^c \leq u$$

in the head.

4.6 Consider the program

$$1 \leq \{p_{i1}, \ldots, p_{in}\}^c \leq 1 \qquad (1 \leq i \leq n),$$

where n is a positive integer. How many answer sets does this program have, in your opinion? Check your conjecture for $n = 3$ using SMODELS.