4. Answer Set Semantics

(Modern) Answer Set Programs

A *rule* is an implication of the form

$$F \leftarrow G \tag{1}$$

where F and G are formulas that contain no connectives other than $\{\bot, \top, \wedge, \vee, \neg\}$. F is called the *head* of the rule and G is called the *body*. If G is $\top$ we often identify the rule with its head by dropping $\leftarrow \top$. If F is $\bot$ we often write the rule as $\leftarrow G$.

A *program* is a finite set of rules.

The answer sets of a program are defined as follows. The *reduct* Π^X of a program Π relative to a set X of atoms is obtained from Π by replacing all maximal occurrences of $\neg H$ by

- $\top$ if $X \models \neg H$,
- $\bot$ otherwise.

We say that X is an *answer set* of Π if X is a minimal set satisfying Π^X. As before the minimality of X is understood here in the sense of set inclusion.

4.1 (a) Program $\neg\neg p$ has no answer sets. (b) Each of the two programs

$$p \vee \neg p \tag{2}$$

and

$$p \leftarrow \neg\neg p \tag{3}$$

has two answer sets: $\emptyset$ and $\{p\}$.

The language of GRINGO is in fact more restricted than the syntax above. A rule allowed in that language is an extension of a traditional rule described in the previous handout. We will discuss such extensions in this handout.

Nonmonotonicity of Answer Set Semantics

It is easy to see that any answer set of program Π satisfies Π. (Try to prove it!) In this sense, the answer set semantics is stronger than the propositional logic semantics, which is given by the concept of satisfaction. The sets satisfying a formula are called its "models," and the answer sets of a formula are called its "stable" models.

The classical semantics of propositional formulas is monotonic, in the sense that conjoining a formula with another formula can only make the set of its models smaller: if X satisfies $F \wedge G$ then X satisfies F. The answer set semantics is nonmonotonic: an answer set of $F \wedge G$ does not have to be an answer set of F. This phenomenon is observed even when F and G are positive programs, and it is related to the minimality condition in the definition of an answer set. For instance, the answer set $\{p, q\}$ of $p \wedge q$ is not an answer set of p. The usefulness of nonmonotonicity as a property of knowledge representation formalisms and its relation to minimality conditions were established in early work on circumscription [McCarthy, 1980].

4.2

(a) Program

$$p \vee q$$

has two answer sets: $\{p\}$ and $\{q\}$.

(b) Program

$$\begin{aligned} p \vee q \\ p \leftarrow q \\ q \leftarrow p \end{aligned}$$

has one answer set: $\{p, q\}$.

These facts provide one more example of nonmonotonicity: after appending the last two rules in (b), the program $p \vee q$ gets a new answer set. They show also that disjunction, under the answer set semantics, is sometimes "exclusive," and sometimes not. On the one hand, among the three sets

$$\{p\},\ \{q\},\ \{p, q\}$$

that satisfy $p \vee q$, the third is not an answer set. On the other hand, appending the last two rules to the disjunction makes $\{p, q\}$ an answer set; the disjunction "becomes inclusive."

Choice Formulas and Constraints

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 two kinds. A "choice rule" is a program with many answer sets that are more than the collection of sets that we want to describe. The additional answer sets can be removed by adding "constraints" to this program.

Choice Formulas

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

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

4.3 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 GRINGO 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

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

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 GRINGO concisely using variables. For instance, the rule

$$\{p_1, \ldots, p_7\}^c \leftarrow q \tag{4}$$

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 the following example:

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

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

$$\{p_i\}^c \leftarrow q \qquad (1 \leq i \leq 7). \tag{5}$$

In this example, the difference between local and global variables is not essential, however: there is a theory that tells that replacing (4) with (5) in any program does not affect the program's answer sets.[1]

Constraints

A *constraint* is a rule with the head $\bot$, that is to say, a rule of the form $\bot \leftarrow F$. (Recall it can be abbreviated as $\leftarrow F$.) The following theorem tells us how adding a constraint to a program affects the collection of its answer sets.

Theorem on Constraints [Lifschitz *et al.*, 1999]. For any program Π and formula F, a set X of atoms is an answer set for $\Pi \cup \{\leftarrow F\}$ iff X is an answer set for Π and does not satisfy F.

4.4^e Find the answer sets of the program

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

(a) using Theorem on Constraints; (b) using CLASP.

We observed earlier that a program $\Pi_1 \cup \Pi_2$ may have answer sets that are not found among the answer sets of Π_1. The assertion of the Theorem on Constraints shows, however, that this cannot happen if Π_2 is a constraint. Conjoining a program with a constraint leads only to the loss of answer sets.

[1] This theory is called "strong equivalence," which is beyond the scope of this class.

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

4.5^e (a) For any propositional formula F, a set X of atoms occurring in F is a model of F iff X is an answer set of $Z^c \wedge \neg\neg F$. (b) The statement (a) remains true if $\neg\neg F$ in it is replaced with F.

This fact allows us to compute the models of propositional formulas using answer set solvers. However, current answer set solvers requires the input to be in rule forms, which can be viewed as a conjunction of certain kinds of implications.

Cardinality Formulas

Constraints used in ASP programs often involve conditions on the cardinality of a set of atoms. We will introduce special notation for such formulas. A *cardinality atom* is an expression of the form

$$l\{A_1, ..., A_n\} \tag{6}$$

where l is a nonnegative integer ("bound") and all A_i are atoms. The concept of *cardinality formula* extends the definition of a formula in propositional logic as follows.

- every atom is a cardinality formula;
- every cardinality atom is a cardinality formula;
- both 0-place connectives are cardinality formulas;
- if F is a cardinality formula, then $\neg F$ is a cardinality formula;
- if F and G are cardinality formulas, then $(F \wedge G)$ and $(F \vee G)$ are cardinality formulas.

Similarly we extend rules and programs by referring to cardinality formulas in place of formulas. That is, a *cardinality rule* is an expression of the form (1) where F and G are cardinality formulas. A *cardinality program* is a finite set of cardinality rules.

The concept of satisfaction for cardinality formulas is defined by extending the concept of satisfaction in propositional logic. We say that a set X of atoms satisfies (6) if the number of atoms among $A_1, ..., A_n$ that belong to X is at least l. Otherwise the definition remains the same.

The definition of a reduct remains the same except that we refer to the extended notion of satisfaction instead.

4.6 Find all answer sets of the following programs.

(a)

$$\{p, q, r\}^c$$
$$s \leftarrow 2\,\{p, q, r\}.$$

(b)

$$\{p, q, r\}^c$$
$$p \leftarrow 2\,\{p, q, r\}.$$

Notation:

$$\{A_1, \ldots, A_n\}\,u$$

stands for

$$\neg(u+1)\,\{A_1, \ldots, A_n\};$$

$$l\,\{A_1, \ldots, A_n\}\,u$$

stands for

$$l\,\{A_1, \ldots, A_n\} \;\wedge\; \{A_1, \ldots, A_n\}\,u$$

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

$$\begin{array}{l} \{p_1, \ldots, p_n\}^c \\ \leftarrow \{p_1, \ldots, p_n\}\,0 \\ \leftarrow 2\,\{p_1, \ldots, p_n\}. \end{array} \tag{7}$$

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 (7) are the singletons $\{p_1\}, \ldots, \{p_n\}$.

Program (7) can be written in the language of GRINGO as

```
index(1..n).

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

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

4.7 Although the syntax of GRINGO does not allow us, generally, to use nested negations, the occurrence of formula $\neg\neg p$ can sometimes be written as a cardinality formula that GRINGO understands. Find such a formula and use CLASP to find the answer sets

$$p \leftarrow \neg\neg p. \tag{8}$$

(Do not use choice rules.)

Notation:

$$\begin{array}{rcl} l\,\{A_1,\ldots,A_n\}^c & \text{stands for} & \{A_1,\ldots,A_n\}^c \ \wedge\ l\,\{A_1,\ldots,A_n\}, \\ \{A_1,\ldots,A_n\}^c\,u & \text{stands for} & \{A_1,\ldots,A_n\}^c \ \wedge\ \{A_1,\ldots,A_n\}\,u, \\ l\,\{A_1,\ldots,A_n\}^c\,u & \text{stands for} & \{A_1,\ldots,A_n\}^c \wedge\ l\,\{A_1,\ldots,A_n\}\,u. \end{array}$$

Proposition on cardinality atom *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 $l\,\{A_1,\ldots,A_n\}^c$ iff $X \subseteq \{A_1,\ldots,A_n\}$ and $l \leq |X|$;

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

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

The language of GRINGO allows us to use expressions

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

in heads of rules, with ^{c} dropped. For instance, program

$$1\,\{p_1,\ldots,p_n\}^c\,1. \tag{9}$$

can be written as

```
index(1..n).

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

Note that GRINGO 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

$$l\,\{\cdots\}\,u$$

in the body, and for

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

in the head.

4.8 Consider the program

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

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

References

[Lifschitz *et al.*, 1999] Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25:369–389, 1999.

[McCarthy, 1980] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39,171–172, 1980.