Stable Models

Vladimir Lifschitz, University of Texas

This is an introduction to the theory of stable models, which is important for understanding answer set programming. The companion document, *Programming with CLINGO*, explains how to use the answer set programming system CLINGO.

There are many exercises here, and you'll find answers to them at the end.

1 Propositional Formulas

To get ready for the study of stable models, we review in this section a few concepts related to propositional formulas. Some details may be different from what you saw in other classes and in textbooks.

Assume that we are given a set σ of symbols called *atomic propositions*. *Formulas* over σ are built from atomic propositions and the logical constants $\bot$ (falsity) and $\top$ (truth) using the connectives $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), and $\leftarrow$ (implication). Note that implications are written here "backwards": $q \leftarrow p$ ("q if p") instead of $p \rightarrow q$ ("if p then q"). Implication binds weaker than the other connectives; for instance,

$$p \leftarrow q \wedge r \tag{1}$$

is understood as shorthand for the formula $p \leftarrow (q \wedge r)$.

If one of the truth values *false*, *true* is assigned to each atomic proposition then the truth values of other formulas are defined as follows. The truth value of $\bot$ is *false*; the truth value of $\top$ is *true*. For propositional connectives, we use the following truth tables:

F	$\neg F$
false	*true*
true	*false*

F	G	$F \wedge G$	$F \vee G$	$F \leftarrow G$
false	*false*	*false*	*false*	*true*
false	*true*	*false*	*true*	*false*
true	*false*	*false*	*true*	*true*
true	*true*	*true*	*true*	*true*

Sets of atomic propositions will be called *interpretations*. An interpretation I can be thought of as an assignment of truth values to atomic propositions: those

that belong to I get the value *true*, and all others get the value *false*. If a formula F gets the value *true* for an interpretation I then we say that I *satisfies* F, or that I is a *model* of F. For instance, the empty set $\emptyset$ satisfies formula (1), because that formula gets the value *true* when all atomic propositions p, q, r get the value *false*.

Exercise S.1.1. Assuming that $\sigma = \{p, q, r\}$, find all interpretations that do not satisfy formula (1).

Exercise S.1.2. Find a formula that is satisfied by $\{p\}$ and by $\{q\}$ but is not satisfied by $\{p, q\}$.

We say that an interpretation I is a *model* of a set Γ of formulas if I satisfies all formulas in Γ.

Exercise S.1.3. Assuming that $\sigma = \{p, q, r\}$, find all models of the set

$$\{p \leftarrow q \wedge r,\ q \leftarrow p,\ r \leftarrow p\}.$$

Exercise S.1.4. Assuming that σ is the infinite set $\{p_0, p_1, p_2, \dots\}$, find all models of the infinite set of formulas

$$\{p_1, \neg p_2, p_3, \neg p_4, \dots\}.$$

A *tautology* is a formula that is satisfied by all interpretations.

Exercise S.1.5. Determine which of the given formulas are tautologies.

(a) $p \vee \top \leftarrow \neg q$.

(b) $p \vee q \leftarrow \neg q \wedge \bot$.

(c) $p \leftarrow q \wedge r \wedge \neg q$.

A set Γ of formulas is *satisfiable* if there exists an interpretation satisfying all formulas in Γ, and *unsatisfiable* otherwise.

Exercise S.1.6. Any set of formulas of the form $A \leftarrow F$, where A is an atomic proposition, is satisfiable. True or false?

Two formulas or sets of formulas are *equivalent* to each other if they have the same models. For instance, the set $\{p, q \leftarrow p\}$ is equivalent to the formula $p \wedge q$.

Exercise S.1.7. Determine whether the given formulas or sets of formulas are equivalent.

(a) $\{p \leftarrow q,\ q \leftarrow r\}$ and $p \leftarrow r$.

(b) $p \wedge q \leftarrow r$ and $\{p \leftarrow r,\ q \leftarrow r\}$.

(c) $p \leftarrow q \vee r$ and $\{p \leftarrow q,\ p \leftarrow r\}$.

(d) $p \leftarrow p$ and $q \vee \neg q$.

(e) $\{p \leftarrow p,\ q\}$ and q.

(f) $\{p,\ \neg p\}$ and $\bot$.

(g) $p \leftarrow \neg q$ and $q \leftarrow \neg p$.

(h) $p \leftarrow \neg q$ and $p \vee q$.

(i) $\bot \leftarrow p$ and $\neg p$.

If Γ is a finite set of formulas then we can form the conjunction and the disjunction of all elements of Γ. We will use this terminology even when Γ is empty; the conjunction of the empty set of formulas is understood as $\top$, and the disjunction as $\bot$. This is similar to the convention adopted in algebra: the sum of the empty set of numbers is understood as 0, and the product (for instance, 2^0 and 0!) is understood as 1.

The formulas in the examples and exercises above use the implication symbol $\leftarrow$ in a limited way. Some of them don't contain implication at all. Others have the form $F \leftarrow G$, where F and G don't contain implication. Formulas of these two kinds will be called *propositional rules* (or simply *rules*), and sets of propositional rules will be called *propositional programs* (or simply *programs*). About a propositional rule $F \leftarrow G$ we say that F is its *head* and G is its *body*. If there are no implications in F then we say that the whole formula F is the head of the rule.

Syntactically, propositional rules are somewhat similar to CLINGO rules (see *Programming with CLINGO*, Section 1). We will define which models of a propositional program are called stable, and this concept will serve as the basis for the study of CLINGO programs.

2 Minimal Models

"Minimal models" that we are going to talk about first are closely related to stable models, but the definition is simpler.

About a model I of a formula F we say that it is *minimal* if no other model of F is a subset of I. For instance, if $\sigma = \{p, q\}$ then the formula $p \vee q$ has 3 models:

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

The first two are minimal, and the third is not. For sets of formulas, the definition of a minimal model is similar: a model I of Γ is called *minimal* if no other model of Γ is a subset of I.

Exercise S.2.1. (a) Assuming that $\sigma = \{p, q, r, s\}$, find all models of the program

$$\{p \vee q,\ r \leftarrow p,\ s \leftarrow q\}.$$

(b) Which of them are minimal?

Exercise S.2.2. Find a propositional rule that has exactly 4 minimal models.

Exercise S.2.3. It is clear that if two formulas or sets of formulas are equivalent then they have the same minimal models. But the converse is not true: two formulas may have the same minimal models even though they are not equivalent to each other. Find such a pair of formulas.

A propositional rule will be called *definite* if

(i) its head is an atomic proposition or a conjunction of several atomic propositions, and

(ii) its body (if it has one) does not contain negation.

A propositional program is *definite* if all its rules are definite. A definite program has a unique minimal model, and we can construct it by accumulating, step by step, the atomic propositions that need to be included to satisfy all rules of the program. For instance, let Γ be the program

$$p, \tag{2}$$
$$q \leftarrow p \wedge r, \tag{3}$$
$$r \leftarrow p \vee t, \tag{4}$$
$$s \leftarrow r \wedge t. \tag{5}$$

Which atomic propositions need to be included in any model of (2)–(5)? To satisfy (2), we must include p. Now the body of (4) became true, and to satisfy that rule we must include r. Now the body of (3) became true, and to satisfy that rule we must include q. The set of atomic propositions that we have accumulated, $\{p, q, r\}$, satisfies all formulas (2)–(5). This is the minimal model of Γ.

Exercise S.2.4. Describe the step-by-step process of constructing the minimal model of the definite program

$$p_1 \wedge p_2 \leftarrow q_1 \vee q_2,$$
$$q_1 \leftarrow r_1 \vee r_2,$$
$$r_1.$$

Exercise S.2.5. Let Γ be the program

$$\{p_1 \leftarrow p_2 \wedge p_3,\ p_2 \leftarrow p_3 \wedge p_4,\ \ldots,\ p_8 \leftarrow p_9 \wedge p_{10}\}.$$

For each of the following programs, describe the step-by-step process of constructing its minimal model.

(a) Γ,

(b) $\Gamma \cup \{p_5\}$,

(c) $\Gamma \cup \{p_5, p_6\}$.

It is clear that condition (i) in the definition of a definite rule is essential: if we drop it then the claim that every definite program has a unique minimal model will become incorrect. Rule $p \vee q$ can serve as a counterexample. Indeed, it satisfies condition (ii), because it has no body, and it has 2 minimal models.

Exercise S.2.6. Condition (ii) in the definition of a definite rule is essential also. Give a counterexample illustrating this fact.

Exercise S.2.7. Let

$$\Gamma = \{p_1 \leftarrow p_2,\ p_2 \leftarrow p_3,\ p_3 \leftarrow p_4,\ p_4 \leftarrow p_5, \dots\},$$
$$\Delta = \{p_2 \leftarrow p_1,\ p_3 \leftarrow p_2,\ p_4 \leftarrow p_3,\ p_5 \leftarrow p_4, \dots\}.$$

For each of the following programs, describe the step-by-step process of constructing its minimal model.

(a) $\Gamma \cup \{p_3\}$,

(b) $\Delta \cup \{p_3\}$.

3 Stable Models of Positive Programs

About a propositional rule or program we say that it is *positive* if it doesn't contain negation. For example, all definite programs are positive, as well as the program from Exercise S.2.1. The rule $p \leftarrow \neg q$ is not positive.

In application to a positive program, the expression "stable model" has the same meaning as "minimal model." We can say that by doing Exercises S.2.1 and S.2.4–S.2.7 we found the stable models of the given programs.

In view of the close relationship between stable models of propositional programs and the functionality of CLINGO, we can use CLINGO to generate the stable/minimal models of positive propositional programs. Table 1 relates some of the symbols found in CLINGO programs to the corresponding symbols in propositional formulas. Note that the comma sometimes corresponds to conjunction and sometimes to disjunction, depending on where it occurs in the rule. The line that shows how negation is represented in a CLINGO program is not relevant at this point, because we are talking about positive programs here, but it will be needed in Section 8.

To instruct CLINGO to find the stable models of the propositional program from Exercise S.2.1, we rewrite the program in the syntax of CLINGO:

CLINGO rules	Propositional rules
`:-`	$\leftarrow$
comma in the body	$\wedge$
comma in the head	$\vee$
`not`	$\neg$
`#false`	$\bot$
`#true`	$\top$

Table 1: Correspondence between symbols in CLINGO rules and in propositional formulas.

```
p,q.
r :- p.
s :- q.
```

If we save these rules in file `S21.lp` and issue the command

```
% clingo S21.lp
```

then CLINGO will generate one of the two stable models of the program and stop without looking for the other one. The command line

```
% clingo S21.lp 2
```

will instruct CLINGO to find both stable models, and in the output we will see:

```
Answer: 1
s q
Answer: 2
r p
```

Nothing will change if we replace the last 2 in the command line by any larger integer. Number 0 would be understood as the instruction to find all stable models.

Rewriting rule (4) in the syntax of CLINGO is less straightforward, because the body of this rule is a disjunction. But this rule can be replaced by an equivalent pair of simpler rules; see Exercise S.1.7(c).

Exercise S.3.1. (a) Rewrite program (2)–(5) in the syntax of CLINGO. (b) Use CLINGO to find its stable model.

Exercise S.3.2. (a) Rewrite the program from Exercise S.2.4 in the syntax of CLINGO. (b) Use CLINGO to find its stable model.

Exercise S.3.3. Use CLINGO to find the number of stable models of the program

$$p \vee q,$$
$$r \vee s,$$
$$s1 \vee s2 \leftarrow s,$$
$$\bot \leftarrow p \wedge s1.$$

4 Ground Terms and Their Values

The simplest terms that we find in CLINGO programs are integers, symbolic constants representing specific objects, and variables. More complex terms contain also symbols for arithmetic operations and intervals (*Programming with CLINGO*, Sections 1 and 2). Placeholders are another kind of symbolic constant that may occur in a term (see *Programming with CLINGO*, Section 2); we assume here that each placeholder in the program has been replaced by a specific value.

Terms that do not contain variables are called *ground*. All terms that we see in stable models generated by CLINGO are ground. Moreover, terms found in a stable model contain neither arithmetic operations nor intervals. If a program contains a ground term with an arithmetic operation in it then CLINGO replaces that term by its value. For instance, given the one-rule program

```
p(2*2).
```

CLINGO will return the atom `p(4)`.

What exactly do we mean when we talk about values of ground terms? In case of the language of CLINGO, this question is not so simple, for two reasons. First, a ground term may have many values; for instance, the values of `1..3` are `1`, `2`, `3`. Second, a ground term may have no values: for instance, `1/0` has no values. Our goal here is to clarify this issue.

The set of *values* of a ground term t is defined recursively, as follows:

1. If t is an integer or a symbolic constant then the only value of t is t itself.

2. If t is $t_1 \circ t_2$, where $\circ$ is an arithmetic operation, then the values of t are integers of the form $n_1 \circ n_2$, where the integer n_1 is a value of t_1, and the integer n_2 is a value of t_2. (Table 2 shows how to translate symbols for arithmetic operations in CLINGO into standard algebraic notation.)

3. If t is $|t_1|$ then the values of t are integers of the form $|n_1|$, where the integer n_1 is a value of t_1.

4. If t is $t_1..t_2$ then the values of t are the integers n for which there exist integers n_1 and n_2 such that

- n_1 is a value of t_1,

CLINGO terms	Algebraic notation
m`*`n	$m \cdot n$
m`/`n	$\lfloor m/n \rfloor$
m`\`n	$m - n \cdot \lfloor m/n \rfloor$
m`**`n	$\lfloor m^n \rfloor$

Table 2: Correspondence between symbols for arithmetic operations in CLINGO terms and in standard algebraic notation. The symbol $\lfloor x \rfloor$ denotes the floor of a real number x, that is, the largest integer less than or equal to x. The use of this symbol in the last line is needed because n can be negative.

- n_2 is a value of t_2,
- $n_1 \leq n \leq n_2$.

For instance, the only value of `2*2` is `4`, because the only value of `2` is `2`, and $\texttt{2} \cdot \texttt{2} = \texttt{4}$. The set of values of `2/0` is empty, because division by 0 is undefined. The set of values of `2*a` is empty as well, because the only value of the symbolic constant `a` is not an integer. The same goes for `2..a`.

It is clear that every value of a ground term t is either a symbolic constant (if t itself is a symbolic constant) or an integer.

Exercise S.4.1. Determine for which of the following terms the set of values is empty.

(a) `6..5`.

(b) `a..(a+1)`.

(c) `2**(-2)`.

The syntax of CLINGO allows us to use the interval symbol in the scope of arithmetic operations. For instance, `(1..3)*2` is a term; its values, according to the definition above, are `2`, `4`, and `6`.

Exercise S.4.2. Find all values of the term `(2..4)*(2..4)`.

Exercise S.4.3. Find a ground term with the values `1`, `3`, `9`.

5 Propositional Image of a CLINGO Program

Recall that our plan is to define what we mean by a stable model of a CLINGO program using the simpler concept of a stable model of a propositional program. We are ready now to do this for CLINGO programs consisting of rules of two kinds:

$$H_1, \ldots, H_m. \tag{6}$$

($m \geq 1$) and

$$H_1, \ldots, H_m \texttt{ :- } B_1, \ldots, B_n. \tag{7}$$

($m, n \geq 0$), where the members $H_1, \ldots, H_m$ of the head and the members $B_1, \ldots, B_n$ of the body are atoms and comparisons. Recall that an atom in a CLINGO program is a symbolic constant followed by a list of terms, and a comparison is a pair of terms separated by one of the comparison symbols `= != < > <= >=` (see *Programming with CLINGO*, Section 1). For instance, the first rule given as an example in *Programming with CLINGO*

```
warm(C) :- t(C,T1), t(austin,T2), T1>T2.
```
(8)

is a rule of type (7): here $m = 1$ and H_1 is the atom `warm(C)`; $n = 3$, B_1 is the atom `t(C,T1)`, B_2 is the atom `t(austin,T2)`, and B_3 is the comparison `T1>T2.` Facts, such as

```
t(austin,88). t(dallas,95). t(houston,90). t(san_antonio,85).
```
(9)

are rules of type (6) with $m = 1$.

For any program Π consisting of rules of these types, we will describe a positive propositional program called the *propositional image* of Π. By the stable models of Π we mean the stable models of the propositional image of Π, that is to say, its minimal models (Section 3). Atomic propositions in the propositional image are ground atoms of the form $p(v_1, \ldots, v_k)$, where each v_i is a symbolic constant or an integer.

An *instance* of a rule is a rule that can be obtained from it by substituting specific values for all its (global) variables. (The distinction between local and global variables is discussed in *Programming with CLINGO*, Section 5.3. In rules of forms (6) and (7) all variables are global.) The values that may be substituted come from the set $\mathbf{S}$ of symbolic constants and the set $\mathbf{Z}$ of integers. For example, the instances of rule (8) are the rules

$$\texttt{warm(}v_0\texttt{) :- t(}v_0\texttt{,}v_1\texttt{), t(austin,}v_2\texttt{), } v_1 > v_2. \tag{10}$$

for all $v_0, v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}$. It is clear that a ground rule has one instance—the rule itself; if a rule contains variables then it has infinitely many instances.

The propositional image of a CLINGO program consists of the instances of its rules rewritten as propositional formulas. To rewrite a ground rule as a formula,

- replace the symbol `:-` and all commas in the head and the body by propositional connectives as shown in Table 1, and drop the period at the end of the rule;
- replace each of the expressions H_i and B_j by its propositional image as described in Table 3;

Expression	Propositional image
atom $p(t_1, \ldots, t_k)$ in the head	conjunction of all formulas of the form $p(v_1, \ldots, v_k)$ where v_i is a value of t_i $(i = 1, \ldots, k)$
atom $p(t_1, \ldots, t_k)$ in the body	disjunction of all formulas of the form $p(v_1, \ldots, v_k)$ where v_i is a value of t_i $(i = 1, \ldots, k)$
comparison $t_1 \prec t_2$ in the head	$\top$ if for every value v_1 of t_1 and every value v_2 of t_2, $v_1 \prec v_2$; $\bot$ otherwise
comparison $t_1 \prec t_2$ in the body	$\top$ if for some value v_1 of t_1 and some value v_2 of t_2, $v_1 \prec v_2$; $\bot$ otherwise

Table 3: Propositional images of ground atoms and comparisons. Here p is a symbolic constant, each t_i is a term, and $\prec$ is a comparison symbol.

- if the head of the rule is empty, replace it by $\bot$; if the body of the rule is empty, replace it by $\top$.

Note that, according to Table 3, the same expression may correspond to different formulas depending on whether it occurs in the head or in the body. For instance,

- if H_i is `p(1..2)` then its propositional image is the conjunction $p(1) \wedge p(2)$;
- if B_j is `p(1..2)` then its propositional image is the disjunction $p(1) \vee p(2)$;
- if H_i is `p(1/0)` then its propositional image is $\top$ (see Section 1 on the conjunction and disjunction of the empty set);
- if B_j is `p(1/0)` then its propositional image is $\bot$;
- if H_i is `1..2=2..3` then its propositional image in $\bot$;
- if B_j is `1..2=2..3` then its propositional image in $\top$.

But in many cases the propositional image of a ground atom is that atom itself. For instance, the propositional images of the atoms in facts (9) are the atomic propositions

$$t(\mathit{austin}, 88),\ t(\mathit{dallas}, 95),\ t(\mathit{houston}, 90),\ t(\mathit{san\_antonio}, 85). \tag{11}$$

Rule (10), rewritten as a formula, is

$$\mathit{warm}(v_0) \leftarrow t(v_0, v_1) \wedge t(\mathit{austin}, v_2) \wedge \top \tag{12}$$

if $v_1 > v_2$, and

$$\mathit{warm}(v_0) \leftarrow t(v_0, v_1) \wedge t(\mathit{austin}, v_2) \wedge \bot \tag{13}$$

otherwise. Consequently the propositional image of CLINGO program (8), (9) consists of propositional rules (11)–(13).

This program is definite, so that is has a unique minimal model, and that model can be constructed by accumulating the atoms that need to be included to satisfy all its rules (Section 2). Before doing that, we will simplify the program. Formulas (13) are tautologies and can be dropped, and in formula (12) the conjunctive term $\top$ can be dropped, so that (11)–(13) is equivalent to the propositional program consisting of rules (11) and

$$warm(v_0) \leftarrow t(v_0, v_1) \wedge t(austin, v_2) \qquad (v_1 > v_2). \tag{14}$$

To satisfy these rules, we need to include, first of all, atoms (11). After that, among the infinitely many rules (14) there will be 2 that are not yet satisfied:

$$warm(dallas) \leftarrow t(dallas, 95) \wedge t(austin, 88)$$

and

$$warm(houston) \leftarrow t(houston, 90) \wedge t(austin, 88).$$

Once the heads

$$warm(dallas),\ warm(houston) \tag{15}$$

of these rules are added, the construction of the minimal model is completed.

This calculation justifies the assertion that atoms (11), (15) form the only stable model of CLINGO program (8), (9).

To take another example, take the one-rule program

```
p(N,N*N+N+41) :- N=1..3.
```
(16)

(This is rule (6) from *Programming with CLINGO*, with the upper bound 10 replaced by 3.) Its propositional image consists of the rules

$$\begin{aligned} p(1, 43) &\leftarrow \top, \\ p(2, 47) &\leftarrow \top, \\ p(3, 53) &\leftarrow \top, \\ p(n, n^2 + n + 41) &\leftarrow \bot \qquad \text{for all } n \in \mathbf{Z} \setminus \{1, 2, 3\}, \\ \top &\leftarrow \bot. \end{aligned} \tag{17}$$

(The last formula corresponds to the instances of (16) obtained by substituting symbolic constants for `N`.) The first 3 rules can be equivalently written as

$$p(1, 43),\ p(2, 47),\ p(3, 53),$$

so that these atoms form the only stable model of (16).

Exercise S.5.1. For each of the given rules, finds its propositional image.

(a) `square(1..2,1..2).`

(b) `square(austin..san_antonio,austin..san_antonio).`

Exercise S.5.2. (a) Find the propositional image of the program

```
p(1..3).
q(N-1..N+1) :- p(N).
```

(b) Describe the step-by-step process of constructing its minimal model.

Exercise S.5.3. Do the same for the program consisting of rules (4), (5), (8) from *Programming with CLINGO*.

In the following exercises, the propositional image is not definite, and there is no guarantee that it has a unique minimal model.

Exercise S.5.4. (a) Find the propositional image of the program

```
p(1..3).
X=1 :- p(X).
```

(b) Find all its minimal models.

Exercise S.5.5. Do the same for the program

```
p(1),p(2),p(3).
:- p(X), X>2.
```

6 Negation as Failure

Our next goal is to extend the definition of a stable model from Section 3 to propositional programs that contain negation. In this section, we discuss informally a few examples.

The program

$$p,$$
$$q,$$
$$r \leftarrow p,$$
$$s \leftarrow q$$

is positive definite, and its stable model can be formed by accumulating the atomic propositions that are needed to satisfy all rules: we include p and q to satisfy the first two rules, and then add r and s to satisfy the other two. The stable model

is $\{p, q, r, s\}$. Consider the modification of that program in which the conjunctive term $\neg s$ is added to the body of the third rule:

$$p, \tag{18}$$

$$q, \tag{19}$$

$$r \leftarrow p \wedge \neg s, \tag{20}$$

$$s \leftarrow q. \tag{21}$$

We think of this conjunctive term as a restriction on the process of accumulating atomic propositions in the process of constructing the stable model. Rule (20) instructs us to add r to the model if p is included under the condition that

s is not included in the model and will not be included in the future. (22)

Since the program contains rules (19) and (21), the model that we are building will include s, so that rule (20) is "disabled." The stable model of program (18)–(21) is $\{p, q, s\}$.

Exercise S.6.1. (a) Rewrite propositional program (18)–(21) as a CLINGO program. (b) Check that the stable model generated by CLINGO is indeed $\{p, q, s\}$.

Consider now the program consisting of only three rules (18)–(20). Since the heads of the rules of this program don't contain s, condition (22) is satisfied, and, in accordance with rule (20), we add r. The stable model of program (18)–(20) is $\{p, q, r\}$.

The idea that formulas beginning with negation in the body of a rule represent restrictions that can "disable" the use of that rule for accumulating atomic propositions can be expressed by saying that we understand the negation symbol as *negation as failure*. (Condition (22) says that the attempt to justify including s in the model will fail.) This explanation of negation as failure is somewhat vague, because it is circular: which rules are disabled depends on which atomic propositions are going to be included in the model, and the other way around. The definition of a stable model in the next section makes the idea of negation as failure precise. But there are many cases when the meaning of the informal explanation above is sufficiently clear.

Exercise S.6.2. (a) Use the process described above to find the stable model of program (18), (20), (21). (b) Check whether your answer agrees with the output of CLINGO.

Exercise S.6.3. (a) Use the process described above to find the stable model of the program

$$p \leftarrow \neg q, \tag{23}$$

$$q \leftarrow \neg r. \tag{24}$$

(b) Check whether your answer agrees with the output of CLINGO.

The last example in this section is the program

$$p \leftarrow \neg q, \tag{25}$$
$$q \leftarrow \neg p. \tag{26}$$

Which of the atomic propositions p, q will we include in the process of constructing its stable model? There are two ways to answer this question. We can include p; then rule (26) is "disabled", so that q is not included, and rule (25) justifies the presence of p in the model. On the other hand, not including p is a reasonable option as well; then rule (26) justifies including q, rule (25) is "disabled," and this fact justifies not including p. So program (25), (26) has two stable models, $\{p\}$ and $\{q\}$.

7 Stable Models of Programs with Negation

After the informal discussion of the examples above, we will turn now to precise definitions.

A *critical part* of a propositional rule is a subformula of its head or body that begins with negation but is not part of another subformula that begins with negation. For instance, the only critical part of rule (20) is $\neg s$. Rules (18), (19), (21) have no critical parts, because they don't contain negation. The rule

$$\neg p \leftarrow \neg(q \wedge \neg r)$$

has two critical parts: the head $\neg p$ and the body $\neg(q \wedge \neg r)$. (The subformula $\neg r$ is not critical because it is contained in a larger subformula that begins with negation.)

Let Γ be a propositional program, and let I be an interpretation. The *reduct* Γ^I of Γ relative to I is the positive propositional program obtained from Γ by replacing each critical part $\neg H$ of each of its rules by $\top$ if I satisfies $\neg H$, and by $\bot$ otherwise. If I is a minimal model of the reduct Γ^I then we say that I is a *stable model* of Γ.

For instance, let Γ be program (18)–(21), and let I be $\{p, q, s\}$. The reduct Γ^I is

$$\begin{array}{l} p, \\ q, \\ r \leftarrow p \wedge \bot, \\ s \leftarrow q. \end{array} \tag{27}$$

This program is definite, and its only minimal model is $\{p, q, s\}$—exactly the interpretation I. Consequently I is a stable model of (18)–(21).

The interepretation $\{p, q\}$ is not a stable model of (18)–(21). Indeed, the reduct of the program relative to this interpretation is

$$\begin{array}{l} p, \\ q, \\ r \leftarrow p \wedge \top, \\ s \leftarrow q, \end{array} \tag{28}$$

and $\{p, q\}$ is not a model of this reduct—it doesn't satisfy the last two rules.

Interpretation $\{p, q, r, s\}$ is not a stable model of (18)–(21) either. Indeed, the reduct of the program relative to this interpretation is (27), and $\{p, q, r, s\}$ is not minimal among the models of (27).

It is easy to show, in fact, that program (18)–(21) has no stable models other than $\{p, q, s\}$. Take any interpretation I different from $\{p, q, s\}$, and consider two cases. *Case 1:* $s \in I$. The reduct of the program relative to I is (27). The only minimal model of the reduct is $\{p, q, s\}$, and it is different from I. Consequently I is not stable. *Case 2:* $s \notin I$. The reduct of the program relative to I is (28), and the only minimal model of the reduct is $\{p, q, r, s\}$. Since this model contains s, it is different from I, so that I is not stable.

Exercise S.7.1. (a) Prove that $\{p, q, r\}$ is a stable model of program (18)–(20). (b) Prove that program (18)–(20) has no other stable models.

Exercise S.7.2. Determine whether the interpretation $\{p, q\}$ is a stable model of program (25), (26).

Let's find now all stable models of the program

$$p \vee q, \tag{29}$$

$$r \leftarrow \neg p. \tag{30}$$

Take an arbitrary interpretation I, and consider two cases. *Case 1:* $p \in I$. The reduct of (29), (30) relative to I is

$$\begin{array}{l} p \vee q, \\ r \leftarrow \bot. \end{array}$$

It has two minimal models, $\{p\}$ and $\{q\}$. The former satisfies the condition $p \in I$ that characterizes Case 1, so that it is a stable model of (29), (30). *Case 2:* $p \notin I$. The reduct of (29), (30) relative to I is

$$\begin{array}{l} p \vee q, \\ r \leftarrow \top. \end{array}$$

It has two minimal models, $\{p, r\}$ and $\{q, r\}$. The latter satisfies the condition $p \notin I$ that characterizes Case 2, so that it is a stable model of (29), (30). We conclude that this program has two stable models, $\{p\}$ and $\{q, r\}$.

Exercise S.7.3. (a) Find all stable models of the program

$$p \vee q,$$
$$r \vee \neg s \leftarrow p.$$

(b) Check whether your answer agrees with the output of CLINGO.

Exercise S.7.4. Find all stable models of the program

$$p \leftarrow \neg q,$$
$$q \leftarrow \neg p,$$
$$r \leftarrow p,$$
$$r \leftarrow q.$$

Exercise S.7.5. Find all stable models of each of the following one-rule programs:

(a) $p \leftarrow \neg p$,

(b) $p \leftarrow \neg\neg p$,

(c) $p \vee \neg p$.

Note that equivalent propositional programs can have different stable models. For instance, the one-rule programs

$$p \leftarrow \neg q, \quad q \leftarrow \neg p, \quad p \vee q$$

are equivalent to each other, but they have different stable models. The one-rule programs

$$p \vee \neg p \text{ and } q \vee \neg q$$

are equivalent to each other as well, because both formulas are tautologies, but their stable models are not the same.

Exercise S.7.6. If a propositional program contains a rule of the form $F \leftarrow \bot$ then removing that rule doesn't change its stable models. True or false?

8 CLINGO Programs with Negation and Choice

The definition of the propositional image in Section 5 applies to CLINGO programs that consist of rules of forms (6) and (7), where each H_i and B_j is an atom or a comparison. Extending that definition to the case when H_i and B_j can be also negated atoms and comparisons is straightforward: the symbol `not`, used in CLINGO rules, is replaced by the symbol $\neg$, which denotes negation in propositional formulas (Table 1). In this more general setting, the propositional image is a propositional program that may contain negation, and stable models of such programs are defined in Section 7 above.

Consider, for example, the program consisting of two rules:

```
composite(N) :- N=1..5, I=2..N-1, N\I=0.          (31)
```

and

```
prime(N) :- N=2..5, not composite(N).             (32)
```

(*Programming with CLINGO*, Listing 1, with $n = 5$.) Its propositional image Γ consists of the rules

$$\begin{aligned} \mathit{composite}(4) &\leftarrow \top \wedge \top \wedge \top, \\ \mathit{prime}(2) &\leftarrow \top \wedge \neg \mathit{composite}(2), \\ \mathit{prime}(3) &\leftarrow \top \wedge \neg \mathit{composite}(3), \\ \mathit{prime}(4) &\leftarrow \top \wedge \neg \mathit{composite}(4), \\ \mathit{prime}(5) &\leftarrow \top \wedge \neg \mathit{composite}(5) \end{aligned}$$

and infinitely many propositional rules containing at least one conjunctive term $\bot$ in the body. Let's check that the interpretation

$$\{\mathit{prime}(2), \mathit{prime}(3), \mathit{composite}(4), \mathit{prime}(5)\} \tag{33}$$

is a stable model of Γ. The reduct of Γ relative to (33) consists of the rules

$$\mathit{composite}(4) \leftarrow \top \wedge \top \wedge \top, \tag{34}$$
$$\mathit{prime}(2) \leftarrow \top \wedge \top, \tag{35}$$
$$\mathit{prime}(3) \leftarrow \top \wedge \top, \tag{36}$$
$$\mathit{prime}(4) \leftarrow \top \wedge \bot, \tag{37}$$
$$\mathit{prime}(5) \leftarrow \top \wedge \top \tag{38}$$

and rules containing at least one conjunctive term $\bot$ in the body. All these rules except for (34)–(36) and (38) are tautologies. These 4 rules are definite, so that the only minimal model of the reduct consists of the heads of these rules. This minimal model equals (33).

Exercise S.8.1. (a) Find the propositional image of the program

```
p(1..3).
q(X) :- X=2..4, not p(X).
```

(b) Find the reduct of this propositional image relative to the interpretation

$$\{p(1), p(2), p(3), q(4)\}. \tag{39}$$

(c) Find the minimal model of the reduct to determine whether (39) is a stable model of the program.

Exercise S.8.2. (a) Find the propositional image of the program

```
p(a).
q(b).
r(X) :- p(X), not q(X).
```

(b) Find the reduct of this propositional image relative to the interpretation

$$\{p(a), q(b), r(a), r(b)\}. \tag{40}$$

(c) Find the minimal model of the reduct to determine whether (40) is a stable model of the program.

Finally, we will extend the definition of propositional image to choice rules with head of the form

$$\{p(t_1, \ldots, t_k)\}. \tag{41}$$

The propositional image of (41) is the conjunction of all formulas of the form

$$p(v_1, \ldots, v_k) \vee \neg p(v_1, \ldots, v_k)$$

where v_i is a value of t_i $(i = 1, \ldots, k)$.

For example, the propositional image of the choice rule

```
{p(a)}.
```

is the disjunction $p(a) \vee \neg p(a)$. This is essentially the formula that we saw in Problem S.7.5(c); its stable models are $\emptyset$ and $\{p(a)\}$.

Consider now the choice rule

```
{p(1..10)}.
```

Its propositional image is

$$(p(1) \vee \neg p(1)) \wedge \cdots \wedge (p(10) \vee \neg p(10)). \tag{42}$$

Any subset I of the set $\{p(1), \ldots, p(10)\}$ is a stable model of (42). Indeed, the reduct of (42) relative to I is the conjunction of the formulas $p(k) \vee \bot$ for all atoms $p(k)$ from I, and $p(k) \vee \top$ for all atoms $p(k)$ from $\{p(1), \ldots, p(10)\} \setminus I$. Formula

$p(k) \vee \bot$ is equivalent to $p(k)$; formula $p(k) \vee \top$ is a tautology. Consequently the reduct is equivalent to the conjunction of the elements of I, and its minimal model is I.

Exercise S.8.3. (a) Find the propositional image of the program

```
{p(a)}.
q(X) :- p(X).
```

(b) Find the reduct of this propositional image relative to the interpretation

$$\{q(a)\}. \tag{43}$$

(c) Find the minimal model of the reduct to determine whether (43) is a stable model of the program.

Answers to Exercises

S.1.1. $\{q, r\}$.

S.1.2. $\neg p \vee \neg q$.

S.1.3. $\emptyset$, $\{q\}$, $\{r\}$, $\{p, q, r\}$.

S.1.4. $\{p_1, p_3, p_5, \dots\}$ and $\{p_0, p_1, p_3, p_5, \dots\}$.

S.1.5. (a)–(c).

S.1.6. True: any such set is satisfied by the set of all atomic propositions.

S.1.7. (b)–(i).

S.2.1. (a) $\{p, r\}$, $\{q, s\}$, $\{p, r, s\}$, $\{q, r, s\}$, $\{p, q, r, s\}$. (b) $\{p, r\}$ and $\{q, s\}$.

S.2.2. $p \vee q \vee r \vee s$.

S.2.3. $p \leftarrow q$ and $q \leftarrow p$.

S.2.4. Step 1: include r_1; step 2: add q_1; step 3: add p_1 and p_2. Minimal model: $\{p_1, p_2, q_1, r_1\}$.

S.2.5. (a) Minimal model: $\emptyset$. (b) Step 1: include p_5. Minimal model: $\{p_5\}$. (c) Step 1: include p_5 and p_6; step 2: add p_4; step 3: add p_3; step 4: add p_2; step 5: add p_1. Minimal model: $\{p_1, \dots, p_6\}$.

S.2.6. $p \leftarrow \neg q$.

S.2.7. (a) Step 1: include p_3; step 2: add p_2; step 3: add p_1. Minimal model: $\{p_1, p_2, p_3\}$. (b) Step 1: include p_3; step 2: add p_4; step 3: add p_5; and so on. Minimal model: $\{p_3, p_4, \dots\}$.

S.3.1. (a)

```
p.
q :- p, r.
r :- p.
r :- t.
s :- r, t.
```

S.3.2. (a)

```
p1 :- q1.
p2 :- q1.
p1 :- q2.
p2 :- q2.
q1 :- r1.
q1 :- r2.
r1.
```

S.4.1. (a) and (b).

S.4.2. `4`, `6`, `8`, `9`, `12`, `16`.

S.4.3. `3**(0..2)`.

S.5.1. (a) $\mathit{square}(1,1) \wedge \mathit{square}(1,2) \wedge \mathit{square}(2,1) \wedge \mathit{square}(2,2)$. (b) $\top$.

S.5.2. (a)

$$\begin{aligned} p(1) \wedge p(2) \wedge p(3), & \\ q(n-1) \wedge q(n) \wedge q(n+1) \leftarrow p(n) & \quad \text{for all } n \in \mathbf{Z}, \\ \top \leftarrow p(v) & \quad \text{for all } v \in \mathbf{S}. \end{aligned}$$

The rules in the last line are tautologies and can be dropped. (b) Step 1: include $p(1)$, $p(2)$, $p(3)$. Step 2: add $q(0)$, $q(1)$, $q(2)$, $q(3)$, $q(4)$. Minimal model: $\{p(1), p(2), p(3), q(0), q(1), q(2), q(3), q(4)\}$.

S.5.3. (a)

$$\begin{aligned} &\mathit{parent}(\mathit{ann}, \mathit{bob}), \\ &\mathit{parent}(\mathit{bob}, \mathit{carol}), \\ &\mathit{parent}(\mathit{bob}, \mathit{dan}), \\ &\quad \mathit{child}(v_1, v_2) \leftarrow \mathit{parent}(v_2, v_1) && \text{for all } v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}, \\ &\mathit{ancestor}(v_1, v_2) \leftarrow \mathit{parent}(v_1, v_2) && \text{for all } v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}, \\ &\mathit{ancestor}(v_1, v_3) \leftarrow \mathit{ancestor}(v_1, v_2) \wedge \mathit{ancestor}(v_2, v_3) && \text{for all } v_1, v_2, v_3 \in \mathbf{S} \cup \mathbf{Z}. \end{aligned}$$

(b) Step 1: include

$$\mathit{parent}(\mathit{ann}, \mathit{bob}), \ \mathit{parent}(\mathit{bob}, \mathit{carol}), \ \mathit{parent}(\mathit{bob}, \mathit{dan}). \tag{44}$$

Step 2: add

$$\begin{array}{c} \mathit{child}(\mathit{bob}, \mathit{ann}),\ \ \mathit{child}(\mathit{carol}, \mathit{bob}),\ \ \mathit{child}(\mathit{dan}, \mathit{bob}), \\ \mathit{ancestor}(\mathit{ann}, \mathit{bob}),\ \ \mathit{ancestor}(\mathit{bob}, \mathit{carol}),\ \ \mathit{ancestor}(\mathit{bob}, \mathit{dan}). \end{array} \tag{45}$$

Step 3: add

$$\mathit{ancestor}(\mathit{ann}, \mathit{carol}), \mathit{ancestor}(\mathit{ann}, \mathit{dan}), \tag{46}$$

Minimal model: atoms (44)–(46).

S.5.4. (a)

$$\begin{array}{ll} p(1) \wedge p(2) \wedge p(3), & \\ \top \leftarrow p(1), & \\ \bot \leftarrow p(v) & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \setminus \{1\}. \end{array} \tag{47}$$

(b) Since the set of formulas in the last line includes $\bot \leftarrow p(2)$, set (47) is unsatisfiable, and consequently has no minimal models.

S.5.5. (a)

$$\begin{array}{ll} p(1) \vee p(2) \vee p(3), & \\ \bot \leftarrow p(v) \wedge \top & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \text{ such that } v > 2, \\ \bot \leftarrow p(v) \wedge \bot & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \text{ such that } v \leq 2. \end{array} \tag{48}$$

(b) The formulas in the second line of (48) can be equivalently rewritten as $\neg p(v)$, and the formulas in the last line are tautologies. It follows that set (48) is equivalent to

$$\begin{array}{ll} p(1) \vee p(2) \vee p(3), & \\ \neg p(v) & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \text{ such that } v > 2. \end{array} \tag{49}$$

Since the set of formulas in the second line includes $\neg p(3)$, the formula in the first line can be equivalently replaced by $p(1) \vee p(2)$. Set (49) has two minimal models, $\{p(1)\}$ and $\{p(2)\}$.

S.6.1. (a)

```
p.
q.
r :- p, not s.
s :- q.
```

S.6.2. (a) Atomic proposition q is not going to be included in the stable model, because it doesn't occur in the head of any rule. From this we can conclude that s is not going to be included either, because the only rule with s in the head has the body q. Consequently condition (22) is satisfied, and the stable model is $\{p, r\}$.

S.6.3. (a) Atomic proposition r is not going to be included in the stable model, because it doesn't occur in the head of any rule. It follows that in accordance with

rule (24), the stable model will include q. From that we conclude that rule (23) is "disabled," and there is no justification for adding p. The stable model is $\{q\}$.

S.7.1. (a) The reduct of (18)–(20) relative to $\{p, q, r\}$ is

$$\begin{array}{l} p, \\ q, \\ r \leftarrow p \wedge \top, \end{array} \tag{50}$$

and the only minimal model of the reduct is $\{p, q, r\}$. (b) Take any interpretation I different from $\{p, q, r\}$, and consider two cases. *Case 1:* $s \in I$. The reduct of the program relative to I is

$$\begin{array}{l} p, \\ q, \\ r \leftarrow p \wedge \bot, \end{array}$$

and its minimal model $\{p, q\}$ doesn't contain s. *Case 2:* $s \notin I$. The reduct is (50), and its only minimal model $\{p, q, r\}$ is different from I.

S.7.2. No. The reduct is

$$\begin{array}{l} p \leftarrow \bot, \\ q \leftarrow \bot, \end{array}$$

and its minimal model $\emptyset$ is different from $\{p, q\}$.

S.7.3. Take an arbitrary interpretation I, and consider two cases. *Case 1:* $s \in I$. The reduct is

$$\begin{array}{l} p \vee q, \\ r \vee \bot \leftarrow p. \end{array}$$

It has two minimal models, $\{p, r\}$ and $\{q\}$. Neither satisfies the condition that characterizes Case 1. *Case 2:* $s \notin I$. The reduct is

$$\begin{array}{l} p \vee q, \\ r \vee \top \leftarrow p. \end{array}$$

Its minimal models are $\{p\}$ and $\{q\}$. Both satisfy the condition that characterizes Case 2, so that both are stable models of the given program.

S.7.4. Take any interpretation I, and consider four cases. *Case 1:* $p, q \in I$. The reduct is

$$\begin{array}{l} p \leftarrow \bot, \\ q \leftarrow \bot, \\ r \leftarrow p, \\ r \leftarrow q. \end{array}$$

The minimal model $\emptyset$ doesn't satisfy the condition characterizing this case. *Case 2:* $p \in I$, $q \notin I$. The reduct is

$$\begin{aligned} p &\leftarrow \top, \\ q &\leftarrow \bot, \\ r &\leftarrow p, \\ r &\leftarrow q. \end{aligned}$$

The minimal model $\{p, r\}$ satisfies the condition characterizing this case, so that $\{p, r\}$ is a stable model of the given program. *Case 3:* $p \notin I$, $q \in I$. A similar calculation gives the stable model $\{q, r\}$. *Case 4:* $p, q \notin I$. The reduct is

$$\begin{aligned} p &\leftarrow \top, \\ q &\leftarrow \top, \\ r &\leftarrow p, \\ r &\leftarrow q. \end{aligned}$$

The minimal model $\{p, q, r\}$ doesn't satisfy the condition characterizing this case. Consequently the given program has two stable models, $\{p, r\}$ and $\{q, r\}$.

S.7.5. (a) If $p \in I$ then the reduct is $p \leftarrow \bot$, and its minimal model is $\emptyset$. If $p \notin I$ then the reduct is $p \leftarrow \top$, and its minimal model is $\{p\}$. Neither model satisfies the condition characterizing the corresponding case, so that the program has no stable models. (b) If $p \in I$ then the reduct is $p \leftarrow \top$, and its minimal model is $\{p\}$. This is a stable model of the given program. If $p \notin I$ then the reduct is $p \leftarrow \bot$, and its minimal model is $\emptyset$. This is a stable model as well. (c) If $p \in I$ then the reduct is $p \vee \bot$, and its minimal model is $\{p\}$. This is a stable model of the given program. If $p \notin I$ then the reduct is $p \vee \top$, and its minimal model is $\emptyset$. This is a stable model as well.

S.7.6. True. Consider a propositional program Γ containing a rule of the form $F \leftarrow \bot$, and the program Δ obtained from Γ by removing that rule. For any interpretation I, the reduct Δ^I can be obtained from Γ^I be removing a rule of the same form, $F \leftarrow \bot$. Since all such rules are tautologies, Γ^I is equivalent to Δ^I. It follows that I is a minimal model of Γ^I if and only if I is minimal model of Δ^I.

S.8.1. (a)

$$\begin{aligned} &p(1) \wedge p(2) \wedge p(3), \\ &q(2) \leftarrow \top \wedge \neg p(2), \\ &q(3) \leftarrow \top \wedge \neg p(3), \\ &q(4) \leftarrow \top \wedge \neg p(4), \\ &q(v) \leftarrow \bot \wedge \neg p(v) \qquad (v \in \mathbf{S} \cup \mathbf{Z} \setminus \{2, 3, 4\}). \end{aligned}$$

(b) Reduct:

$$
\begin{aligned}
& p(1) \wedge p(2) \wedge p(3), \\
& q(2) \leftarrow \top \wedge \bot, \\
& q(3) \leftarrow \top \wedge \bot, \\
& q(4) \leftarrow \top \wedge \top, \\
& q(v) \leftarrow \bot \wedge \top \qquad (v \in \mathbf{S} \cup \mathbf{Z} \setminus \{2,3,4\}).
\end{aligned}
$$

(c) The minimal model $\{p(1), p(2), p(3), q(4)\}$ is the same as (39). Consequently (39) is a stable model of the program.

S.8.2.

$$
\begin{aligned}
& p(a), \\
& q(b), \\
& r(v) \leftarrow p(v) \wedge \neg q(v) \qquad (v \in \mathbf{S} \cup \mathbf{Z}).
\end{aligned}
$$

(b) Reduct:

$$
\begin{aligned}
& p(a), \\
& q(b), \\
& r(b) \leftarrow p(b) \wedge \bot, \\
& r(v) \leftarrow p(v) \wedge \top \qquad (v \in \mathbf{S} \cup \mathbf{Z} \setminus \{b\}).
\end{aligned}
$$

(c) The minimal model $\{p(a), q(b), r(a)\}$ is different from (40). Consequently (40) is not a stable model of the program.

S.8.3. (a)

$$
\begin{aligned}
& p(a) \vee \neg p(a), \\
& q(v) \leftarrow p(v) \qquad (v \in \mathbf{S} \cup \mathbf{Z}).
\end{aligned}
$$

(b) Reduct:

$$
\begin{aligned}
& p(a) \vee \top, \\
& q(v) \leftarrow p(v) \qquad (v \in \mathbf{S} \cup \mathbf{Z}).
\end{aligned}
$$

(c) The minimal model $\emptyset$ is different from (43). Consequently (43) is not a stable model of the program.