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.

In preparation for the study of stable models, we review in the first two sections a few concepts related to propositional formulas. Some details may be different from what you saw in other classes and in textbooks.

1 Propositional Formulas

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 $\sigma = \{p, q, r, s\}$, find all models of the set

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

Exercise S.1.5. Assuming that $\sigma = \{p_1, p_2, \dots, p_8\}$, find the number of models of

(a) $p_1 \wedge p_2$,

(b) $p_1 \leftarrow p_2$.

Exercise S.1.6. Assuming that σ is the infinite set $\{p_0, p_1, p_2, \dots\}$, find all models of the infinite sets

(a) $\{p_1, \neg p_2, p_3, \neg p_4, \dots\}$,

(b) $\{p_0 \leftarrow p_1, p_1 \leftarrow p_2, p_2 \leftarrow p_3, \dots\} \cup \{\neg p2\}$.

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 first define the concept of a stable model for propositional programs. This definition will serve as the basis for the study of stable models of CLINGO programs.

2 Equivalence

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 $p \wedge q$.

When two formulas or sets of formulas are not equivalent, this fact can be demonstrated by a counterexample—an interpretation that satisfies one of them but not the other. For instance, the formula $p \wedge q \wedge r$ is not equivalent to $p \wedge q$ because the interpretation $\{p, q\}$ satisfies the former but not the latter.

Exercise S.2.1. Determine whether the given formulas or sets of formulas are equivalent. If they are not, give a counterexample.

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

To simplify a propositional formula F means to find a propositional formula that is simpler than F and equivalent to F. (This terminology is useful, but it is not mathematically precise, because we didn't define "simpler.") For instance, we can say that p is the result of simplifying $p \wedge p$. In a similar way, we can talk about simplifying a set of formulas.

Exercise S.2.2. Simplify the given formulas.

(a) $\bot \leftarrow p$.

(b) $p \leftarrow \neg q$.

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

(d) $p \vee q \leftarrow r \wedge \top$.

(e) $p \vee q \leftarrow r \wedge \bot$.

A *tautology* is a formula that is satisfied by all interpretations. In other words, a tautology is a formula equivalent to $\top$. A set of formulas containing a tautology can be simplified by removing it.

Exercise S.2.3. Determine which of the given formulas are tautologies.

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

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

A formula is *satisfiable* if it is satisfied by at least one interpretation, and *unsatisfiable* otherwise. In other words, an unsatisfiable formula is a formula equivalent to $\bot$. About a set Γ of formulas we say that it is satisfiable if there exists an interpretation that satisfies all formulas in Γ.

Exercise S.2.4. The set of all formulas of the form $A \leftarrow F$, where A is an atomic proposition, is satisfiable. True or false?

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 of the empty set of numbers (for instance, 2^0 and 0!) is understood as 1. It is clear that any finite set of formulas is equivalent to the conjunction of all its elements.

3 Minimal Models

"Minimal models" that we are going to talk about now 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.3.1. (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.3.2. Find all minimal models of the program

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

Exercise S.3.3. Describe (a) models of the empty set of formulas, (b) minimal models of the empty set.

Exercise S.3.4. For the case when Γ is a set of atoms, describe (a) models of Γ, (b) minimal models of Γ.

Exercise S.3.5. Find a propositional rule that has exactly 4 minimal models.

Exercise S.3.6. 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. Once p is included, the body of (4) becomes 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.3.7. Describe the step-by-step process of constructing the minimal models of the definite programs

(a)

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

(b)

$$p \leftarrow q_1 \wedge q_2,$$
$$q_1 \leftarrow r_1 \vee r_2,$$
$$q_2 \leftarrow p \vee q_1,$$
$$r_1.$$

Exercise S.3.8. 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.3.9. Condition (ii) in the definition of a definite rule is essential also. Give a counterexample illustrating this fact.

Exercise S.3.10. Let

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

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

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

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

4 Stable Models of Positive Propositional 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 programs from Exercises S.3.1 and S.3.2. The rules $p \leftarrow \neg q$ and $\neg p \leftarrow q$ are 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.3.1(b), S.3.2, S.3.6, S.3.7, S.3.9 we found the stable models of the given propositional 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

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.

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 later.

To instruct CLINGO to find the stable models of the propositional program from Exercise S.3.1, we rewrite the program in the syntax of CLINGO:

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

If we save these rules in file `S31.lp` and issue the command

```
% clingo S31.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 S31.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 2 in the command line by any larger integer, because this program has only 2 stable models. 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, and in the (current version of) clingo there is no

symbol for disjunction in the body. But this rule can be replaced by an equivalent pair of simpler rules; see Exercise S.2.1(c).

Exercise S.4.1. (a) Rewrite program (2)–(5) in the syntax of CLINGO. (b) Use CLINGO to find its stable model.

Exercise S.4.2. (a) Rewrite the program from Exercise S.3.6(a) in the syntax of CLINGO. (b) Use CLINGO to find its stable model.

Exercise S.4.3. Use CLINGO to find the number of stable models of the program

$$
\begin{aligned}
&p \vee q,\\
&r \vee s,\\
&s1 \vee s2 \leftarrow s,\\
&\bot \leftarrow p \wedge s1.
\end{aligned}
$$

5 Propositional Image of a CLINGO Program

As discussed at the end of Section 1, 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 satisfying two conditions. First, we assume that each rule of the program has the form

$$H_1, \ldots, H_m \texttt{ :- } B_1, \ldots, B_n. \tag{6}$$

$(m, n \geq 0)$ or the simpler form

$$H_1, \ldots, H_m. \tag{7}$$

$(m \geq 1)$, 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. Second, we assume that the program doesn't contain arithmetic operations, intervals, pooling, and placeholders (see *Programming with CLINGO*, Sections 1 and 2). Every term occurring in such a program is a symbolic constant, an integer, or a variable.

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 (6): 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`. The facts

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

are rules of type (7) with $m = 1$.

For any program Π consisting of such rules 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 4). 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.

The set of all symbolic constants will be denoted by $\mathbf{S}$, and the set of all integers by $\mathbf{Z}$. If R is a rule of type (6) or (7), then an *instance* of R is any rule obtained from R by substituting values from the set $\mathbf{S} \cup \mathbf{Z}$ for all its variables. Since that set is infinite, any rule containing at least one variable has infinitely many instances. 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 $v_0, v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}$.

The *propositional image* of a program consists of the instances of its rules transformed into propositional formulas

(i) by replacing the symbol `:-` and all commas in the head and the body by propositional connectives as shown in Table 1, and dropping the period at the end of the rule;

(ii) by replacing each comparison $t_1 \prec t_2$ in the head and in the body by $\top$ if it is true, and by $\bot$ if it is false;

(iii) by replacing the head of the rule by $\bot$ if it is empty, and replacing the body by $\top$ if it is empty.

For instance, the propositional images of 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 3). 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 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 in *Programming with CLINGO* that atoms (11), (15) form the only stable model of CLINGO program (8), (9).

Exercise S.5.1. (a) Find the propositional image of the program

```
p(0,1).
p(1,2).
q(X,Y) :- p(X,Y), X>0, Y>0.
```

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

Exercise S.5.2. (a) Find the propositional image of the program consisting of rules (4), (5), (8) from *Programming with CLINGO*. (b) Describe the step-by-step process of constructing its minimal model.

Exercise S.5.3. (a) Find the propositional image of the program

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

(b) Simplify it. (c) Find all its minimal models.

6 Ground Terms and Their Values

In the next section, the definition of the propositional image is extended to programs that may contain arithmetic operations and intervals. We will see that in this more general setting the process of constructing the propositional image includes an additional element—replacing terms by their values. For instance, the propositional image of the fact

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; then m^n is a fraction.

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

is the atomic proposition $p(4)$.

Terms that do not contain variables, such as `2*2`, are called *ground*. We will talk here about values of ground terms that do not contain placeholders. The exact meaning of this concept is not so easy to define, 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. We can clarify this issue by defining the set of *values* of t 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,
- 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`.

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.

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.6.1. 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.6.2. Find all values of the term `(2..4)*(2..4)`.

Exercise S.6.3. Find a ground term (a) with the values `1`, `3`, `9`; (b) with the values `22`, `32`, `42`.

7 More on Propositional Images

We will talk now about CLINGO programs consisting of rules of forms (6) and (7) that may contain arithmetic operations and intervals (but no pooling and no placeholders). The *propositional image* of such a program is formed by the process described in Section 5, with clause (ii) modified as follows:

(ii′) by replacing each atom and each comparison in every rule by its propositional image as described in Table 3.

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,

- the propositional image of the atom `p(1..2)` in the head of a rule is the conjunction $p(1) \wedge p(2)$, but the propositional image of the same atom in the body of a rule is the disjunction $p(1) \vee p(2)$;
- the propositional image of the atom `p(1/0)` in the head is the empty conjunction $\top$, but the propositional image of the same atom in the body is the empty disjunction $\bot$;
- the propositional image of the comparison `1..2=2..3` in the head is $\bot$ (because $\{1, 2\}$ and $\{2, 3\}$ are different sets), but the propositional image of the same comparison in the body is $\top$ (because sets $\{1, 2\}$ and $\{2, 3\}$ have a common element).

However, if each of the terms $t_1, \ldots, t_k$ is a symbolic constant or an integer, as in Section 5, then the only value of t_i is t_i itself, so that the conjunction in the first line of Table 3 is the atom $p(t_1, \ldots, t_k)$, and the disjunction in the second line of the table is the same atom. Similarly, if each of the terms t_1, t_2 is a symbolic constant or an integer then the truth value described in the third line of the table is $\top$ or $\bot$ depending on whether the condition $t_1 \prec t_2$ is true or false, and the value in the fourth line is calculated in the same way. We conclude that the definition of the propositional image above has the same meaning as the definition in Section 5 when the program contains neither arithmetical operations nor intervals.

If each of the terms $t_1, \ldots, t_k$ is formed from integers using addition, subtraction, and multiplication, then t_i has a unique value v_i, so that the conjunction in the first line of Table 3 is the atom $p(v_1, \ldots, v_k)$, and so is the disjunction in the second line. For instance, the propositional image of `2*2` is `4` no matter whether this term occurs in the head and in the body.

Consider now a one-rule program containing an interval:

```
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 rules in the last two lines are tautologies, and the

other rules can be equivalently written as

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

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

The propositional image of the program

```
p(2).
p(a).
q(X+1) :- p(X).
```

consists of the formulas

$$\begin{array}{lll} p(2), & & \\ p(a), & & \\ q(m) \leftarrow q(n) & \text{for all } n \in \mathbf{Z}, & \text{where } m \text{ is the value of } n+1, \\ \top \leftarrow q(v) & \text{for all } n \in \mathbf{V}. & \end{array}$$

The rules in the last line are tautologies and can be dropped. The stable model of the program is $\{p(2), p(a), q(3)\}$.

Exercise S.7.1. 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.7.2. (a) Find the propositional image of the rule

```
p(1/N) :- N=0..1.
```

(b) Simplify it.

Exercise S.7.3. Find the propositional image of the program

```
p(1..3).
q(X) :- p(X), X=2..4.
```

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

Exercise S.7.4. Do the same for the program

```
p(1,1..2).
q(X,Y) :- p(X,Y), X!=Y.
q(X,Y) :- q(Y,X).
```

Exercise S.7.5. Do the same for the program

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

Exercise S.7.6. (a) Find the propositional image of the program

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

(b) Find all its minimal models.

8 Safety

In some cases, the propositional image of a CLINGO program has an infinite stable model. This often happens when the head of one the rules contains a variable that doesn't occur in the body. Consider, for instance, the one-rule program `p(X).` Its propositional image is the set of atoms $p(v)$ for all $v \in \mathbf{S} \cup \mathbf{Z}$. That infinite set is its own stable model.

Exercise S.8.1. (a) Find the propositional image of the one-rule program `p(2*X).` (b) Find its stable model.

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

```
p(a).
q(X,Y) :- p(X).
```

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

It is impossible, of course, to display infinitely many atoms in a finite amount of time, and it is not surprising that the algorithm used by CLINGO for generating stable models is not applicable to rules like these. Given a rule containing a variable that doesn't occur in its body, CLINGO produces an error message saying that the variable is "unsafe."

In some cases, a variable is considered unsafe even though it does occur in the body. For instance, CLINGO produces an unsafe variable message for each of the rules

```
p(X) :- X>0.
```
(18)

and

```
q(X) :- p((-1)**X).
```
(19)

There is a good reason for this: programs containing one of these rules may have infinite stable models.

Exercise S.8.3. (a) Find the propositional image of the one-rule program (18). (b) Find its stable model.

Exercise S.8.4. (a) Find the propositional image of the program consisting of the fact

```
p(1).
```

and rule (19). (b) Describe the step-by-step process of constructing its minimal model.

The program

```
p(1).
p(N+2) :- p(N).
```
(20)

has an infinite stable model too, but it is not considered unsafe. Given this program, (the current version of) CLINGO dies and has to be restarted. The difference between this program and the examples discussed earlier is that the set of ground atoms generated in the process of constructing its minimal model remains finite at each step; the minimal model is infinite only because the number of steps is infinite. This is similar to the program from Exercise S.3.10(b).

Exercise S.8.5. (a) Find the propositional image of program (20). (b) Describe the step-by-step process of constructing its minimal model.

9 Negation as Failure

Our next goal is to extend the definition of a stable model from Section 4 to propositional programs that contain negation. In this section, we discuss informally a few examples.

The program

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

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:

$$
\begin{aligned}
&p, && (21)\\
&q, && (22)\\
&r \leftarrow p \wedge \neg s, && (23)\\
&s \leftarrow q. && (24)
\end{aligned}
$$

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 (23) 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. (25)

Since the program contains rules (22) and (24), the model that we are building will include s, so that rule (23) is "disabled." The stable model of program (21)–(24) is $\{p, q, s\}$.

Exercise S.9.1. (a) Rewrite propositional program (21)–(24) 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 (21)–(23). Since the heads of the rules of this program don't contain s, condition (25) is satisfied, and, in accordance with rule (23), we add r. The stable model of program (21)–(23) 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 (25) 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.9.2. (a) Use the process described above to find the stable model of program (21), (23), (24). (b) Check whether your answer agrees with the output of CLINGO.

Exercise S.9.3. (a) Use the process described above to find the stable model of the program

$$p \leftarrow \neg q, \tag{26}$$
$$q \leftarrow \neg r. \tag{27}$$

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

The last example in this section is the program

$$p \leftarrow \neg q, \tag{28}$$
$$q \leftarrow \neg p. \tag{29}$$

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 (29) is "disabled", so that q is not included, and rule (28) justifies the presence of p in the model. On the other hand, not including p is a reasonable option as well; then rule (29) justifies including q, rule (28) is "disabled," and this fact justifies not including p. So program (28), (29) has two stable models, $\{p\}$ and $\{q\}$.

10 Stable Models of Programs with Negation

After the informal discussion of programs with negation in the previous section, we will turn now to precise definitions.

A *critical part* of a propositional formula F is a subformula of F that begins with negation but is not part of any other subformula that begins with negation. For instance, the only critical part of rule (23) is $\neg s$. Rules (21), (22), (24) 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 F be a formula, and let I be an interpretation. The *reduct* F^I of F relative to I is the formula obtained from F by substituting $\top$ for each critical part that is satisfied by I, and substituting $\bot$ for each critical part that is not satisfied by I. The reduct Γ^I of a propositional program Γ relative to I is the positive program obtained from Γ by the same process.

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 (21)–(24), and let I be $\{p, q, s\}$. Since the critical part $\neg q$ of the third rule is not satisfied by I, the reduct Γ^I is

$$\begin{array}{l} p, \\ q, \\ r \leftarrow p \wedge \bot, \\ s \leftarrow q. \end{array} \tag{30}$$

This program is definite, and its only minimal model is $\{p, q, s\}$—exactly the interpretation I. Consequently I is a stable model of (21)–(24).

The interepretation $\{p, q\}$ is not a stable model of (21)–(24). Indeed, the

reduct of the program relative to this interpretation is

$$\begin{array}{l} p, \\ q, \\ r \leftarrow p \wedge \top, \\ s \leftarrow q, \end{array} \tag{31}$$

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 (21)–(24) either. Indeed, the reduct of the program relative to this interpretation is (30), and $\{p, q, r, s\}$ is not minimal among the models of (30).

It is easy to show, in fact, that program (21)–(24) 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 (30). 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 (31), 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.10.1. Find the reduct of each of the given rules relative to $\{p\}$, and simplify it.

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

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

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

Exercise S.10.2. (a) Prove that $\{p, q, r\}$ is a stable model of program (21)–(23). (b) Prove that program (21)–(23) has no other stable models.

Exercise S.10.3. Determine whether the interpretation $\{p, q\}$ is a stable model of program (28), (29).

Let's find now all stable models of the program

$$p \vee q, \tag{32}$$
$$r \leftarrow \neg p. \tag{33}$$

Take an arbitrary interpretation I, and consider two cases. *Case 1:* $p \in I$. The reduct of (32), (33) 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 (32), (33). *Case 2:* $p \notin I$. The reduct of (32), (33) relative to I is

$$p \vee q,$$
$$r \leftarrow \top.$$

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 (32), (33). We conclude that this program has two stable models, $\{p\}$ and $\{q, r\}$.

Exercise S.10.4. (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.10.5. Do the same for the program

$$p,$$
$$\bot \leftarrow p \wedge \neg q.$$

Exercise S.10.6. Do the same for the program

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

Exercise S.10.7. Find all stable models of each of the following one-rule programs:

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

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

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

11 Strong Equivalence

If two sets of formulas are equivalent to each other then they have the same models (this is the definition of equivalence, see Section 2), and consequently the same minimal models. With the concept of a stable model, however, the situation is

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

$$p \leftarrow \neg q, \quad q \leftarrow \neg p, \quad p \vee q \tag{34}$$

are equivalent to each other. But the only stable model of the first of them is $\{p\}$; the only stable model of the second is $\{q\}$; the third has two stable models. Another example: even though the program

$$\begin{array}{l} p, \\ q \leftarrow p \end{array} \tag{35}$$

is equivalent to the program from Exercise S.10.5, these two programs have different stable models: $\{p, q\}$ is a stable model of the former, but not of the latter.

We say that a propositional rule R_1 is *strongly equivalent* to a propositional rule R_2 if replacing R_1 by R_2 in any propositional program does not change the collection of its stable models. For example, the first of rules (34) is not strongly equivalent to either of the other two rules. The second rule of the program from Exercise S.10.5 is not strongly equivalent to the second of rules (35).

In some cases, the fact that two rules are not strongly equivalent to each other can be established by comparing the stable models of these rules viewed as one-rule programs. But sometimes we have to look at the result of replacing one rule by the other in a program containing additional rules. For instance, if we consider the second rule from Exercise S.10.5 and the second of rules (35) in isolation then we will see that they have the same stable model, $\emptyset$. To argue that these rules are not strongly equivalent, we need to look at them in the presence of an additional rule—for instance, p.

Exercise S.11.1. Prove that $\neg\neg p$ is not strongly equivalent to p.

Exercise S.11.2. Prove that $\neg q \leftarrow \neg p$ is not strongly equivalent to $p \leftarrow q$.

On the other hand, we can sometimes see that a given rule is strongly equivalent to a simpler rule. Being able to recognize such cases is useful, because it may help us simplify a program before calculating its stable models.

We can assert that propositional rules R_1 and R_2 are strongly equivalent whenever we can show that, for every interpretation I, the reduct R_1^I is equivalent to the reduct R_2^I. Indeed, if R_1^I is always equivalent to R_2^I then replacing R_1 by R_2 in a program doesn't affect the set of models of any of its reducts, and consequently doesn't affect the set of the minimal models of any reduct. And the set of minimal models of the reduct relative to I is what determines whether I is a stable model.

It is clear, for instance, that if two equivalent rules are positive then they are strongly equivalent. Indeed, if R_1 and R_2 are positive and equivalent to each other then R_1^I is R_1, R_2^I is R_2, so that R_1^I is equivalent to R_2^I. In other words, the

stable models of a program are not affected by equivalent transformations that turn positive rules into positive rules.

Exercise S.11.3. Prove that $p \vee q$ is strongly equivalent to $q \vee p$.

Any rule of the form $F \leftarrow G \wedge H$ is strongly equivalent to $F \leftarrow H \wedge G$. Indeed,

$$(F \leftarrow G \wedge H)^I \text{ is } F^I \leftarrow G^I \wedge H^I,$$
$$(F \leftarrow H \wedge G)^I \text{ is } F^I \leftarrow H^I \wedge G^I,$$

and these two reducts are equivalent. In other words, changing the order of conjunctive terms in the body of a rule doesn't affect the set of stable models.

Any rule of the form $F \leftarrow G \wedge \top$ is strongly equivalent to $F \leftarrow G$. Indeed,

$$(F \leftarrow G \wedge \top)^I \text{ is } F^I \leftarrow G^I \wedge \top,$$
$$(F \leftarrow G)^I \text{ is } F^I \leftarrow G^I,$$

and these two reducts are equivalent. In other words, dropping the conjunctive term $\top$ from the body of a rule doesn't affect the set of stable models.

Exercise S.11.4. $F \wedge (G \vee H)$ is strongly equivalent to $(F \wedge G) \vee (F \wedge H)$. True or false?

Let us check now that $\neg\neg\neg p$ is strongly equivalent to $\neg p$. Consider any interpretation I. If $p \in I$ then both formulas have the same reduct:

$$(\neg\neg\neg p)^I = \bot,\ (\neg p)^I = \bot.$$

If $p \notin I$ then the formulas have the same reduct as well:

$$(\neg\neg\neg p)^I = \top,\ (\neg p)^I = \top.$$

Exercise S.11.5. The rule $p \leftarrow \neg\neg p$ is strongly equivalent to $p \vee \neg p$. True or false?

Removing a tautology from a set of formulas does not affect the set of its models, and consequently does not affect the set of its minimal models. But the set of stable models of a propositional program can change when we remove from it a rule that is a tautology. Consider, for instance, the one-rule program $p \vee \neg p$. This formula is a tautology, and it has two stable models (see Exercise S.10.7(c)). Removing it gives the empty program, which has only one stable model.

If removing a propositional rule R from any program that contains it has no effect on the set of stable models then we will say that R is *strongly tautological*. We can say, for instance, that the rule $p \vee \neg p$ is not strongly tautological.

We can assert that a propositional rule R is strongly tautological whenever we can show that, for every interpretation I, the reduct R^I is a tautology. For instance, any rule of the form

$$F \leftarrow G \wedge \bot$$

is strongly tautological, because its reduct

$$F^I \leftarrow G^I \wedge \bot$$

relative to any interpretation I is a tautology. In other words, if the body of a rule contains the conjunctive term $\bot$ then removing the whole rule from the program will not change the set of stable models.

Exercise S.11.6. Any rule of the form $F \leftarrow F \wedge G$ is strongly tautological. True or false?

12 CLINGO Programs with Negation and Choice

The definition of the propositional image in Sections 5 and 7 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 negated 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 10 above. Thus we have now a precise definition of a stable model for sets of rules (6) and (7) in which every H_i and B_j is an atom, a comparison, a negated atom, or a negated comparison.

Let's check, for example, that

$$\{p(a), q(a)\} \tag{36}$$

is a stable model of the program

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

The propositional image Γ of this program consists of the rules

$$\begin{array}{ll} p(a), & \\ q(a), & \\ r(v) \leftarrow p(v) \wedge \neg q(v) & \text{for all } v \in \mathbf{S} \cup \mathbf{Z}. \end{array}$$

The reduct of Γ relative to (36) is

$$\begin{array}{ll} p(a), & \\ q(a), & \\ r(a) \leftarrow p(a) \wedge \bot, & \\ r(v) \leftarrow p(v) \wedge \top & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \setminus \{a\}, \end{array}$$

or, equivalently,

$$\begin{array}{ll} p(a), & \\ q(a), & \\ r(v) \leftarrow p(v) & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \setminus \{a\}. \end{array}$$

This is a definite program, and (36) is its only minimal model.

Exercise S.12.1. (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

$$\{p(a), q(b), r(a), r(b)\}. \tag{37}$$

(c) Simlplify the reduct. (d) Find the minimal model of the reduct to determine whether (37) is a stable model of the program.

Consider now the program consisting of two rules:

```
composite(N) :- N=1..5, I=2..N-1, N\I=0.
```
(38)

and

```
prime(N) :- N=2..5, not composite(N).
```
(39)

(*Programming with CLINGO*, Listing 1, with $n = 5$.) We will check that the interpretation

$$\{prime(2), prime(3), composite(4), prime(5)\} \tag{40}$$

is a stable model of this program.

The propositional image Γ of the program consists of the rules

$$\begin{array}{rl} composite(4) & \leftarrow \top \wedge \top \wedge \top, \\ prime(2) & \leftarrow \top \wedge \neg composite(2), \\ prime(3) & \leftarrow \top \wedge \neg composite(3), \\ prime(4) & \leftarrow \top \wedge \neg composite(4), \\ prime(5) & \leftarrow \top \wedge \neg composite(5) \end{array} \tag{41}$$

and infinitely many propositional rules containing at least one conjunctive term $\bot$ in the body. It's useful to note that rules (41) are strongly equivalent to the rules

$$\begin{array}{rl} composite(4), & \\ prime(2) & \leftarrow \neg composite(2), \\ prime(3) & \leftarrow \neg composite(3), \\ prime(4) & \leftarrow \neg composite(4), \\ prime(5) & \leftarrow \neg composite(5), \end{array} \tag{42}$$

and that all other rules of Γ are strongly tautological (Section 11). We conclude that Γ has the same stable models as the finite program (42). The reduct of (42) relative to (40) consists of the rules

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

and is equivalent to (40). The only minimal model of that set of atoms is (40) itself.

Exercise S.12.2. (a) Find the propositional image of the program

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

(b) Simplify it using strongly equivalent transformations. (c) Find the reduct of the simplified propositional image relative to the interpretation

$$\{p(1), p(2), p(3), q(4)\}. \tag{43}$$

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

We will now extend the definition of propositional image to choice rules with head of the form

$$\{p(t_1, \ldots, t_k)\}. \tag{44}$$

The propositional image of (44) 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 Exercise S.10.7(c); its stable models are $\emptyset$ and $\{p(a)\}$.

Consider now the choice rule

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

Its propositional image is

$$(p(1) \vee \neg p(1)) \wedge \cdots \wedge (p(10) \vee \neg p(10)). \tag{45}$$

Any subset I of the set $\{p(1), \ldots, p(10)\}$ is a stable model of (45). Indeed, the reduct of (45) 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.12.3. (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

$$\{q(a)\}. \tag{46}$$

(c) Simplify the reduct. (d) Find the minimal model of the reduct to determine whether (46) is a stable model of the program.

We observed in Section 8 that a variable that occurs in the head of a rule but doesn't occur in the body is considered unsafe, and that programs including such rules often have infinite stable models. The same can be said about a variable that does occur in the body, but only in negated atoms, such as `X` in the second rule of the program

```
p(a).
q(X) :- not p(X).
```
(47)

Exercise S.12.4. Check that the set consisting of the atom $p(a)$ and the atoms $q(v)$ for all $v \in \mathbf{S} \cup \mathbf{Z} \setminus \{a\}$ is a stable model of program (47).

Answers to Exercises

S.1.1. Just one interpretation: $\{q, r\}$.

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

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

S.1.4. $\emptyset$, $\{p, q, r, s\}$.

S.1.5. (a) A subset of $\{p_1, p_2, \ldots, p_8\}$ satisfies $p_1 \wedge p_2$ iff it includes both p_1 and p_2. For each of the other 6 atomic propositions we can decide arbitrarily whether to include it. Consequently the number of models of $p_1 \wedge p_2$ is 2^6, or 64.

(b) The set $\{p_1, p_2, \ldots, p_8\}$ has 2^8 subsets. A subset of that set satisfies $p_1 \leftarrow p_2$ unless it includes p_2 but does not include p_1. The number of such exceptional subsets is 2^6. It follows that the number of models of $p_1 \leftarrow p_2$ is $2^8 - 2^6$, or 192.

S.1.6. (a) $\{p_1, p_3, p_5, \ldots\}$ and $\{p_0, p_1, p_3, p_5, \ldots\}$. (b) $\emptyset$, $\{p_0\}$, $\{p_0, p_1\}$.

S.2.1. (a): not equivalent; for example, $\{q\}$ satisfies $p \leftarrow r$ but does not satisfy the first of the formulas $p \leftarrow q$, $q \leftarrow r$. (b)–(e): equivalent.

S.2.2. (a) $\neg p$. (b) $p \vee q$. (c) $\bot$. (d) $p \vee q \leftarrow r$. (e) $\top$.

S.2.3. Both (a) and (b).

S.2.4. True: any such set is satisfied by the set of all atomic propositions.

S.3.1. (a) $\{p, r\}$, $\{q, s\}$, $\{p, r, s\}$, $\{q, r, s\}$, $\{p, q, r, s\}$. (b) $\{p, r\}$ and $\{q, s\}$.

S.3.2. $\{p, q\}$ and $\{r\}$.

S.3.3. (a) All interpretations are models. (b) The only minimal model is $\emptyset$.

S.3.4. (a) All sets of interpretations containing Γ are models. (b) The only minimal model is Γ itself.

S.3.5. $p \vee q \vee r \vee s$.

S.3.6. $p \leftarrow q$ and $q \leftarrow p$.

S.3.7. (a) Step 1: include r_1; step 2: add q_1; step 3: add p_1 and p_2. Minimal model: $\{p_1, p_2, q_1, r_1\}$. (b) Step 1: include r_1; step 2: add q_1; step 3: add q_2; step 4: add p. Minimal model: $\{p, q_1, q_2, r_1\}$.

S.3.8. (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: $\{p_1, \ldots, p_6\}$.

S.3.9. $p \leftarrow \neg q$.

S.3.10. (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: $\{p_3, p_4, \ldots\}$.

S.4.1. (a)

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

S.4.2. (a)

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

S.5.1. (a)

$p(0,1)$,
$p(1,2)$,
$q(v_1,v_2) \leftarrow p(v_1,v_2) \wedge \top \wedge \top$ for all $v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}$ such that $v_1, v_2 > 0$,
$q(v_1,v_2) \leftarrow p(v_1,v_2) \wedge \bot \wedge \top$ for all $v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}$ such that $v_1 \leq 0$, $v_2 > 0$,
$q(v_1,v_2) \leftarrow p(v_1,v_2) \wedge \top \wedge \bot$ for all $v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}$ such that $v_1 > 0$, $v_2 \leq 0$,
$q(v_1,v_2) \leftarrow p(v_1,v_2) \wedge \bot \wedge \bot$ for all $v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}$ such that $v_1, v_2 \leq 0$.

(b)

$p(0,1)$,
$p(1,2)$,
$q(v_1,v_2) \leftarrow p(v_1,v_2)$ for all $v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}$ such that $v_1, v_2 > 0$.

(c) Step 1: include $p(0,1)$ and $p(1,2)$. Step 2: add $q(1,2)$.

S.5.2. (a)

$parent(ann, bob)$,
$parent(bob, carol)$,
$parent(bob, dan)$,
$child(v_1,v_2) \leftarrow parent(v_2,v_1)$ for all $v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}$,
$ancestor(v_1,v_2) \leftarrow parent(v_1,v_2)$ for all $v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}$,
$ancestor(v_1,v_3) \leftarrow ancestor(v_1,v_2) \wedge ancestor(v_2,v_3)$ for all $v_1, v_2, v_3 \in \mathbf{S} \cup \mathbf{Z}$.

(b) Step 1: include

$$parent(ann, bob),\ parent(bob, carol),\ parent(bob, dan). \tag{48}$$

Step 2: add

$$\begin{gathered} child(bob, ann),\ child(carol, bob),\ child(dan, bob), \\ ancestor(ann, bob),\ ancestor(bob, carol),\ ancestor(bob, dan). \end{gathered} \tag{49}$$

Step 3: add

$$ancestor(ann, carol), ancestor(ann, dan). \tag{50}$$

Minimal model: atoms (48)–(50).

S.5.3. (a)

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

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

$$\begin{array}{ll} p(1) \vee p(2) \vee p(3), & \\ \neg p(v) & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \text{ such that } v > 2. \end{array} \tag{51}$$

Since the set of formulas in the second line includes $\neg p(3)$, the formula in the first line can be equivalently replaced by $p(1) \vee p(2)$. (c) Set (51) has two minimal models, $\{p(1)\}$ and $\{p(2)\}$.

S.6.1. (a) and (b).

S.6.2. `4`, `6`, `8`, `9`, `12`, `16`.

S.6.3. (a) `3**(0..2)`; (b) `10*(2..4)+2`.

S.7.1. (a) $\mathit{square}(1,1) \wedge \mathit{square}(1,2) \wedge \mathit{square}(2,1) \wedge \mathit{square}(2,2)$. (b) $\top$.

S.7.2. (a)

$$\begin{array}{rl} p(-1) & \leftarrow \bot, \\ p(0) & \leftarrow \bot, \\ p(1) & \leftarrow \top, \\ \top & \leftarrow \bot. \end{array}$$

(b) $p(1)$.

S.7.3. (a)

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

(b)

$$\begin{array}{l} p(1) \wedge p(2) \wedge p(3), \\ q(2) \leftarrow p(2), \\ q(3) \leftarrow p(3), \\ q(4) \leftarrow p(4). \end{array}$$

(c) Step 1: include $p(1)$, $p(2)$, $p(3)$. Step 2: add $q(2)$ and $q(3)$.

S.7.4. (a)

$$
\begin{array}{rll}
p(1,1) \wedge\ p(1,2), & & \\
q(v_1, v_2) \leftarrow p(v_1, v_2) \wedge \top & & \text{for all } v_1, v_2 \in \mathbf{S} \cup \mathbf{Z} \text{ such that } v_1 \neq v_2, \\
q(v, v) \leftarrow p(v, v) \wedge \bot & & \text{for all } v \in \mathbf{S} \cup \mathbf{Z}, \\
q(v_1, v_2) \leftarrow q(v_2, v_1) & & \text{for all } v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}.
\end{array}
$$

(b)

$$
\begin{array}{rll}
p(1,1) \wedge\ p(1,2), & & \\
q(v_1, v_2) \leftarrow p(v_1, v_2) & & \text{for all } v_1, v_2 \in \mathbf{S} \cup \mathbf{Z} \text{ such that } v_1 \neq v_2, \\
q(v_1, v_2) \leftarrow q(v_2, v_1) & & \text{for all } v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}.
\end{array}
$$

(c) Step 1: include $p(1,1)$, $p(1,2)$. Step 2: add $q(1,2)$. Step 3: add $q(2,1)$.

S.7.5. (a)

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

(b) The rules in the last line are tautologies and can be dropped.

(c) Step 1: include $p(1)$, $p(2)$, $p(3)$. Step 2: add $q(0)$, $q(1)$, $q(2)$, $q(3)$, $q(4)$.

S.7.6. (a)

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

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

S.8.1. (a)

$$
\begin{array}{ll}
p(m) & \text{for all even numbers } m, \\
\top. &
\end{array}
$$

(b) The set of atoms $p(m)$ for all even numbers m.

S.8.2. (a)

$$
\begin{array}{ll}
p(a), & \\
q(v_1, v_2) \leftarrow p(v_1) & \text{for all } v_1, v_2 \in \mathbf{S} \cup \mathbf{Z}.
\end{array}
$$

(b) Step 1: include $p(a)$. Step 2: Add $q(a, v)$ for all $v \in \mathbf{S} \cup \mathbf{Z}$.

S.8.3. (a)

$$
\begin{array}{ll}
p(v) \leftarrow \top & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \text{ such that } v > 0, \\
p(v) \leftarrow \bot & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \text{ such that } v \leq 0.
\end{array}
$$

(b) The set of atoms $p(v)$ for all $v \in \mathbf{S} \cup \mathbf{Z}$ such that $v > 0$.

S.8.4. (a)

$$\begin{array}{l} p(1), \\ q(n) \leftarrow p(m) \qquad \text{for all } n \in \mathbf{Z}, \text{ where } m \text{ is the value of } (-1)^n, \\ q(v) \leftarrow \bot \qquad \text{for all } v \in \mathbf{S}. \end{array}$$

(b) Step 1: Include $p(1)$. Step 2: Add $q(n)$ for all even numbers n.

S.8.5. (a)

$$\begin{array}{l} p(1), \\ p(m) \leftarrow p(n) \qquad \text{for all } n \in \mathbf{Z}, \text{ where } m \text{ is the value of } n+2, \\ \top \leftarrow p(v) \qquad \text{for all } v \in \mathbf{S}. \end{array}$$

(b) Step 1: include $p(1)$. Step 2: add $p(3)$. Step 3: add $p(5)$, and so on. Stable model: $\{p(1), p(3), p(5), \dots\}$.

S.9.1. (a)

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

S.9.2. (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 (25) is satisfied, and the stable model is $\{p, r\}$.

S.9.3. (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 (27), the stable model will include q. From that we conclude that rule (26) is "disabled," and there is no justification for adding p. The stable model is $\{q\}$.

S.10.1. (a) $p \leftarrow q \wedge \top$, equivalent to $p \leftarrow q$. (b) $p \leftarrow \top \wedge \top$, equivalent to p. (a) $p \leftarrow \bot \wedge \top$, equivalent to $\top$.

S.10.2. (a) The reduct of (21)–(23) relative to $\{p, q, r\}$ is

$$\begin{array}{l} p, \\ q, \\ r \leftarrow p \wedge \top, \end{array} \tag{52}$$

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

$$p,$$
$$q,$$
$$r \leftarrow p \wedge \bot,$$

and its minimal model $\{p, q\}$ doesn't contain s. *Case 2:* $s \notin I$. The reduct is (52), and its only minimal model $\{p, q, r\}$ is different from I.

S.10.3. No. The reduct consists of two tautologies

$$p \leftarrow \bot,$$
$$q \leftarrow \bot,$$

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

S.10.4. (a) Take an arbitrary interpretation I, and consider two cases. *Case 1:* $s \in I$. The reduct is

$$p \vee q,$$
$$r \vee \bot \leftarrow p.$$

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

$$p \vee q,$$
$$r \vee \top \leftarrow p.$$

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.10.5. Take any interpretation I, and consider two cases. *Case 1:* $q \in I$. The reduct is

$$p,$$
$$\bot \leftarrow p \wedge \bot.$$

The second line is a tautology, so that the only minimal model of this program is $\{p\}$. It does not satisfy the condition characterizing Case 1. *Case 2:* $q \notin I$. The reduct is

$$p,$$
$$\bot \leftarrow p \wedge \top.$$

This program is inconsistent. Consequently the given program has no stable models.

S.10.6. (a) Take any interpretation I, and consider four cases. *Case 1:* $p, q \in I$. The reduct is

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

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

$$p \leftarrow \top,$$
$$q \leftarrow \bot,$$
$$r \leftarrow p,$$
$$r \leftarrow q.$$

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

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

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.10.7. (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.11.1. The stable model $\{p\}$ of the one-rule program p is not a stable model of the one-rule program $\neg\neg p$. Indeed, the reduct of $\neg\neg p$ relative to $\{p\}$ is $\top$, and $\{p\}$ is not a minimal model of $\top$.

S.11.2. Consider the programs

$$\begin{array}{l} q, \\ \neg q \leftarrow \neg p \end{array} \tag{53}$$

and

$$q,$$
$$p \leftarrow q.$$

The stable model $\{p, q\}$ of the latter is not a stable model of the former. Indeed, the reduct of (53) relative to $\{p, q\}$ is

$$q,$$
$$\bot \leftarrow \bot,$$

and its only minimal model is $\{q\}$.

S.11.3. The given rules are positive and equivalent to each other.

S.11.4. True. Indeed,

$$(F \wedge (G \vee H))^I \text{ is } F^I \wedge (G^I \vee H^I),$$
$$((F \wedge G) \vee (F \wedge H))^I \text{ is } (F^I \wedge G^I) \vee (F^I \wedge H^I),$$

and these reducts are equivalent.

S.11.5. True. Indeed, consider any interpretation I. If $p \in I$ then

$$(p \leftarrow \neg\neg p)^I \text{ is } p \leftarrow \top,\ (p \vee \neg p)^I \text{ is } p \vee \bot;$$

both reducts are equivalent to p. If $p \notin I$ then

$$(p \leftarrow \neg\neg p)^I \text{ is } p \leftarrow \bot,\ (p \vee \neg p)^I \text{ is } p \vee \top;$$

both reducts are equivalent to $\top$.

S.11.6. True, because the reduct $F^I \leftarrow F^I \wedge G^I$ of this rule relative to any interpretation I is a tautology.

S.12.1. (a)

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

(b)

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

(c)

$$p(a),$$
$$q(b),$$
$$r(v) \leftarrow p(v) \qquad (v \in \mathbf{S} \cup \mathbf{Z} \setminus \{b\}).$$

(d) The minimal model $\{p(a), q(b), r(a)\}$ is different from (37). Consequently (37) is not a stable model of the program.

S.12.2. (a)

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

(b)

$$\begin{array}{l} p(1) \wedge p(2) \wedge p(3), \\ q(2) \leftarrow \neg p(2), \\ q(3) \leftarrow \neg p(3), \\ q(4) \leftarrow \neg p(4). \end{array}$$

(c)

$$\begin{array}{l} p(1) \wedge p(2) \wedge p(3), \\ q(2) \leftarrow \bot, \\ q(3) \leftarrow \bot, \\ q(4) \leftarrow \top. \end{array}$$

(d)

$$\begin{array}{l} p(1) \wedge p(2) \wedge p(3), \\ q(4). \end{array}$$

(e) The minimal model $\{p(1), p(2), p(3), q(4)\}$ is the same as (43). It follows that (43) is a stable model of the program.

S.12.3. (a)

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

(b)

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

(c)

$$q(v) \leftarrow p(v) \qquad (v \in \mathbf{S} \cup \mathbf{Z}).$$

(d) The minimal model $\emptyset$ is different from (46). Consequently (46) is not a stable model of the program.

S.12.4. The propositional image of program (47) is

$$\begin{array}{l} p(a), \\ q(v) \leftarrow \neg p(v) \qquad \text{for all } v \in \mathbf{S} \cup \mathbf{Z}. \end{array}$$

Its reduct relative to the given set of atoms is

$$\begin{array}{ll} p(a), & \\ q(a) \leftarrow \bot, & \\ q(v) \leftarrow \top & \text{for all } v \in \mathbf{S} \cup \mathbf{Z}, \end{array}$$

or, equivalently,

$$\begin{array}{ll} p(a), & \\ q(v) & \text{for all } v \in \mathbf{S} \cup \mathbf{Z} \end{array}$$

This set of atoms is its own minimal model.