Use splitting sets to compute stable models of the following:

1. \[ p \leftarrow q, \text{not } s \]
   \[ r \leftarrow p, \text{not } q, \text{not } s \]
   \[ s \leftarrow \text{not } q \]
   \[ q \leftarrow \text{not } s \]

2. \[ p \]
   \[ r \leftarrow p, \text{not } q \]
   \[ q \leftarrow p, \text{not } r \]
   \[ s \leftarrow r, \text{not } s \]

3. (a) \[ p \leftarrow \text{not } q \]
   \[ r \leftarrow q \]
   (b) \[ p \leftarrow \text{not } q \]
   \[ r \leftarrow q \]
   \[ r \]
   (c) \[ p \leftarrow \text{not } q \]
   \[ r \leftarrow q \]
   \[ r \]

4. \text{can\_fly} \leftarrow \text{bird}, \text{not ab\_bird}
   \text{can\_fly} \leftarrow \text{bird}, \text{ab\_bird}
   \text{ab\_bird} \leftarrow \text{ostrich}
   \text{bird} \leftarrow \text{ostrich}

5. Suppose that logic program \( P \) contains a clause
   \[ p \leftarrow r, \text{not } p \]
   where \( p \) does not occur anywhere else in \( P \). (In particular, \( p \) is not defined in \( P \).)

Show that there is no stable model of \( P \) that contains \( r \).

Question 1
Let’s take the splitting set \( U = \{ q, s \} \). \( \{ p, q, s \} \) is also a splitting set. We’ll look at that one later.

\[
\begin{align*}
p & \leftarrow q, \text{not } s \\
r & \leftarrow p, \text{not } q, \text{not } s \\
s & \leftarrow \text{not } q \\
q & \leftarrow \text{not } s \\
\end{align*}
\]

\[ U = \{ q, s \} \]

The bottom part has two stable models: \( \{ q \} \) and \( \{ s \} \). Consider them in turn.

1. \( \{ q \} \) Simplifying the top part gives the program \( \{ p \} \). This obviously has just one stable model, \( \{ p \} \).
   A stable model of the original program is therefore \( \{ q \} \cup \{ p \} \).

2. \( \{ s \} \) Simplifying the top part gives \( \emptyset \). This obviously has just one stable model, \( \emptyset \).
   A stable model of the original program is \( \{ s \} \cup \emptyset = \{ s \} \).

There are no other stable models.

Just to check, suppose we started with the other splitting set \( U = \{ p, q, s \} \).

\[
\begin{align*}
 p & \leftarrow q, \text{not } s \\
r & \leftarrow p, \text{not } q, \text{not } s \\
s & \leftarrow \text{not } q \\
q & \leftarrow \text{not } s \\
\end{align*}
\]

\[ U = \{ p, q, s \} \]

We need to find stable models of the bottom part. We can split again:

\[
\begin{align*}
 p & \leftarrow q, \text{not } s \\
s & \leftarrow \text{not } q \\
q & \leftarrow \text{not } s \\
\end{align*}
\]

\[ U' = \{ q, s \} \]

One can see there are two stable models: one is \( \{ p, q \} \) and the other is \( \{ s \} \).

In both cases, simplifying the top part of the original program gives us \( \emptyset \). So the original program has two stable models: \( \{ p, q \} \) and \( \{ s \} \). (Same as above.)
Question 2 (Note in passing that there is no stable model containing \( r \). Why? See Question 5.)

There are two splitting sets: \( \{ p \} \) and \( \{ p, q, r \} \). The first seems easier to handle. So we have:

\[
\begin{align*}
    r & \leftarrow p, \not q \\
    q & \leftarrow p, \not r \\
    s & \leftarrow r, \not s \\
    p & \quad \quad U = \{ p \}
\end{align*}
\]

The bottom part obviously has one stable model: \( \{ p \} \). Simplifying the top part gives:

\[
\begin{align*}
    r & \leftarrow \not q \\
    q & \leftarrow \not r \\
    s & \leftarrow r, \not s
\end{align*}
\]

This program can be split thus:

\[
\begin{align*}
    s & \leftarrow r, \not s \\
    r & \leftarrow \not q \\
    q & \leftarrow \not r \\
    U' & = \{ q, r \}
\end{align*}
\]

There are two stable models for the bottom part: \( \{ q \} \) and \( \{ r \} \). Simplifying \( \{ s \leftarrow r, \not s \} \) with \( \{ q \} \) relative to \( U' = \{ q, r \} \) gives \( \emptyset \). This has one stable model, \( \emptyset \). So one stable model for the original program is \( \emptyset \cup \{ q \} \cup \{ p \} = \{ p, q \} \).

Simplifying \( \{ s \leftarrow r, \not s \} \) with \( \{ r \} \) relative to \( U' = \{ q, r \} \) gives \( \{ s \leftarrow \not s \} \). This has no stable model. (Check: there are only two candidates, \( \{ s \} \) and \( \emptyset \), and neither is stable.) So \( \{ r \} \) for the bottom part does not yield a stable model for the original program.

There is only one stable model for the original program, viz. \( \{ p, q \} \).

(We already knew there could not be one containing \( r \).) (Thanks to Tim Pierce and Robin Bennett for pointing out some errors in earlier versions of this handout.)

Question 3 In each case take the splitting set \( U = \{ r, q \} \).

1. \( p \leftarrow \not q \\
    r \leftarrow q \\
    U = \{ r, q \}
\]

The (unique) stable model of the bottom part is \( \emptyset \).

Simplifying the top part gives \( \{ p \} \). This has one stable model \( \{ p \} \).

So the only stable model of the original program is \( \emptyset \cup \{ p \} = \{ p \} \).

2. \( p \leftarrow \not q \\
    r \leftarrow q \\
    U = \{ r, q \}
\]

The (unique) stable model of the bottom part is \( \{ r \} \).

Simplifying the top part gives \( \{ p \} \). This has one stable model \( \{ p \} \).

So the only stable model of the original program is \( \{ r \} \cup \{ p \} = \{ r, p \} \).

3. \( p \leftarrow \not q \\
    r \leftarrow q \\
    \emptyset \\
    U = \{ r, q \}
\]

The (unique) stable model of the bottom part is \( \{ r, q \} \).

Simplifying the top part gives \( \emptyset \). This has one stable model, \( \emptyset \).

So the only stable model of the original program is \( \emptyset \cup \{ r, q \} = \{ r, q \} \).

Question 4 Take the splitting set \( U = \{ \text{bird, ostrich, ab\_bird} \} \).

\[
\begin{align*}
    \text{can\_fly} & \leftarrow \text{bird, not ab\_bird} \\
    \text{can\_fly} & \leftarrow \text{bird, ab\_bird} \\
    \text{can\_fly} & \leftarrow \text{ab\_bird} \leftarrow \text{ostrich} \\
    \text{bird} & \leftarrow \text{ostrich} \\
    U & = \{ \text{bird, ostrich} \}
\end{align*}
\]

The stable model of the bottom part is obviously \( \{ \text{bird} \} \). Simplifying the top part with \( \{ \text{bird} \} \) and relative to \( U = \{ \text{bird, ostrich} \} \) gives:

\[
\begin{align*}
    \text{can\_fly} & \leftarrow \not \text{ab\_bird} \\
    \text{can\_fly} & \leftarrow \text{ab\_bird} \\
    \text{can\_fly} & \leftarrow \text{can\_fly}
\end{align*}
\]

This has the same form as part (b) of the previous question. There is thus one stable model, \( \{ \text{can\_fly, can\_fly} \} \), and so one stable model for the original program: \( \{ \text{bird} \} \cup \{ \text{can\_fly, can\_fly} \} \).

(As an expression of default rules about flying birds and ostriches, the above formulation is obviously inadequate.)

Question 5 \( P \) contains a clause

\[
\begin{align*}
    p & \leftarrow r, \not p \\
    r & \leftarrow q \\
    U & = \{ r, q \}
\end{align*}
\]

where \( p \) does not occur anywhere else in \( P \). (In particular, \( p \) is not defined in \( P \).)

Clearly \( P \) can be split with the clause above in the top part and everything else in \( P \) in the bottom part. (The splitting set is all atoms of \( P \) except \( p \).)

If \( r \) belongs to a stable model of \( P \), it must belong to a stable model of the bottom part.

Suppose there is such a model. Then simplifying the top part using this stable model will give us

\[
\begin{align*}
    \{ p \leftarrow \not p \}
\end{align*}
\]

But that program has no stable model. (Easy to check.)