491 Knowledge Representation

Tutorial Exercise

Splitting Sets

Use splitting sets to compute stable models of the following:

1. $p \leftarrow q,$ `not` s
   $r \leftarrow p,$ `not` $q,$ `not` s
   $s \leftarrow$ `not` q
   $q \leftarrow$ `not` s

2. p
   $r \leftarrow p,$ `not` q
   $q \leftarrow p,$ `not` r
   $s \leftarrow r,$ `not` s

3. (a) $p \leftarrow$ `not` q
   $r \leftarrow q$

   (b) $p \leftarrow$ `not` q
   $r \leftarrow q$
   r

   (c) $p \leftarrow$ `not` q
   $r \leftarrow q$
   r
   q

4. *can_fly* $\leftarrow$ *bird,* `not` *ab_bird*
   cant_fly $\leftarrow$ *bird, ab_bird*
   ab_bird $\leftarrow$ *ostrich*
   bird $\leftarrow$ *ostrich*
   bird
   cant_fly

5. Suppose that logic program P contains a clause

$$p \leftarrow r, \texttt{not}\ p$$

   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.

491 Knowledge Representation

Tutorial Exercise

Splitting Sets
SOLUTIONS

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.

$p \leftarrow q,$ `not` s
$r \leftarrow p,$ `not` $q,$ `not` s
____
$s \leftarrow$ `not` q $\qquad U = \{q, s\}$
$q \leftarrow$ `not` s

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

1. $\{q\}$ Simplifying the top part gives the program $\{p \leftarrow\}$. 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 progam is $\{s\} \cup \emptyset = \{s\}$.

There are no other stable models.

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

$r \leftarrow p,$ `not` $q,$ `not` s
____
$p \leftarrow q,$ `not` s $\qquad U = \{p, q, s\}$
$s \leftarrow$ `not` q
$q \leftarrow$ `not` s

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

$p \leftarrow q,$ `not` s
____
$s \leftarrow$ `not` q $\qquad U' = \{q, s\}$
$q \leftarrow$ `not` 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:

$$\begin{array}{ll} r \leftarrow p,\ \texttt{not}\ q & \\ q \leftarrow p,\ \texttt{not}\ r & \\ s \leftarrow r,\ \texttt{not}\ s & \\ \hline p & U = \{p\} \end{array}$$

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

$$\begin{array}{l} r \leftarrow \texttt{not}\ q \\ q \leftarrow \texttt{not}\ r \\ s \leftarrow r,\ \texttt{not}\ s \end{array}$$

This program can be split thus:

$$\begin{array}{ll} s \leftarrow r,\ \texttt{not}\ s & \\ \hline r \leftarrow \texttt{not}\ q & U' = \{q, r\} \\ q \leftarrow \texttt{not}\ r & \end{array}$$

There are two stable models for the bottom part: $\{q\}$ and $\{r\}$.

Simplifying $\{s \leftarrow r, \texttt{not}\ s\}$ with $\{q\}$ relative to $U' = \{q, r\}$ gives $\{\}$. This has one stable model, $\emptyset$. So one stable model for the original program is $\emptyset \cup \{q\} \cup \{p\} = \{p, q\}$.

Simplifying $\{s \leftarrow r, \texttt{not}\ s\}$ with $\{r\}$ relative to $U' = \{q, r\}$ gives $\{s \leftarrow \texttt{not}\ s\}$. 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. $$\begin{array}{ll} p \leftarrow \texttt{not}\ q & \\ \hline r \leftarrow q & U = \{r, q\} \end{array}$$

   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. $$\begin{array}{ll} p \leftarrow \texttt{not}\ q & \\ \hline r \leftarrow q & U = \{r, q\} \\ r & \end{array}$$

   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. $$\begin{array}{ll} p \leftarrow \texttt{not}\ q & \\ \hline r \leftarrow q & U = \{r, q\} \\ r & \\ q & \end{array}$$

   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 $\{r, q\} \cup \emptyset = \{r, q\}$.

Question 4 Take the splitting set $U = \{bird, ostrich\}$. One could also use the splitting set $\{bird, ostrich, ab\_bird\}$.

$$\begin{array}{ll} can\_fly \leftarrow bird,\ \texttt{not}\ ab\_bird & \\ cant\_fly \leftarrow bird,\ ab\_bird & \\ cant\_fly & \\ ab\_bird \leftarrow ostrich & \\ \hline bird \leftarrow ostrich & U = \{bird, ostrich\} \\ bird & \end{array}$$

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

$$\begin{array}{l} can\_fly \leftarrow \texttt{not}\ ab\_bird \\ cant\_fly \leftarrow ab\_bird \\ cant\_fly \end{array}$$

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

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

Question 5 P contains a clause

$$p \leftarrow r,\ \texttt{not}\ p$$

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

$$\{p \leftarrow \texttt{not}\ p\}$$

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