CS240B MIDTERM EXAM: Open Book, 110 Minutes

*Attach extra pages as needed. Write your name and ID on the extra pages.*

*Please, write neatly.*

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(32%)</td>
</tr>
<tr>
<td>2</td>
<td>(32%)</td>
</tr>
<tr>
<td>3</td>
<td>(36%)</td>
</tr>
<tr>
<td>Total</td>
<td>(100%)</td>
</tr>
</tbody>
</table>

Extra credit: 10%

Adjusted Total:
Problem 1: 32 points

Given the following program $P$:

$$
\begin{align*}
  a & \leftarrow \neg b. \\
  b & \leftarrow c, \neg a. \\
  c & \leftarrow b.
\end{align*}
$$

(1) Give all the models and fixpoints for $P$.
(2) Find all minimal models and minimal fixpoints for $P$.
(3) Find all stable models for $P$.
(4) Give $T_P \uparrow \omega(\emptyset)$.

Answer

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>M</th>
<th>Fx</th>
<th>mM</th>
<th>mFx</th>
<th>Stbl</th>
</tr>
</thead>
<tbody>
<tr>
<td>abc</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>ab</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>bc</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ac</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

The stability transformation for $\{b, c\}$ yields:

$$
\begin{align*}
  b & \leftarrow c. \\
  c & \leftarrow b.
\end{align*}
$$

The the least model for this program is the empty set.

The stability transformation for $\{a\}$ yields:

$$
\begin{align*}
  a & \leftarrow . \\
  c & \leftarrow b.
\end{align*}
$$

The the least model for this program is $\{a\}$.

(4) $T_P \uparrow \omega(\emptyset) = \{a\}$
Problem 2: 32 points

1. \(\text{\textit{L}}\) is a nonempty list of numbers given as the first argument of the query: \(\text{\textit{findmax}}(\text{\textit{L}}, \text{\textit{M}})\).
   Write a recursive predicate that finds the maximum value in the list and returns it as \(\text{\textit{M}}\) (the second argument in the query).

2. Describe how your query will be compiled, and rewrite the original program according to the recursive compilation method used by the compiler.

   \[
   \begin{align*}
   \text{\textit{findmax}}(\text{\textit{L}}, \text{\textit{M}}) & \leftarrow \text{\textit{L}} = \text{\textit{X}}|\text{\textit{L1}}, \text{\textit{findmax}}(\text{\textit{L1}}, \text{\textit{M1}}), \text{\textit{larger}}(\text{\textit{X}}, \text{\textit{M}}, \text{\textit{M1}}). \\
   \text{\textit{findmax}}(\text{\textit{M}}, \text{\textit{M}}) & \\
   \text{\textit{larger}}(\text{\textit{X}}, \text{\textit{M1}}, \text{\textit{M1}}) & \leftarrow \text{\textit{M1}} > \text{\textit{X}}.
   \text{\textit{larger}}(\text{\textit{X}}, \text{\textit{M1}}, \text{\textit{X}}) & \leftarrow \text{\textit{M1}} \leq \text{\textit{X}}.
   \end{align*}
   \]

2. Binding passing property: \(\text{\textit{findmax}}^{bf} \rightarrow \text{\textit{findmax}}^{bf}\)

Recursive method: the magic-set could be used:

   \[
   \begin{align*}
   \text{\textit{m.findmax}}(\text{\textit{L}}) & \\
   \text{\textit{m.findmax}}(\text{\textit{L1}}) & \leftarrow \text{\textit{m.findmax}}(\text{\textit{L}}), \text{\textit{L}} = \text{\textit{X}}|\text{\textit{L1}}.
   \text{\textit{m.findmax}}(\text{\textit{M}}) & \leftarrow \text{\textit{m.findmax}}(\text{\textit{L1}}).
   \text{\textit{findmax}}(\text{\textit{L}}, \text{\textit{M}}) & \leftarrow \text{\textit{L}} = \text{\textit{X}}|\text{\textit{L1}}, \text{\textit{findmax}}(\text{\textit{L1}}, \text{\textit{M1}}), \text{\textit{larger}}(\text{\textit{X}}, \text{\textit{M}}, \text{\textit{M1}}), \text{\textit{m.findmax}}(\text{\textit{L}}).
   \end{align*}
   \]

The supplementary magic set is preferable:

   \[
   \begin{align*}
   \text{\textit{m.findmax}}(\text{\textit{L}}) & \\
   \text{\textit{spmm.findmax}}(\text{\textit{L}}, \text{\textit{L1}}) & \leftarrow \text{\textit{m.findmax}}(\text{\textit{L}}), \text{\textit{L}} = \text{\textit{X}}|\text{\textit{L1}}.
   \text{\textit{m.findmax}}(\text{\textit{L1}}) & \leftarrow \text{\textit{spmm.findmax}}(\text{\textit{L}}, \text{\textit{L1}}).
   \text{\textit{m.findmax}}(\text{\textit{M}}) & \leftarrow \text{\textit{m.findmax}}(\text{\textit{L1}}).
   \text{\textit{findmax}}(\text{\textit{L}}, \text{\textit{M}}) & \leftarrow \text{\textit{findmax}}(\text{\textit{L1}}, \text{\textit{M1}}), \text{\textit{spmm.findmax}}(\text{\textit{L}}, \text{\textit{L1}}), \text{\textit{larger}}(\text{\textit{X}}, \text{\textit{M}}, \text{\textit{M1}}).
   \end{align*}
   \]

These are linear rules; thus in the differential fixpoint the recursive predicates will be replaced by their deltas.
Problem 3: 36 points. You have a database table with the time series

\[ \text{temperature}(\text{Day int, Degrees int}) \]

that shows the noon temperature for each day since January 1, 1900. Say that the days are numbered consecutively; thus 01/01/1900 is day 1, and 01/01/1991 is day 366, and so on. The records are stored by ascending day.

1. Write an efficient ATLAS program with nonblocking UDAs to compute:

   (i) the hottest temperature in the last 1000 days (i.e., the last 1000 records), with

   (ii) the day in which this max last occurred (several days can share the with same max temperature)

Suggestion: write a maxpair UDA to search a table for the max temperature and the last day in which it occurred. This can be used in another UDA that manages the computation of maxpair on the memorized values in the window. The max should be recomputed only when needed.

2. Write these two queries in SQL:1999. Suggestion: use the window construct to express (i) and then see how to extend this query to return (ii).

```sql
AGGREGATE maxpair(iValue int, iPoint int):(mValue int, mPoint int)
{
    TABLE mpair(thevalue int, thepoint int);
    INITIALIZE: { INSERT INTO mpair VALUES (iValue, iPoint) }
    ITERATE: { UPDATE mvalue SET thevalue = iValue, thepoint=iPoint
               WHERE iValue > thevalue }
    TERMINATE: { INSERT INTO RETURN
                 SELECT thevalue, thepoint FROM mpair }
}

AGGREGATE lastmaxpair(iTemp int, iDay int):(mTemp int, mDay int)
{
    TABLE window(thevalue int, thepoint int);
    INITIALIZE: { INSERT INTO window VALUES (iTemp, iDay) }

    ITERATE: { DELETE window WHERE thepoint<iDay-1000;
               INSERT INTO window VALUES (iTemp, iDay);
               INSERT INTO RETURN
               SELECT maxpair(thevalue, thepoint) FROM window }
}

SELECT lastmaxpair(Day, Degrees)
FROM temperatures
```
SQL:1999

1. SELECT max(Degrees) OVER (ORDER BY Day ROWS 1000 PRECEDING)
   FROM temperatures

2. We can use the previous query in a table expression:

   SELECT W.Day, T.Degrees, max(T.Day)
   FROM temperatures AS T,
   (SELECT Day, max(Degrees) OVER (ORDER BY Day ROWS 1000 PRECEDING)
    FROM temperatures) AS W(CDay, LastMax)
   WHERE W.LDay >= T.Day AND W.Lday-1000 <= T.DAY
   GROUP BY W.Day, T.Degrees

Without an increasing sequence# attached to each day, things would have been harder.
Extra Credit Problem: 10 points

Rewrite the UDA from the previous problem to minimize the use of memory (e.g., rather than keeping all temperatures in the window, just keep those that have a chance to become the max).

Answer: For each new day, eliminate from the windows those with lesser or equal temperature.

... no change ...
ITERATE: { DELETE window
    WHERE thepoint<iDay-1000 OR thevalue <= iTemp;
... no change ...