Computations

1. The logical operator, xor, is defined as follows.

\[ a \ XOR \ b = (\text{not} \ a \ AND \ b) \ OR (a \ AND \ \text{not} \ b) \]

The truth table for \( c = a \ XOR \ b \) is shown at right.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Note that the number of 1s in each row is even.

Write a computation, \( \text{xor}(a, b, c) \), which can accept the most possible subsets of \( \{a, b, c\} \) as inputs.

2. Write a computation which gives all possible results from the relationship among the integers dividend, divisor, quotient, and remainder.

3. a) and b) Write two computations, one for each of the relationships

\[ d = d_0 + v_0 \times t + a \times t \times t/2 \]

and

\[ v = v_0 + a \times t \]

c) Discuss how these might be combined, in a future implementation of Aldat, to solve the two equations together: discuss each possible invocation.

4. Write the post update handlers that will display “Trigger” and “New” for the code

\[ \text{update} \ R \ \text{change} \ X \leftarrow "c" \ \text{using} \ ijoin \ P; \]
\[ \text{update} \ R \ \text{add} \ Q; \]
\[ \text{update} \ R \ \text{delete} \ S; \]

and show the results produced by relix for

\[
\begin{array}{ccc}
R(X & Y) & P(X & Y) & Q(X & Y) & S(X & Y) \\
a & 1 & a & 1 & a & 3 & b & 1 \\
a & 2 & b & 1
\end{array}
\]

5. Define a computation, \( \text{comp} \ X(R, S, T, B, C, D) \), which puts in \( T \) all pairs of values \( B \) from \( R \) and \( D \) from \( S \), which are associated with the same sets of values of \( C \) from both \( R \) and \( S \). Invoke your computation on the relations \( R(A, B, C) \) and \( S(C, D, E) \). Write a second alt block in your computation which constructs as much of \( S \) as possible given \( R \) and \( T \).

6. Write an event handler which prints out a sequence of consecutive integers from 4 to 1 then stops, using only a relation counter which is always a singleton. Write the code to invoke it.
requires prior specific permission and/or fee. Request permission to republish from: T. H. Merrett, School of Computer Science, McGill University, fax 514 398 3883.

The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.