<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example:
- Quite restrictive.

Procedure:
- BEGIN TRANSACTION
- Assume transactions are numbered sequentially.
- Acquire information.
- Use simple way of ensuring correctness.
- One single way to ensure correctness.
procedure READ: current value (date, h, t, f, t).

... procedure...

This announcement is called "mail point". Its modification, and announce change it is finished. Pendency versions of every variable it is finished, mark point: each transaction should create no.

is no longer pending.

transaction has reached its mark point or begin ready if inputs until the procedure...
end procedure

re Guilin id

release (outcome.recor + lock)

mark INC \(\rightarrow\) NurK

id.outcome.record + id + Econ

allocate id.outcome.record

Acquire (outcome + record + squence)

procedure

NEW.outcome + record (5 thr)

and front i =

return i =

if (PENDING)

procedure (id.outcome + record + thr)

mark entry = MARKED

wait until (previous + id.outcome + record + id =

I New.outcome + record (PENDING)

BEGIN TRANSACTION

18
procedure MARK-POINT-ANNOUNCE (tid) 

sets Mark-state = MARKED .

end proc
EndTRANSACTION;

if x < y then
  WRITE-VALUE (credit - account, y, value, myid)
  y value = y value + amount
  y.value = READ-CURRENT-VALUE (credit - account, myid)

  WRITE-VALUE (debit - account, x, value, myid)
  x.value = x.value - amount
  x.value = READ-CURRENT-VALUE (debit - account, myid)

  MARK-POINT ANNOUNCE (myid).
  NEW-VERSION (credit - account, myid).
  NEW-VERSION (debit - account, myid).

my-id -> BEGIN-TRANSACTION;

procedure TRANSFER (my, debit - account, y, 1.8)
procedure  READ-CURRENT-VA-VALUE (var decl-in, volts,  

    if procedure

    if cont. be churn

    return v-value

    caller-id

    max (v-high-water-marke

    v-high-water-marke

    if contains them

    abort, wait for veterinarian to continue

    if pending then

    examine v-acidish-kind, outcome: recover.

    if v-acidish-kind > caller-id from sin.

    u = previous version of decl-id

    story of end of decl-id, r-u.

    caller-id

    end procedure
If not found, 

if action-id = caller-id: 

write-value (rt: data-id, new-value); 

locate version of data-id. 

return new-value; 

procedure write-value (rt: data-id, new-value: 

add new version v to end of data-id. 

then add another task to manipulate the 

action-id; 

else (caller-id < latest version[data-id]. Confined with 

if (caller-id < data-id. Has hot spot menu) 

procedure new-version (rt: data-id, caller-id);