(Brock-Ackerman) The following picture shows a network of communicating processes.

```
   a! 0
   |     b
   |     ↓
   choose <- c?. b! c
   |     d
```

The formal description of this network is

\[ \text{chan } a, b, c : a ! 0 \parallel \text{choose } \parallel (c ?. b ! c) \]

Formally define \( \text{choose} \) as follows:

\[ \text{choose } \equiv (a ?. (c ! 0 \parallel d ! 0)) \lor (b ?. (c ! 1 \parallel d ! 1)) \]

Now the network is

\[ \text{chan } a, b, c : a ! 0 \parallel \text{choose } \parallel (c ?. b ! c) \]

\[ \equiv \exists a, r, a', w, w', b, r, b', w, w', c, r, c', w, c'. \]

\[ \begin{align*}
\text{ta} &= \exists a, r, a', w, w', b, r, b', w, w', c, r, c', w, c'. \]
\[ & \quad \land (ta' = w' = t) \land \exists a ! 0 \land w = 1 \\
& \quad \lor (ta = tc = t + 1) \land r = wc = 1 \land \exists c ! 0 \land \exists d = w = 1 \\
& \quad \lor (ta = tc = t + 1) \land r = wc = 1 \land \exists c ! 0 \land \exists d = w = 1 \\
& \quad \lor (tb = t + 1) \land c = wc = 1 \land \exists b ! 0 \land \exists c ! 0 \\
& \quad \lor (t' = \text{MAX } [ta; tc; tb]) \quad \text{use One-Point laws to eliminate most quantifiers} \\
\end{align*} \]

Except for time, all processes in independent compositions change different variables, so \( \parallel \) is easily replaced by conjunction.

Also, make all substitutions indicated by assignments.

\[ \begin{align*}
\text{ta} &= \exists a, r, a', w, w', b, r, b', w, w', c, r, c', w, c'. \]
\[ & \quad \land (ta' = w' = t) \land \exists a ! 0 \land w' = 1 \\
& \quad \lor (ta = tc = t + 1) \land \exists d = w = 1 \\
& \quad \lor (ta = tc = t + 1) \land \exists d = w = 1 \\
& \quad \lor (tb = t + 1) \land c = wc = 1 \land \exists b ! 0 \land \exists c ! 0 \\
& \quad \lor (t' = \text{MAX } [ta; tc; tb]) \quad \text{move the conjunctions into the disjunction} \\
\end{align*} \]

There is probably a better way to do this question by using laws of programs and not translating to ordinary logic.
as in part (a), choose either reads from \( a \) and then outputs a 0 on \( c \) and \( d \), or reads from \( b \) and then outputs a 1 on \( c \) and \( d \). But this time the choice is not made freely; choose reads from the channel whose input is available first (if there's a tie, then take either one).

We define choose as follows:

\[
\text{choose} = (a \lor T a r a \leq T b r b) \land (a? \cdot (c! 0 \parallel d! 0))
\]

\[
\lor (b \lor T b r b \leq T a r a) \land (b? \cdot (c! 1 \parallel d! 1))
\]

Now the network is

\[
\begin{align*}
\text{chan } a, b, c & \quad a! 0 \parallel \text{choose} \parallel (c?. b! c) \\
= & \quad \exists b a, T a, ra, ra', wa, wa', \exists b, T b, rb, rb', wb, wb', \exists c, T c, rc, rc', wc, wc'. \\
ra = 0. \quad wa = 0. \quad rb = 0. \quad wb = 0. \quad wc = 0. \\
\text{if } b a w a = 0 \land T a w a = t \land (wa = wa + 1) \land (t := \text{max } t (T a r a + 1)). \quad ra := ra + 1. \\
\text{then } (M c w c = 0 \land T c w c = t \land (wc = wc + 1)) \lor (M d w d = 0 \land T d w d = t \land (wd = wd + 1)) \\
\lor (t := \text{max } t (T b r b + 1)). \quad \text{then } (M c w c = 1 \land T c w c = t \land (wc = wc + 1)) \lor (M d w d = 1 \land T d w d = t \land (wd = wd + 1)) \\
\text{else } (t := \text{max } t (T c r c + 1)). \quad \text{then } (M c w c = 1 \land T c w c = t \land (wc = wc + 1)) \lor (M d w d = 1 \land T d w d = t \land (wd = wd + 1)) \\
\text{end if}
\end{align*}
\]

Excepts for time, all processes in independent compositions change different variables, so \( \parallel \) is easily replaced by conjunction.

Also, make all substitutions indicated by assignments.

\[
\begin{align*}
= & \quad \exists b a, T a, ra, ra', wa, wa', \exists b, T b, rb, rb', wb, wb', \exists c, T c, rc, rc', wc, wc'. \\
\exists a, t, c, t b.
\end{align*}
\]

\[
\begin{align*}
t a &= T a 0 = t \land b a 0 = 0 \land wa' = 1 \\
\land (T a 0 = t \land b a 0 = 0 \land wa' = 1 \\
\land tc = T c 0 = T d w d = T a 0 + 1 \land ra' = wc' = 1 \land M c 0 = M d w d = 0 \land wd' = wd + 1 \\
\lor (T b 0 = t \land T b 0 = T a r a \\
\land tc = T c 0 = T d w d = T b 0 + 1 \land rb' = wc' = 1 \land M c 0 = M d w d = 1 \land wd' = wd + 1) \\
\land t b = T b 0 = T c 0 + 1 \land rc' = wb' = 1 \land M b 0 = M c 0 \\
\land t' = \text{MAX } [t a; t c; t b]
\end{align*}
\]

use the One-Point laws to eliminate most quantifiers

\[
\begin{align*}
= & \quad \exists c, t b.
\end{align*}
\]

\[
\begin{align*}
t c &= T d w d = t + 1 \land M d w d = 0 \land wd' = wd + 1 \\
\lor (t b \leq t \land t b \leq t) \\
\land t b = t c + 1 \\
\land t' = \text{MAX } [t; t c; t b]
\end{align*}
\]

simplify the two minor disjunctions

\[
\begin{align*}
= & \quad \exists c, t c.
\end{align*}
\]

\[
\begin{align*}
t c &= T d w d = t + 1 \land M d w d = 0 \land wd' = wd + 1 \land t b = t c + 1 \land t' = \text{MAX } [t; t c; t b] \\
\lor t b \leq t \land t c = T d w d = t b + 1 \land M d w d = 1 \land wd' = wd + 1 \land \text{now we can eliminate } t c \text{ and } t b \text{ in each disjunct separately}
\end{align*}
\]

\[
\begin{align*}
= & \quad T d w d = t + 1 \land M d w d = 0 \land wd' = wd + 1 \land t' = t + 2 \\
\lor \infty t \land T a 0 = \infty \land M d w d = 1 \land wd' = wd + 1 \land t' = \infty \\
= & \quad (t := t + 1. \quad d! 0. \quad t := t + 1) \lor t = t + \infty \land (d! 1)
\end{align*}
\]

If the computation starts before time \( \infty \), the output is definitely 0. Again, there is probably a better way to do this question by using laws of programs and not translating to ordinary logic.