1. An alarm clock is defined with the following property:

When the alarm event occurs, a bell should ring for either 30 seconds or until a stop button is pressed, whichever happens first.

Express this property in RTL. Let AC, ↑BELL, ↓BELL, and STOP be the names of the alarm event, start_bell_ringing, stop_bell_ringing, and stop_button events, respectively.

Assumptions: The ↑BELL events are separated by at least 30 seconds. The alarm event and start_bell_ringing events are matched.

∀x@(AC, x) = @(↑ BELL, x))∧

(∃y@((↑ BELL, x) ≤ @(STOP, y)) ≤ @(↑ BELL, x) + 30)∧

((@(↑ BELL, x) = @(STOP, y))) ∨

(¬∃y@((↑ BELL, x) ≤ @(STOP, y)) ≤ @(↑ BELL, x) + 30)∧

((@(↑ BELL, x) = @(↑ BELL, x) + 30))

2. Let the train crossing gate operate so that the gate will close within 5 time units of receiving a close command, provided that an intervening open command has not been issued. The transition events are classes in the RTL sense. The constraint may be stated precisely in terms of these events:

If a cg command is issued at some time \( t_1 \), then either

(a) the gate will close (c-c event) at some time \( t_2 \leq t_1 + 5 \), and there will have been no og command issued between \( t_1 \) and \( t_2 \), or

(b) the gate receives an og command at some time \( t_2 \leq t_1 + 5 \) and the gate will not close (no c-c event) at any time between \( t_1 \) and \( t_2 \).

∀i(∃j@(cg, i) < @(c - c, j) < @(cg, i) + 5) ∧ ¬∃k@(cg, i) < @(cg, k) < (c - c, j)) ∨

∃k@(cg, i) < @(og, k) < @(cg, i) + 5 ∧ ¬∃j@(cg, i) < @(c - c, j) < @(og, k))

If an og command occurs, then the gate may or may not close. The specification is a little strange in this respect.

3. Computer question. This solution was presented by Jong Hee Kang.
Legend:
- \(<\text{Instr}>\) : operation (Instr) = string
- \(\text{OPND}\) : operand (Instr)
- \(\text{OPTR}\) : operation (Instr)
- \(\text{ALU ops}\) : "add", "subtract", "and", or "or"

Assumption:
- PC increment can be done without using ALU.
- Branch instruction takes 1 time unit.
- "Load" loads value to AC, "Store" stores AC to memory.
- ALU operations store result to AC.