MODULE *GalsIzhikevich*

EXTENDS *FiniteSets*, *Integers*, *Sequences*, *RandMessages*

CONSTANTS *Nodes*,
InNeighbours,
OutNeighbours,
MaxTime,
MaxMem

VARIABLE *state*,
messages

$$
\begin{array}{l}
DCGInit \triangleq \\
\quad \land\ messages = InitMsg \\
\quad \land\ state = [n \in 1 \,..\, Nodes \mapsto [\\
\qquad t \mapsto 0, \\
\qquad c \mapsto [m\ \in 1 \,..\, MaxMem \mapsto \\
\qquad\quad \text{IF } m = 1 \text{ THEN } Cardinality(InNeighbours[n]) \text{ ELSE } 0 \\
\qquad], \\
\qquad \text{time difference (ahead) from out nueron} \\
\qquad tDiff \mapsto [o \in OutNeighbours[n] \mapsto 0] \\
\quad]]
\end{array}
$$

$$
\begin{array}{l}
Fire(n) \triangleq \\
\quad \land\ state[n].t < MaxTime \\
\quad \land\ state[n].c[1] = Cardinality(InNeighbours[n]) \\
\quad \land \forall\, o \in OutNeighbours[n] : state[n].tDiff[o] < MaxMem - 1 \\
\quad \land\ \text{LET } msg \triangleq \\
\qquad \{[\ type \mapsto \text{“fire”}, \\
\qquad\quad sender \mapsto n, \\
\qquad\quad out \mapsto o, \\
\qquad\quad t \mapsto state[n].t + 1 \\
\qquad] : o \in OutNeighbours[n]\} \cup \\
\qquad \text{send current time to IN nodes} \\
\qquad \text{to limit messages to a particular time ahead} \\
\qquad \{[\ type \mapsto \text{“confirm”}, \\
\qquad\quad sender \mapsto n, \\
\qquad\quad out \mapsto i, \\
\qquad\quad t \mapsto state[n].t + 1 \\
\qquad] : i \in InNeighbours[n]\} \\
\qquad \text{IN}\quad messages' = SendMsg(messages,\ msg) \\
\quad \land\ state' = \\
\qquad [state \text{ EXCEPT} \\
\qquad\quad ![n].t = state[n].t + 1, \\
\qquad\quad ![n].c = \quad [m \in 1 \,..\, MaxMem \mapsto \\
\qquad\qquad\qquad \text{IF } m = MaxMem \text{ THEN } 0 \text{ ELSE } @[m+1]
\end{array}
$$

```
                                    ],
                        ![n].tDiff =
                            [o ∈ OutNeighbours[n] ↦ 1 + @[o]]
                    ]

Receive(m) ≜    ∨  ∧  m.type = "fire"
                   ∧  messages' = RemoveMsg(messages, m)
                   ∧  LET diff ≜ m.t − state[m.out].t + 1
                          IN   state' =
                                 [state EXCEPT ![m.out].c[diff] = 1 + @]

                ∨  ∧  m.type = "confirm"
                   ∧  messages' = RemoveMsg(messages, m)
                   ∧  LET diff ≜
                             state[m.out].t − m.t
                      IN   state' = [state EXCEPT
                             ![m.out].tDiff[m.sender] =
                                 IF diff < @ THEN diff ELSE @
                      ]

DCGNext ≜   ∨ ∃ n ∈ 1 .. Nodes : Fire(n)
            ∨  ∧ MsgAvailable(messages)
               ∧ ∃ m ∈ GetMsg(messages) : Receive(m)
```

```
NeighbourOK ≜  ∀ n ∈ 1 .. Nodes :
                  ∧ ∀ i ∈ InNeighbours[n] : n ∈ OutNeighbours[i]
                  ∧ ∀ o ∈ OutNeighbours[n] : n ∈ InNeighbours[o]

TypeOK ≜  ∧  ∀ n ∈ 1 .. Nodes :
                ∧ state[n].t ≤ MaxTime
                ∧ ∀ m ∈ 1 .. MaxMem :
                     state[n].c[m] ≤ Cardinality(InNeighbours[n])

TimeDiffOK ≜  ∀ n ∈ 1 .. Nodes :
                 ∧ ∀ i ∈ InNeighbours[n] :
                     state[i].t − state[n].t < MaxMem
                 ∧ ∀ o ∈ OutNeighbours[n] :
                     ∧ state[n].tDiff[o] < MaxMem
                     ∧ state[n].t − state[o].t < MaxMem
```
