MODULE *Door*

```
  --algorithm door
variables
  opened = FALSE,
  locked  = FALSE,
  key ∈ BOOLEAN ;

process open_door = "Opened Door"
begin
  OpenedDoor:  Things you can do when the door is already opened.
    await opened ;
    either  lock/unlock
      locked := ¬locked
    or  close the door
      await ¬locked ;
      opened := FALSE ;
    end either ;
    goto OpenedDoor ;
end process ;

process closed_door = "Closed Door"
begin
  ClosedDoor:  Things you can do when the door is already closed.
    await ¬opened ;
    either  lock/unlock
      await key ;
      locked := ¬locked ;
    or  open the door
      await ¬locked ;
      opened := TRUE ;
    end either ;
    goto ClosedDoor ;
end process ;

end algorithm  ;
```

```
 BEGIN TRANSLATION
VARIABLES opened, locked, key, pc

vars ≜ ⟨opened, locked, key, pc⟩

ProcSet ≜ {"Opened Door"} ∪ {"Closed Door"}

Init ≜  Global variables
         ∧ opened = FALSE
         ∧ locked  = FALSE
         ∧ key ∈ BOOLEAN
```

$$\begin{array}{l}
\land\ pc = [self \in ProcSet \mapsto \textsc{case}\ self = \text{“Opened Door”} \to \text{“OpenedDoor”} \\
\qquad\qquad\qquad\qquad\qquad\qquad \Box\ \ self = \text{“Closed Door”} \to \text{“ClosedDoor”}]
\end{array}$$

$$\begin{array}{l}
OpenedDoor \triangleq \land\ pc[\text{“Opened Door”}] = \text{“OpenedDoor”} \\
\qquad\qquad\qquad \land\ opened \\
\qquad\qquad\qquad \land\ \lor\ \land\ locked' = \neg locked \\
\qquad\qquad\qquad\qquad\quad \land\ \textsc{unchanged}\ opened \\
\qquad\qquad\qquad\quad\ \lor\ \land\ \neg locked \\
\qquad\qquad\qquad\qquad\quad \land\ opened' = \textsc{false} \\
\qquad\qquad\qquad\qquad\quad \land\ \textsc{unchanged}\ locked \\
\qquad\qquad\qquad \land\ pc' = [pc\ \textsc{except}\ ![\text{“Opened Door”}] = \text{“OpenedDoor”}] \\
\qquad\qquad\qquad \land\ key' = key
\end{array}$$

$$open\_door \triangleq OpenedDoor$$

$$\begin{array}{l}
ClosedDoor \triangleq \land\ pc[\text{“Closed Door”}] = \text{“ClosedDoor”} \\
\qquad\qquad\qquad \land\ \neg opened \\
\qquad\qquad\qquad \land\ \lor\ \land\ key \\
\qquad\qquad\qquad\qquad\quad \land\ locked' = \neg locked \\
\qquad\qquad\qquad\qquad\quad \land\ \textsc{unchanged}\ opened \\
\qquad\qquad\qquad\quad\ \lor\ \land\ \neg locked \\
\qquad\qquad\qquad\qquad\quad \land\ opened' = \textsc{true} \\
\qquad\qquad\qquad\qquad\quad \land\ \textsc{unchanged}\ locked \\
\qquad\qquad\qquad \land\ pc' = [pc\ \textsc{except}\ ![\text{“Closed Door”}] = \text{“ClosedDoor”}] \\
\qquad\qquad\qquad \land\ key' = key
\end{array}$$

$$closed\_door \triangleq ClosedDoor$$

$$\begin{array}{l}
Next \triangleq open\_door \lor closed\_door \\
\qquad\qquad \lor\ \ \text{Disjunct to prevent deadlock on termination} \\
\qquad\qquad\quad ((\forall\, self \in ProcSet : pc[self] = \text{“Done”}) \land \textsc{unchanged}\ vars)
\end{array}$$

$$Spec \triangleq Init \land \Box[Next]_{vars}$$

$$Termination \triangleq \Diamond(\forall\, self \in ProcSet : pc[self] = \text{“Done”})$$

END TRANSLATION

* Modification History
* Last modified Sun May 05 11:27:05 *PDT* 2019 by *jasondebolt*
* Created Sun May 05 10:55:20 *PDT* 2019 by *jasondebolt*