Aufgabe 2b. Parallele Spezifikationen

Trafficlightprotokoll_02

```
% Definition der Richtungen
sort
        CardinalDirection = struct north | east | south | west;

% Definition der Farben
sort
        Colour = struct red | yellow | green;
map
        nextColour: Colour -> Colour;   % Der Wert des nextColour als Colour deklariert

        % Initialisierung die Werte der deklarierten Colourstypen
eqn
        nextColour(red) = green;
        nextColour(green) = yellow;
        nextColour(yellow) = red;

map
isCrossingSafe: Colour # Colour # Colour # Colour -> Bool;  % Der Wert des
isCrossingSafe als Boolean deklariert
```

```
var

        n,e,s,w : Colour;


% isCrossingSafe gibt boolean true zurück, wenn alle die gegebene Bedingungen erfüllt sind

eqn

   isCrossingSafe(n,e,s,w) = if(((n == red && w == red && s == red && e == red) || (n == red && w == green && s == red && e == green) || (n == red && w == yellow && s == red && e == yellow) || (n == red && w == green && s == red && e == yellow) || (n == red && w == yellow && s == red && e == green)  || (n == green && w == red && s == green && e == red) || (n == yellow && w == red && s == yellow && e == red) || (n == green && w == red && s == yellow && e == red)  || (n == yellow && w == red && s == green && e == red) || (n == red && w == red && s == red && e == green) || (n == red && w == red && s == red && e == yellow) || (n == red && w == red && s == green && e == red) || (n == red && w == red && s == yellow && e == red) || (n == red && w == green && s == red && e == red) || (n == red && w == yellow && s == red && e == red) || (n == green && w == red && s == red && e == red)|| (n == yellow && w == red && s == red && e == red)),true,false);


%Definition der Aktionen und Prozesse

act

        show: CardinalDirection # Colour;

        crossingUnsafe : Colour # Colour # Colour # Colour;

        seeColour: CardinalDirection # Colour;

        colourSeen: CardinalDirection # Colour;


        % Es wird immer durch eine Farbe die nächste Farbe gebracht

proc

        TrafficLight(dir : CardinalDirection, colour : Colour) = show(dir,colour).TrafficLight(colour = nextColour(colour));
```

```
% Der Monitor im Zentrum guckt in vier Richtungen und entscheidet, ob es sicher oder unsicher ist

Monitor(n: Colour, e: Colour, s:Colour, w:Colour) =isCrossingSafe(n,e,s,w) -> (

sum dir: CardinalDirection.sum colour:Colour.seeColour(dir, colour).(dir == north)-> Monitor(n = colour) <>(dir == east) -> Monitor(e = colour) <>(dir == south)-> Monitor(s = colour) <>(dir == west) -> Monitor(w = colour))<>(

crossingUnsafe(n,e,s,w).delta  % Passiert ein Deadlock und das Prozess wird blockiert

);


% Allow nur die Aktionen colourseen und crossingUnsafe

% Die Aktionen show und seeColour werden zusammen verbunden , und in multi-Prozesse TrafficLight mit der Aktion colourseen kommunizieren.

% TrafficLight Prozesse laufen parallel, und werden auf Red initialisiert

% Monitor wird auch auf red initialisiert


CrossRoad =allow({colourSeen, crossingUnsafe},comm({show|seeColour -> colourSeen},(TrafficLight(north, red) || TrafficLight(east, red) || TrafficLight(south, red) || TrafficLight(west, red) || Monitor(red,red,red,red)));


% Es wird mit dem Prozess CrossRoad begonnen

init

        CrossRoad;
```

Beschreibung der Probleme :

1. Der Monitor entscheidet, ob die Überfahrt sicher oder unsicher ist.
2. Die Ampelfarbe muss in alle Richtungen durch den Monitor überprüft werden.
3. Soll das Prozess blockiert werden, wenn die Funktion Überfahrt aufgerufen wird.
4. Allow nur die Aktionen colourseen und crossingUnsafe.
5. Die Aktionen show und seeColour werden zusammen verbunden,und in multi-Prozesse TrafficLight mit der Aktion colourseen kommunizieren.

Lösungen der Probleme :

1. A- Map isCrossingSafe: Colour # Colour # Colour # Colour -> Bool; Mit Map erhält die Funktion isCrossingSafe 4 Colortypen, und wird den Wert der Funktion isCrossingSafe als Boolean definiert.
   B- Act crossingUnsafe : Colour # Colour # Colour # Colour; Es wird hier eine Aktion crossingUnsafe definiert,die 4 Colortypen als Parameter akzeptieren kann.
2. A- Eqn isCrossingSafe(n,e,s,w) = Alle die zulässige Bedingungen; Es wird mit eqn isCrossingSafe(n,e,s,w) überprüft,ob alle die Ampelfarben in alle Richtungen die gegebene Bedingungen erfüllen oder nicht. B- Monitor(n: Colour, e: Colour, s:Colour, w:Colour) = isCrossingSafe(n,e,s,w) ->(...).<>(crossingUnsafe(n,e,s,w).delta. Prozess Monitor erhält vier Colourtypen für alle die Richtungen in seinem Constructor

und dann muss einer von die zwei Situationen ausgewählt werden.

3. crossingUnsafe(n,e,s,w).delta Wenn die Überfahrt die gegebene Bedingungen nicht erfüllt, und unsicher ist, dann wird das gesamte Prozess durch delta blockiert.
4. allow({colourSeen, crossingUnsafe} nur die Aktionen colourseen und crossingUnsafe sind zulässig.
5. comm({show|seeColour -> colourSeen}.