Aufgabe 2b. Parallele Spezifikationen

Trafficlightprotokoll_04

```
 % 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
oppSide: CardinalDirection -> CardinalDirection; % Der Wert des oppSide
als CardinalDirection deklariert
nextSide: CardinalDirection -> CardinalDirection; % Der Wert des nextSide
als CardinalDirection deklariert
```

```
% Initialisierung die Werte der deklarierten oppSide und nextSide
eqn
oppSide(north) = south;
oppSide(east) = west;
oppSide(south) = north;
oppSide(west) = east;
nextSide(north) = east;
nextSide(east) = south;
nextSide(south) = west;
nextSide(west) = north;


        % Definition der Aktionen und Prozesse

act

        show: CardinalDirection # Colour;

        sync: Set(CardinalDirection) # Colour;

        synced: Set(CardinalDirection) # Colour;

        signal: CardinalDirection;

        wait: CardinalDirection;

        notify: CardinalDirection;


% Die Ampeln werden gegenseitig synchronisiert,und die unsichere
Kombinationen von Ampelwerten werden vermieden

% Nord-Süd Richtungen werden mit der ersten Grünphase anfangen

proc

TrafficLight(_dir : CardinalDirection) =(_dir == east) ->
(wait(_dir).TrafficLight(_colour = red))<> (_dir == west) ->
(wait(_dir).TrafficLight(_colour = red))<> TrafficLight(_colour = red);

TrafficLight(_dir : CardinalDirection, _colour : Colour) =
```

show(_dir,_colour).sync({_dir,oppSide(_dir)},_colour)

.(_colour == red) -> (signal(nextSide(_dir)).wait(_dir).TrafficLight(_colour = nextColour(_colour)))<> (TrafficLight(_colour = nextColour(_colour)));

% TrafficLight Prozesse laufen parallel, und werden auf Colour Red initialisiert

% synced und notify Aktionen werden mit hide verschwunden

% Die Aktionen synced,notify und show sind nur zulässig

% Die zwei Aktionen sync werden zusammen verbunden und in multi-Prozesse TrafficLight und mit der Aktion synced kommunizieren

% Die Aktionen signal und wait werden zusammen verbunden und in multi-Prozesse TrafficLight mit der Aktion notify kommunizieren

init

hide({synced, notify},

allow({show, synced, notify},

comm({sync|sync -> synced, signal|wait -> notify},

TrafficLight(north)||TrafficLight(south)||TrafficLight(east)||TrafficLight(west)))
);

Beschreibung der Probleme :

1. Nord-Süd Richtungen werden mit der ersten Grünphase anfangen.
2. Die Ampeln werden gegenseitig synchronisiert,und die unsichere Kombinationen von Ampelwerten werden vermieden.
3. Synced und notify Aktionen werden verschwunden.
4. Die Aktionen synced,notify und show sind nur zulässig.
5. Die Aktionen sync,sync und signal,wait werden zusammen verbunden.

Lösungen der Probleme :

1. In der ersten Initialisierung vom Prozess Trafficlight wird überprüft, ob die Richtungen Ost und West entsprechen, dann wird die Farbe auf Red initialisiert und Trafficlight als Rekrsiv aufgerufen.Gleichzeigit schickt wait Aktion die Richtung nach signal(nextSide(_dir)) in der zweiten Initialisierung vom Prozess Trafficlight. TrafficLight(_dir : CardinalDirection) = (_dir == east) -> (wait(_dir).TrafficLight(_colour = red))<> (_dir == west) -> (wait(_dir).TrafficLight(_colour = red))<> TrafficLight(_colour = red); - dann werden die North und South Richtungen durch nextSide aufgerufen, und erhält die nächste Grün Farbe in Trafficlightlight.
   signal(nextSide(_dir)).wait(_dir).TrafficLight(_colour = nextColour(_colour)))

2. Wenn die zwei gegenseitige Richtungen auf red initialisiert worden sind, dann schickt signal(nextSide(_dir)) die andere umgekehrte gegenseitige Richtungen nach wait(_dir), und erhält die nächste Farbe in Trafficlightlight, bis sie auf red wieder kommen, und so läuft das Prozess in Endlossschleife. TrafficLight(_dir : CardinalDirection, _colour : Colour) =show(_dir,_colour).sync({_dir,oppSide(_dir)},_colour)

(_colour == red) -> (signal(nextSide(_dir)).wait(_dir).TrafficLight(_colour = nextColour(_colour)))<> (TrafficLight(_colour = nextColour(_colour)));

3. hide({synced, notify}.

4. allow({show, synced, notify}.

5. comm({sync|sync -> synced, signal|wait -> notify}