Systeme hoher Sicherheit und Qualität WiSe 19/20

Christoph Lüth
Helmar Hutschenreuter

6. Übungsblatt

Ausgabe: 28.11.19

Dieses Übungsblatt ist ein PDF-Formular. Sie können es in einem PDF-Viewer Ihrer Wahl ausfüllen, abspeichern, und an die Veranstalter mailen, oder ausdrucken, mit Gänsefeder ausfüllen und per Brieftaube an die Veranstalter schicken.

Gruppe:

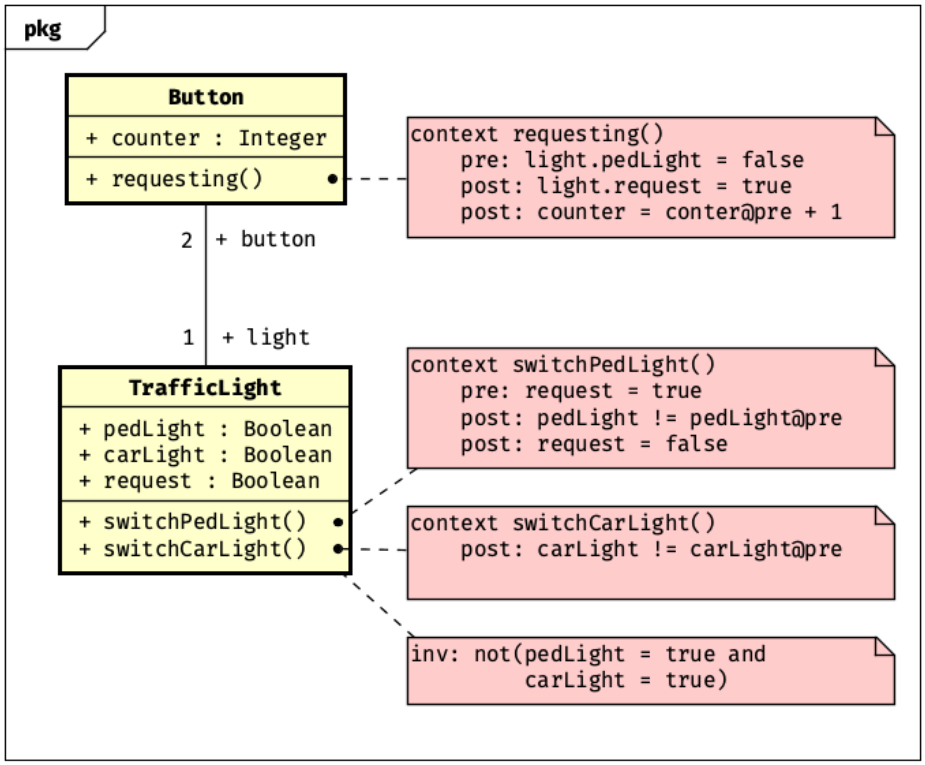
Name: Matrikelnummer:

Name: Matrikelnummer:

Name: Matrikelnummer:

In der Vorlesung wurde folgendes Beispiel zum Thema „OCL“ vorgestellt. Um Fußgängern das sichere Überqueren einer Straße zu ermöglichen, wurde eine Ampelanlage mit Bedarfsschaltung eingerichtet. Fußgänger, die die Straße überqueren möchten, drücken auf den Knopf (Button) an der Ampel und nach einiger Zeit schaltet die Ampelanlage um.

Der interne Aufbau und das Verhalten der Ampelanlage wird über das UML-Klassendiagramm und die im Klassendiagramm annotierten OCL-Constraints verdeutlicht:



6.1 *Vorsicht Fehler!*

Leider hat sich in den Aufbau der Ampelschaltung ein Fehler eingeschlichen. Geht man davon aus, dass im initialen Zustand der Ampelanlage die Variablen `pedLight = False` `carLight = True`, `request = False` und `counter = 0` gesetzt sind, tritt ein Deadlock-Zustand auf, wenn von diesem Zustand aus die Methoden `requesting`, `switchCarLight`, `switchPedLight` in der angegebenen Reihenfolge nacheinander aufgerufen werden.

Welche der folgenden alternativen OCL-Constraints beheben diesen Fehler? Prüfen Sie die angegebenen Alternativen getrennt voneinander. Ersetzen Sie das jeweilige OCL-Constraint mit identischem Context in der Systembeschreibung und prüfen Sie anschließend, ob der Fehler behoben wird, ohne dass die sichere und sinnvolle Funktion des System verloren geht.

☐
```
context Button::requesting()
    post: light.request = true
    post: light.counter = counter@pre + 1
```

☐
```
context TrafficLight
    inv: (pedlight = true implies carLight = false) and
         (carLight = false implies pedlight = true)
```

☐
```
context TrafficLight::switchPedLight()
    pre: request = true or carLight = false
    post: pedLight != pedLight@pre
    post: request = false
```

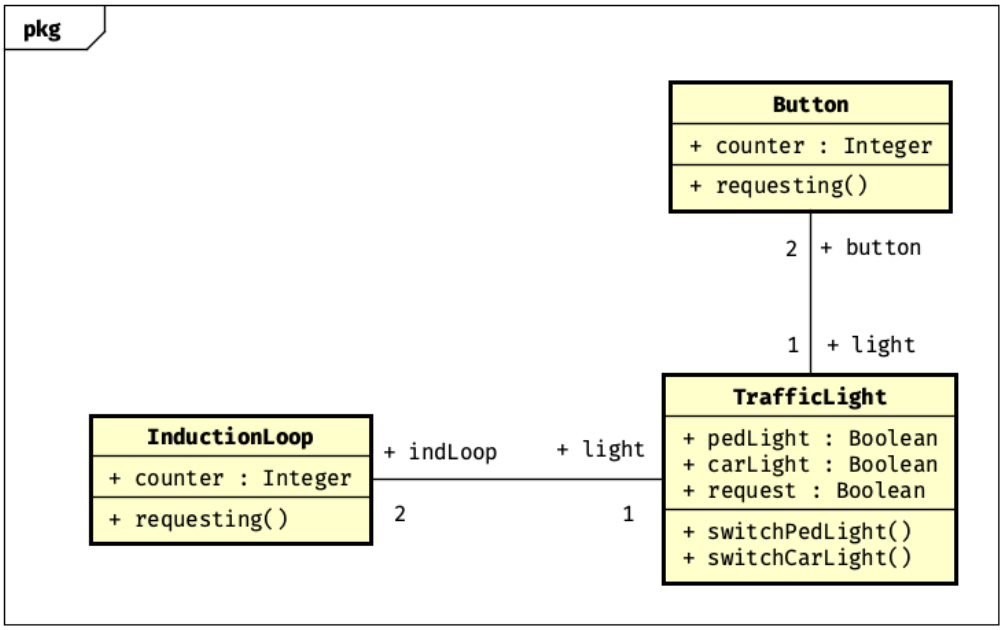
☐
```
context TrafficLight::switchCarLight()
    pre: pedLight = false and request = false
    post: carLight != carLight@pre
```

☐
```
context Button::requesting()
    pre: light.pedLight = false or carLight = false
    post: light.request = true
    post: light.counter = counter@pre + 1
```

☐
```
context TrafficLight::switchPedLight()
    pre: request = true or pedLight = true
    post: pedLight != pedLight@pre
    post: request = false
```

6.2 *Induktionsschleife*

Moderne Ampelschaltungen reagieren auf wartende Fahrzeuge, um den Verkehrsfluss zu optimieren. Dafür wird unter dem Asphalt der Fahrbahn eine Induktionsschleife verbaut, die der Ampelschaltung meldet, wenn ein Fahrzeug vor der Ampel auf der Fahrbahn wartet. Das UML-Diagramm wird um die Klasse `InductionLoop` erweitert:



Geben Sie OCL-Constraints für das System an, so dass diese sinnvoll auf wartende Fahrzeuge reagiert. Nehmen Sie dabei an, dass die im Kontext `TrafficLight` definierte Invariante (siehe Aufgabe 6.1) weiterhin definiert ist.

- `context Button::requesting()`

- `context InductionLoop::requesting()`

- `context TrafficLight::switchPedLight()`

- `context TrafficLight::switchCarLight()`