Systeme hoher Sicherheit und Qualität WiSe 15/16

Christoph Lüth
Martin Ring

3. Übungsblatt

Ausgabe: 17.11.15
Abgabe: 01.12.15

3.1 *Hindernisse* — *7 Punkte*

In dieser Übung formaliseren wir einen Aspekt der Hindernisvermeidung. Gegeben seien Klassen `Point` für Punkte und `Polygon` für Polygone, letztere mit Operationen `move(p: Point): Polygon` und `dist(p: Point): Real`, welche ein Polygon verschieben und die Distanz eines Punktes zu einem Polygon angeben. (Diese Operationen wollen wir hier nicht weiter spezifizieren).

Ein Hindernis besteht aus einem Polygon (seiner Fläche), und zusätzlich einem Attribut `dangerous`; das Auto wiederum besteht auch aus einem Polygon (seiner Fläche), einer Menge von Hindernissen, und hat eine Methode `move(p: Point)`, welche einen Vektor qua Punkt `p` als Argument bekomment.

Formalisieren Sie folgende Sicherheitsanforderungen und Hilfsdefinitionen:

- Ein Punkt ist in einem Polygon enthalten, wenn die Distanz 0 oder kleiner ist;
- Zwei Polygone schneiden sich, wenn es einen Punkt gibt, der in beiden Polygonen enthalten ist;
- Die Sicherheitsinvariante: die Fläche des Autos und keines der Hindernisse sollen sich schneiden;
- Wenn das Auto sich mit einem Vektor `p` bewegt, soll die Fläche des Autos nach der Bewegung die um den Vektor `p` verschobene Fläche des Autos vor der Bewegung sein;
- Wenn sich das Auto bewegt, sollen sich nur die gefährlichen Hindernisse bewegen können (d.h. ihre Fläche ändert sich);
- Wenn das Auto sich bewegt, muss die Distanz des Autos vor der Bewegung zu allen gefährlichen Hindernissen größer als eine globale Konstante `SAFE_DIST` sein;
- Es soll höchstens drei gefährliche Hindernisse geben.

3.2 *Formal A^2* — *8 Punkte*

Die Qualitätskontrolle bemängelt, dass die auf dem letzten Übungsblatt gefundenen Sicherheitsanforderungen noch nicht präzise genug formuliert wurden— so würde das der TÜV nie abnehmen.

Formalisieren Sie deshalb fünf dieser Sicherheitsanforderungen geeignet in OCL. Gegenfalls müssen Sie die Sicherheitsanforderung umformulieren, so dass sie formalisiert werden können.

3.3 *Prädikatenlogik* *5 Punkte*

Monika ist nach den Erfahrungen mit den Eclipse Plugins ziemlich frustriert. So richtig weiß sie aber auch gar was das für Plugins waren. Sie schaut sich die Dokumentation an und bekommt folgende Informationen:

- Spring macht nicht die Benutzeroberfläche bunter.
- Das Plugin von Apache heißt weder MoonriseUI noch ist es 2,3 MB groß.
- Eines der Plugins kann Versionskontrolle.
- Weder das Plugin von der Eclipse Foundation noch Spring sind 2,5MB groß.
- Das Plugin von Mozilla ist 2,1MB groß.
- MoonriseUI ist nicht 2,2MB groß.
- Das Plugin das Dependency Injection kann ist weder 2,2MB groß noch ist es JBoss, noch ist es von Mozilla.
- Das Plugin der Eclipse Foundation ist nicht 2,3MB groß.
- Weder das 2,2MB große Plugin noch JBoss können remote deployment.
- Es gibt ein Plugin namens Subclipse.
- JBoss ist von RedHat.
- Jedes Plugin hat jeweils genau einen Herausgeber, eine Größe und einen Namen.

Formalisieren sie die Aussagen mit Hilfe von Prädikatenlogik. Angenommen es gibt nur die vier erwähnten Plugins, wie heißen sie, was können sie und wieviel Speicherplatz benötigen sie?