technische universität dortmund

Fakultät für Informatik
Lehrstuhl 14 für Software Engineering
Prof. Dr. Falk Howar
Simon Dierl, M. Sc.
Softwarekonstruktion WiSe 18 / 19

Übungsblatt 4

Ausgabe: 26.11.2018 **Abgabe:** 06.12.2018, 12:00 Uhr

Organisatorisches

Es gelten die Hinweise von Übungsblatt 3.

Hausübung

Aufgabe 4.1 (Auswertung von Ausdrücken) 4 P.

Sei $\texttt{State} = \{\texttt{Li}, \texttt{Ri}, \texttt{Lu}, \texttt{Pl}\}$ und die Variablenbelegung

$$v = b \mapsto \texttt{true}, c \mapsto \texttt{false}, i \mapsto 23, j \mapsto 42, s \mapsto \texttt{Pl}.$$

Werten Sie jetzt folgende Ausdrücke aus. Benutzen Sie dazu die Baumdarstellung aus der Vorlesung.

(a) $(b \vee (s = \texttt{Lu})) \wedge ((i < j) \wedge (j < (i + i)))$ 2 P.

(b) $(c \vee (b \wedge (j = 1))) \wedge ((i > (j - 1)) \vee (s = \texttt{Pl}))$ 2 P.

Aufgabe 4.2 (Erweiterte endliche Automaten) 8 P.

Um Unfälle im Bahnverkehr zu verhindern, verfügen Züge bzw. Bahnstrecken über diverse automatische Sicherheitssysteme. Zwei dieser Systeme sollen für diese Aufgabe näher betrachtet werden.

Sicherheitsfahrschaltungen (SiFa) werden im Triebfahrzeug angebracht und stellen sicher, dass, sollte der Triebfahrzeugführer – z. B. durch Schlaf oder medizinische Probleme – nicht mehr in der Lage sein, den Zug zu kontrollieren, dieses automatisch anhält. Dazu muss alle t Sekunden ein Pedal getreten oder ein Knopf gedrückt werden[1]. Geschieht dies nicht, ertönt ein Alarm; nach weiteren u Sekunden wird zusätzlich eine Zwangsbremsung ausgelöst. Die Zwangsbremsung wird beendet, sobald die SiFa betätigt wird[2].

[1] In der Praxis gibt es auch bessere Ansätze, die ein Kollabieren auf dem Pedal erkennen. Diese ignorieren wir aber hier.

[2] Auch hier ist die Praxis noch ein wenig komplexer.

Fahrsperren sind mechanische Elemente an einer Fahrstrecke, die beim Überfahren einen Sensor am Zug berühren. Sollte dies geschehen, wird unmittelbar eine Zwangsbremsung ausgelöst. Diese Elemente diesen zur Absicherung eingleisiger Strecken, von Baustellen usw. Ein Abschalten der Zwangsbremsung darf hier natürlich nicht möglich sein. Der Alarm soll ebenfalls inaktiv sein.

Des Weiteren nehmen wir an, dass der Triebfahrzeugführer den Zug beschleunigen oder bremsen kann; die *gewünschte* Beschleunigung normieren wir auf das Intervall $[-1, 1] \subset \mathbb{R}$. Gleichzeitig nehmen wir an, dass die *aktuelle* Motorleistung sowie die *aktuelle* Bremswirkung jeweils auf das Intervall $[0, 100] \subset \mathbb{Z}$ normiert sind. Eine Zwangsbremsung entspricht maximaler Bremswirkung ohne Motorleistung.

Entwerfen Sie einen *deterministischen* erweiterten endlichen Automaten, der ein um diese Sicherheitssysteme erweitertes Steuerungssystem modelliert. Nehmen Sie dazu an, dass

(i) die gewünschte Beschleunigung linear (und gerundet) auf die Steuerung von Motor und Bremse abgebildet wird,
(ii) die Sicherheitsfahrschaltung nach 30 Sekunden ohne Betätigung einen Alarm und nach 35 Sekunden eine Zwangsbremsung auslöst,
(iii) mindestens folgende Eingabe- und Zustandsvariablen existieren:

Variable	Semantik	Typ
t	Zeit in Sekunden (d. h. $\in \mathbb{Z}_{\geq 0}$) seit Beginn der Fahrt	Eingabe
a	*Gewünschte* Beschleunigung $\in [-1, 1] \subset \mathbb{R}$	Eingabe
s	`true`, wenn *in diesem Moment* die Sicherheitsfahrschaltung betätigt wird, sonst `false`	Eingabe
f	`true`, wenn *in diesem Moment* eine Fahrsperre überfahren wird, sonst `false`	Eingabe
a^+	*Aktuelle* Motorleistung $\in [0, 100] \subset \mathbb{Z}$	Zustand
a^-	*Aktuelle* Bremswirkung $\in [0, 100] \subset \mathbb{Z}$	Zustand
ℓ	`true`, wenn der Alarm der SiFa aktiv ist, sonst `false`	Zustand

(a) Zeichnen Sie diesen Automaten. Es steht Ihnen frei, Ihre Antwort zur besseren Verständlichkeit zu kommentieren. Sie können gängige mathematische Operationen wie min, max, $\lfloor \cdot \rfloor$ und $\lceil \cdot \rceil$ nutzen. 6 P.

(b) Geben Sie eine Folge von Konfigurationen Ihres Automaten an, die folgende Abfolge von Szenarien widerspiegelt: 2 P.

(i) Start des Zuges.

(ii) 23 Sekunden nach Start, volle Beschleunigung voraus.
(iii) 29 Sekunden nach Start, volle Beschleunigung voraus, SiFa kurz betätigt.
(iv) 40 Sekunden nach Start, halbe Beschleunigung voraus.
(v) 61 Sekunden nach Start, halbe Beschleunigung rückwärts.
(vi) 75 Sekunden nach Start, volle Beschleunigung voraus, SiFa kurz betätigt.
(vii) 110 Sekunden nach Start, volle Beschleunigung rückwärts.
(viii) 112 Sekunden nach Start, halbe Beschleunigung rückwärts, kurzer Kontakt mit Fahrsperre.

Sie müssen nur die Übergänge zu den genannten Zeitpunkten sowie bei einem möglichen Auslösen der SiFa-Funktionen zwischen den genannten Zeitpunkten darstellen.

Aufgabe 4.3 (Domänenspezifische Sprachen) **8 P.**

Moderne Firewalls werden üblicherweise über einen Satz *Regeln* konfiguriert, die entscheiden, wie mit einem ein- oder ausgehenden Paket verfahren wird. Eine Regel kann folgende Eigenschaften eines Paketes untersuchen[3]:

(i) Quelle (eine IP-Adresse oder Netzmaske)
(ii) Ziel (eine IP-Adresse oder Netzmaske)
(iii) Protokoll (TCP oder UDP)
(iv) Port (eine oder mehrere TCP- oder UDP-Portnummern)
(v) Netzwerkschnittstelle (die Schnittstelle, über die das Paket läuft, unter Linux z. B. `eth0`, unter Windows z. B. `Ethernet`).

Regeln müssen nicht jede dieser Eigenschaften untersuchen. Genau dann, wenn alle definierten Eigenschaften auf ein Paket zutreffen, *matcht* die Regel. Am Ende muss das Paket *akzeptiert*, *verworfen* oder *abgelehnt* werden; dies nennen wir *Entscheidung*.

Die Regeln müssen natürlich in der Praxis in einer bestimmten Reihenfolge ausgewertet werden. Dazu gibt es eine Startregel. Jeder Regel werden zwei Nachfolger (einer für Matches, einer für Nicht-Matches) zugewiesen. Diese können entweder eine Regel oder eine Entscheidung sein.

(a) Benutzen Sie einen geeigneten UML-Diagrammtyp, um ein Metamodell zu zeichnen, welches derartige Regelsätze ausdrücken kann. Sie können das Vorhandensein geeigneter Datenklassen und Aufzählungen für IP-Adressen etc. voraussetzen. Das Spezifizieren von Operationen ist *nicht* nötig. **2 P.**

[3] Wie üblich ist hier die echte Welt komplexer.

(b) Entwerfen Sie dann eine *eigene* grafische domänenspezifische Sprache für Firewall-Regelsätze, die die abstrakte Synatax des UML-Diagramms umsetzt. Zeichnen Sie die Elemente Ihrer Sprache und erläutern Sie deren Zusammenhang zur abstraken Syntax, soweit nötig. 2 P.

(c) Illustrieren Sie die Anwendung Ihrer Sprache mit einem Beispiel mit mindestens zwei Regeln und vier Eigenschaften. 1 P.

(d) Entwerfen Sie außerdem eine textuelle domänenspezifische Sprache für das selbe Szenario. Skizzieren Sie die Syntax Ihrer Sprache und erläutern Sie deren Zusammenhang zur abstraken Syntax, soweit nötig. 2 P.

(e) Illustrieren Sie die Anwendung Ihrer zweiten Sprache mit dem selben Beispiel wie oben. 1 P.

Kommentieren Sie Ihre Antworten, wenn die Umsetzung nicht trivial aus den Namen hervorgeht.