UNIKASSEL VERSITÄT | ELEKTROTECHNIK INFORMATIK

Jens Kosiol

Ausgabe: 5. Juni 2023
Abgabe: 19. Juni 2023

Graph- and Model-Driven Engineering
Übungsblatt 4

1 Berechnung von Anwendungsbedingungen (12 Punkte)

In dieser Aufgabe verwenden wir wieder die Ontologie für den Verkauf von Medien von Übungsblatt 2. Abbildung 1 zeigt eine Graphformel für diese Anwendungsdomäne.

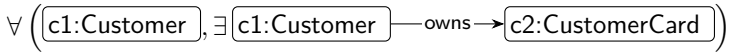

Abbildung 1: Graphformel zur Existenz von Bonuskarten

a) Erklären sie natürlichsprachlich, was die Graphformel aus Abbildung 1 bedeutet. [2 Punkte]

b) Berechnen Sie für diese Graphformel und die Regel *deleteCard_R* von Übungsblatt 2 die *garantierende Anwendungsbedingung*. Ihre Lösung soll die Rechenschritte zeigen. [6 Punkte]

c) Angenommen Sie wissen, dass Sie Ihre Regel(n) nur auf valide Graphen anwenden werden, also solche, die die Graphformel erfüllen. Sie wollen sicherstellen, dass berechnete Ergebnisse (Graphen nach Regelanwendung) weiterhin valide sind.

   i) Wie können Sie die oben berechnete garantierende Anwendungsbedingung in diesem Fall vereinfachen? Begründen Sie, warum Ihre Anwendungsbedingung ausreicht und welche Vorteile sie hat. [2 Punkte]

   ii) Würden Sie in diesem Szenario die Regeln *useCustomerCard_R* und *order_and_create_CustomerCard* mit Anwendungsbedingungen versehen? Warum (nicht)? [2 Punkte]

2 Definition von domänenspezifischen Modellierungssprachen (12 Punkte)

Entwerfen Sie in Analogie zum Vorgehen in der Vorlesung eine domänenspezifische Modellierungssprache für *Entity-Relationship-Diagramme*. Gehen Sie in den folgenden Schritten vor:

Jens Kosiol

Ausgabe: 5. Juni 2023
Abgabe: 19. Juni 2023

Graph- and Model-Driven Engineering

Übungsblatt 4

a) Geben Sie ein Metamodell (mit Kanten-Multiplizitäten) an, das die abstrakte Syntax definiert. [4 Punkte]

b) Formalisieren Sie die Constraints, die für die Sprache gelten sollen, aber nicht durch das Metamodell ausdrückbar sind, als geschachtelte Graphformeln. [2 Punkte]

c) Entwickeln Sie eine Grammatik (Startgraph und Regeln), die es erlaubt, solche Entity-Relationship-Diagramme zu generieren, die keines der geforderten Constraints verletzen. Außerdem sollen Instanzen von jedem Element des Metamodells auch erzeugbar sein. [6 Punkte]

Die Struktur einfacher *Entity-Relationship-Diagramme* ist gegeben durch:

- Es gibt Entitäten, Beziehungen und Attribute.
- Entitäten haben Attribute.
- Beziehungen verknüpfen Entitäten.
- Eine Beziehung kann vom Typ Verknüpfung, Aggregation oder Generalisierung sein.
- Während Verknüpfungen und Aggregationen ungerichtet sind, sind Generalisierungen gerichtet.

Außerdem sollen die folgenden Bedingungen für *Entity-Relationship-Diagramme* gelten:

- Es gibt mindestens eine Entität.
- Jede Entität besitzt mindestens ein Attribut.
- Eine Beziehung verbindet mindestens zwei Entitäten, aber darf nur dann mehr als zwei Entitäten miteinander verbinden, wenn sie vom Typ Aggregation ist.
- Eine Entität wird höchstens von einer anderen Entität (direkt) generalisiert.
- Eine Entität darf sich nicht direkt über eine Generalisierung selbst generalisieren.

Jens Kosiol

Ausgabe: 5. Juni 2023
Abgabe: 19. Juni 2023

Graph- and Model-Driven Engineering
Übungsblatt 4

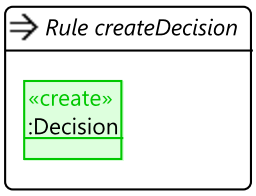

Abbildung 2: Regel zum freien Hinzufügen von Decision-Knoten.

3 Editieren und Modellevolution (10 Punkte)

In der Vorlesung wurde ein Metamodell für einfache Aktivitätendiagramme präsentiert. Nutzen Sie dieses als Grundlage folgender Aufgaben.

a) Gegeben sei die in Abb. 2 dargestellte Regel zum freien Einfügen von Knoten des Typs *Decision*. Entwerfen Sie einen Quickfix, der zwei bereits vorhandene Knoten vom Typ *SimpleActivity* verwendet. Dieser soll so gestaltet sein, dass die Constraints für Aktivitätendiagramme aus der Vorlesung nach Anwendung des Quickfixes wieder erfüllt sind, sofern das Eingabemodell M vor dem Einfügen des *Decision*-Knoten valide war.

   Geben Sie je ein valides Modell M an, auf das Ihr Quickfix nach der Anwendung von *createDecision* anwendbar ist bzw. nicht anwendbar ist. [6 Punkte]

b) In der Vorlesung wurde eine Evolution der Sprache für Aktivitätendiagramme vorgestellt. Erstellen Sie für diese Sprache eine Regel *insertFork*, welche einen Knoten vom Typ *Fork* nach einer *SimpleActivity* einfügt. Diese Editieroperation soll syntaxgesteuert sein, also die Korrektheit eines Modells erhalten. Zusätzlich zu den Constraints, die bereits vor der Sprachevolution galten (und die nur an die Namensänderungen adaptiert werden müssen), sollen folgende drei Constraints berücksichtigt werden:

   - Jeder *Fork* hat genau eine ein- und genau zwei ausgehende *Transitionen*.
   - Ausgehende *Transitionen* eines *Fork* haben keine *Guards*.
   - Die ausgehenden Transitionen eines *Fork* führen nicht direkt zum selben *Join*.

   Geben Sie alle benötigten Parameter explizit mit an. [4 Punkte]