UNIKASSEL VERSITÄT | ELEKTROTECHNIK INFORMATIK

*Jens Kosiol*

*Ausgabe: 17. Juni 2024*
*Abgabe: 1. Juli 2024*

*Graph- and Model-Driven Engineering*

***Übungsblatt 4***

**Es gelten die gleichen Regelungen zur Abgabe wie für Blatt 1.**

# 1 Multiplizitäten als Graphformeln (6 Punkte)

Definieren Sie Multiplizitäten (Knoten- und Kantenmultiplizitäten wie in der Vorlesung als eigenständiges Konzept eingeführt) als (geschachtelte) Graphformeln. Das heißt, gegeben einen endlichen Typgraphen mit Multiplizitäten $TGM = (TGI, m_n, m_s, m_t)$, geben Sie drei Graphformeln $c_n, c_s, c_m$ an, sodass ein Graph $G$ den Typgraphen mit Multiplizitäten $TGM$ erfüllt, genau dann wenn er die drei Graphformeln $c_n, c_s, c_m$ erfüllt.

**Hinweis:** Die Graphformeln sind notwendig schematisch, da konkrete Formeln abhängig von der Anzahl der Knoten, Kanten und den konkreten Werten von $m_n, m_s, m_t$ wären.

UNIKASSEL VERSITÄT | ELEKTROTECHNIK INFORMATIK

*Jens Kosiol*

*Ausgabe: 17. Juni 2024*
*Abgabe: 1. Juli 2024*

*Graph- and Model-Driven Engineering*
***Übungsblatt 4***

# 2 Entwurf und Auswertung von Graphformeln (8 Punkte)

Gegeben sei der Typgraph für Klassendiagramme von Aufgabenblatt 2 (Sie dürfen Ihre eigene Lösung verwenden oder die in Abbildung 1 gezeigte).

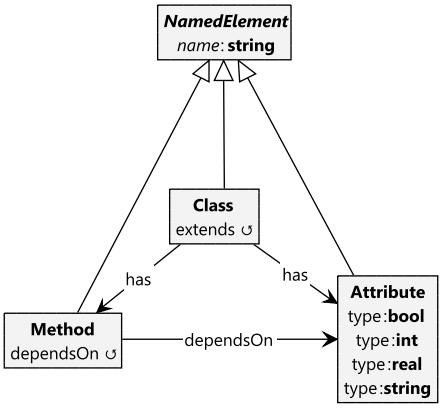


Abbildung 1: Typgraph für vereinfachte Klassendiagramme

**Aufgaben:**

a) Formalisieren Sie die folgenden Aussagen als (über dem gewählten Typgraphen getypte) geschachtelte Graphformeln. Sie dürfen die Formeln in Groove erstellen oder wie in der Vorlesung eingeführt zeichnen; die besprochenen Vereinfachungen in der Darstellung dürfen Sie dabei benutzen.

   i) Wenn eine Klasse ein Attribut besitzt, besitzt sie auch eine Methode, die dieses Attribut verwendet. (2 Punkte)

   ii) Es gibt eine Klasse `c1` mit Methode `m1` sodass jede weitere Klasse `c2` ein Attribut besitzt, auf das `m1` zugreift. (2 Punkte)

b) Geben Sie für die Formel ii) von oben jeweils einen (über dem gewählten Typgraphen getypten) Graphen an, der die Formel erfüllt, und einen, der dies nicht tut. Begründen Sie jeweils, warum dies der Fall ist (unter Verweis auf die Semantik Ihrer erstellten Graphformel, nicht der natürlichsprachlichen Bedeutung). (4 Punkte)

UNIKASSEL VERSITÄT | ELEKTROTECHNIK INFORMATIK

*Jens Kosiol*

*Ausgabe: 17. Juni 2024*
*Abgabe: 1. Juli 2024*

*Graph- and Model-Driven Engineering*
***Übungsblatt 4***

# 3 Berechnung von Anwendungsbedingungen (12 Punkte)

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

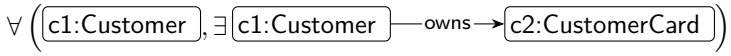


Abbildung 2: Graphformel zur Existenz von Bonuskarten

a) Erklären sie natürlichsprachlich, was die Graphformel aus Abbildung 2 bedeutet. (2 Punkte)

b) Berechnen Sie für diese Graphformel und die Regel *deleteCard_R* von Übungsblatt 3 die garantierende Anwendungsbedingung. Ihre Lösung soll die einzelnen 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)

UNIKASSEL VERSITÄT | ELEKTROTECHNIK INFORMATIK

*Jens Kosiol*

*Ausgabe: 17. Juni 2024*
*Abgabe: 1. Juli 2024*

*Graph- and Model-Driven Engineering*
***Übungsblatt 4***

# 4 Definition von domänenspezifischen Modellierungssprachen (12 Punkte)

Entwerfen Sie in Analogie zum Vorgehen in der Vorlesung eine domänenspezifische Modellierungssprache für GUIs. Gehen Sie in den folgenden Schritten vor:

a) Geben Sie ein Metamodell (mit Kanten-Multiplizitäten) an, das die abstrakte Syntax definiert. (2 Punkte)

b) Formalisieren Sie (mindestens zwei) Constraints, die für die Sprache gelten sollen, aber nicht durch das Metamodell ausdrückbar sind, als geschachtelte Graphformeln (soweit möglich). (2 Punkte)

c) Entwickeln Sie eine Grammatik (Startgraph und Regeln), die es erlaubt, solche GUIs zu generieren, die keines der geforderten Constraints verletzen. Außerdem sollen Instanzen von jedem Element des Metamodells auch erzeugbar sein. Dokumentieren Sie explizit (z. B. durch Kommentarknoten in Groove), wie Sie dafür sorgen, dass Ihre Regeln die geforderten Constraints nicht verletzen. (6 Punkte)

d) Entwerfen Sie zwei Graphtransformationsregeln, die Teile der intendierten Semantik modellieren (z. B. einen Szenenwechsel, das Einblenden von Elementen o. Ä.). Diese Regeln dürfen über einem im Vergleich zu i) erweiterten Metamodell getypt sein. (2 Punkte)

Orientieren Sie sich für ihre Modellierungssprache an der Syntax, Struktur (und Semantik) von JavaFX (vereinfacht): Eine Stage zeigt eine Scene an, deren Inhalt durch einen Baum gegeben wird (der Scene Graph). Die Knoten des Baums sind die (visuellen) Komponenten aus denen die Oberfläche besteht. Es gibt (u. a.) Controls (Button, TextField, ...), um Funktionalität zur Verfügung zu stellen, und Layouts (Group, HBox, ...), um zu steuern, welches Element wo erscheint. Ihr Metamodell soll mindestens 6 Komponenten enthalten, darunter Kontroll- und Layoutkomponenten. Versehen Sie Ihre Komponenten auch mit (einer Auswahl an) sinnvollen Attributen.