Integritätsbedingung

Wechseln zu:Navigation, Suche

Dieser Artikel erfüllt die GlossarWiki-Qualitätsanforderungen **nur teilweise**:

Korrektheit: 5 (vollständig überprüft)	**Umfang**: 5 (wesentliche Fakten vorhanden)	**Quellenangaben**: 1 (fehlen großteils)	**Quellenarten**: 2 (befriedigend)	**Konformität**: 5 (ausgezeichnet)

Inhaltsverzeichnis

1 Definition

Integritätsbedingungen (**Zusicherungen**, **Assertions**) sind Bedingungen, die die Inhalte von Variablen, Objekte oder Entitäten zu bestimmten Zeitpunkten oder dauerhaft erfüllen müssen. Derartigen Bedingungen können durch boolesche Terme formal dargestellt werden.

Man unterscheidet verschiedene Arten von Integritätsbedingungen:

Invarianten: Diese Bedingungen müssen dauerhaft erfüllt sein.
Vorbedingungen: Diese Bedingungen müssen erfüllt sein, bevor eine bestimmte Methode/Funktion aufgerufen werden kann.
Nachbedingungen: Diese Bedingungen müssen erfüllt sein, nachdem eine bestimmte Methode/Funktion aufgerufen wurde.

2 Bemerkung

Integritätsbedingungen können vom System selbstständig überwacht oder mit Hilfe vom Programmierer eingebauten Tests zu bestimmten Zeitpunkten überprüft werden. Beispielsweise werden Typüberprüfungen in typisierten Sprachen schon vom Compiler vorgenommen. Für die „händische“ Überprüfung von Integritätsbedingungen gibt es häufig spezielle Befehle (wie z.B. `assert`) oder sogar ganze Testumgebungen wie z.B. Modultests (Unit-Tests).

Die Überprüfungen können zum Zeitpunkt der Übersetzung stattfinden oder erst zur Laufzeit. Im Allgemeinen sind Fehler (d.h. Verletzungen von Integritätsbedingungen), die zum Übersetzungszeitpunkt erkannt werden, wesentlich unkritischer, als Fehler, die erst zur Laufzeit auftreten.

3 Beispiele

Für ein Dreiecksobjekt mit den drei Seiten `a`, `b` und `c` gelten folgende Invarianten:

`a>0`, `b>0`, `c>0`
`a+b>c`, `b+c>a`, `a+c>b`

Für die Methode `pop` eines Kellerobjekts (Stack) `k` gilt die Vorbedingung `!k.empty()`, d.h., das oberste Kellerelement kann nur entfernt werden, wenn der Keller nicht leer ist.

Nachbedingungen haben häufig einen zeitlichen Bezug, d.h. ein Wert, der sich geändert hat, muss in einer bestimmten Beziehung zu dem ursprünglichen Wert stehen. Zu Beispiel gelten für die Methode `scale(f: double): void`, die auf ein Dreieck mit den drei Seiten `a`, `b` und `c` angewendet werden, folgende Nachbedingungen:

`newvalue(a)=f*oldvalue(a)`, `newvalue(b)=f*oldvalue(b)`,
`newvalue(c)=f*oldvalue(c)`

3.1 Datentypen und Signaturen

Datentypen und Signaturen von Methoden, Prozeduren, Funktionen sind spezielle Integritätsbedingungen.

Zum Beispiel sagt die Variablendefinition `var x: int` aus, dass die Variable `x` zu jedem Zeitpunkt nur Werte vom Typ `int` enthalten darf. Genauso besagt die Funktionsdefinition `function sqrt(x: double): double`, dass die Funktion nur mit Double-Werten aufgerufen werden darf (Vorbedingung) und dass sie nur Double-Werte als Ergebnis liefert (Nachbedingung). Siehe auch: Klasse: Klassenschema.

3.2 SQL

Gerade in SQL gibt es viele Integritätsbedingungen, die zur Laufzeit automatisch vom System überprüft werden: Man kann Not-Null-Bedingungn, Primärschlüssel, Unique-Constraints, Fremdschlüssel und Check-Constraints formulieren. Das zugehörige System garantiert, dass diese Bedingungen dauerhaft erfüllt sind (Invarianten), indem es dafür sorgt, dass eine Transaktion nicht erfolgreich abgeschlossen wird, wenn irgendwelche derartige Integritätsverletzungen vorliegen. Weitere Integritätsbedingungen (insbesonder Vor- und Nachbedinungen) können mit Hilfe von Triggern formalisiert und überprüft werden.

4 Quellen

1. **Kowarschick (MMProg)**: Wolfgang Kowarschick; Vorlesung „Multimedia-Programmierung“; Hochschule: Hochschule Augsburg; Adresse: Augsburg; Web-Link; 2018; Quellengüte: 3 (Vorlesung)

Kategorien:
Objektorientierte Programmierung
Glossar
Programmierung

Kapitel:Multimedia-Programmierung

Diese Seite wurde zuletzt am 27. April 2016 um 12:16 Uhr bearbeitet.
Inhalt verfügbar unter CC BY-NC-SA 4.0, falls Dokument nach dem 5. 3. 2011 erstellt wurde, sonst CC BY-SA DE 3.0.