TU-Berlin

Visuelle Sprachen Projekt SS05

Theoretische Informatik /
Formale Spezifikation (TFS)

Testdokumentation

Gruppe:

Manaljav Battogtokh *(manaljav@cs.tu-berlin.de)*
Tuguldur Erdene-Ochir *(tugstugi@cs.tu-berlin.de)*
Stefan Föll *(foell@cs.tu-berlin.de)*
Jan Loewe *(jloewe@cs.tu-berlin.de)*
Bjoern Tischer *(bjoernt@cs.tu-berlin.de)*

Testfall 1: test1.vocl

Dieser Testfall zeigt ein einfaches VOCL-Constraint vom Typ inv (Invariante), bei dem die Navigation ausgehend von der im Context als StartClassifierRole definierten ClassifierRole c startet. Darüberhinaus wird gezeigt, dass ein mit festem Wert belegtes Attribut (also keine Variable) im resultierenden OCL-String als Bedingung berücksichtigt wird.

OCL-String:
context c : Company inv: c.numberOfEmployees=50

Testfall 2: test2.vocl

Dieser Testfall zeigt im Unterschied zu Testfall1, dass die Navigation immer von ClassifierRoles mit Namen „self“ startet, wenn im Context keine explizite Angabe zur einer anderen Variablen gemacht wird.

OCL-String:
context Company inv: self.numberOfEmployees=50

Testfall 3: test3.vocl

Dieser Testfall zeigt im Unterschied zu den beiden vorigen Testfällen, dass die Navigation auch von einer unbenannten ClassifierRole starten kann, wenn ihr Typ mit dem des Contextes übereinstimmt. Dazu darf lediglich eine unbenannte Instanz des Context-Types vorliegen. Der resultierende OCL-String ist gleich dem aus Testfall 2, da die ClassiferRole in diesem Fall als mit einer „self“-Instanz äquivalent betrachtet wird.

OCL-String:
context Company inv: self.numberOfEmployees=50

Testfall 4: test4.vocl

Dieser Testfall zeigt die Möglichkeit zur Navigation von/zu ClassifierRoles mit Hilfe von AssociationRoles . Die Navigation endet hier bei einem festen Attributwert. Im resultierenden OCL-String wird die Navigation über AssociationEnd-Roles ausgedrückt.

OCL-String:
context Company inv: self.manager.isUnemployed=false

Testfall 5: test5.vocl

Dieser Testfall zeigt, dass Navigationen über AssociationRoles gleichgesetzt werden, wenn sie in der selben ClassifierRole (später auch SetClassifierRoles) enden.
Darüberhinaus ist an diesem Beispiel im Editor ersichtlich, dass Anchorpoints für die Verbindung von ClassifierRoles durch AssociationRoles zur Verfügung stehen, als auch dass AssociationEnd-Roles als verschiebbare Labels realisiert wurden.
Auch ist der Tesftfall ein Bsp. für einen zyklischen Navigationsausdruck

OCL-String:
context Person inv: self.child=self.wife.child

Testfall 6: test6.vocl

Dieser Testfall soll an Hand eines etwas komplexeren Navigationsausdruckes die eben beschrieben Konzepte nochmals verifzieren.

OCL-String:
context Person inv: self.employer=self.wife.employer and self.wife.child=self.child

Testfall 7: test7.vocl

Dieser Testfall zeigt einen komplexeren Navigationsausdruck über AssociationRoles, der zu 3 Gleichungen im resultierenden OCL-String führt.

OCL-String:
context Person inv: self.child.employer=self.employer and self.child.employer=self.wife.employer and self.wife.employer=self.employer

Testfall 8: test8.vocl

Dieser Testfall zeigt, dass Attribute mit festen Werten, die mitten auf einem Navigationspfad liegen, ebenfalls bei der Konvertierung berücksichtigt werden (entgegen der ursprünglichen Einführung ,jedoch wie nach neuer Festlegung in der Vorlesung) .
Für das Constraint kann optional wie in diesem Testfall immer noch ein Name angegeben (Bsp.: “TestNavigation“)
Auch wird gezeigt, dass unser Editor über 2 Arten des Layouts für AssociationRoles verfügt. Wie in diesem Beispiel kann es für die Übersichtlichkeit sehr nützlich sein sowohl Bendpoint- als auch ManhattanStyle-Routing zu verwenden.

OCL-String:
context Person invTestNavigation:
self.child.employer=self.employer and self.child=self.wife.child and self.wife.child.employer=self.child.employer and self.employer=self.wife.child.employer

Testfall 9: test9.vocl

In diesem Testfall werden mehrere Attribute für die selbe ClassifierRole modelliert. Hinsichtlich der Konvertierung ist es nicht eindeutig, mit welchem Navigationspfad das Attribut konvertiert wird, wenn wie in diesem Bsp. mehrere Pfade dafür existieren. Dies geht zurück auf die nichtdeterministische Auswahl der AGG-Regeln.
In unserem Editor werden Attribute einer ClassifierRole zur besseren Darstellung abwechselnd mit einer anderen Hintergrundfarbe belegt.

OCL-String:
context Person inv: self.wife.employer.place=Berlin and self.employer=self.wife.employer and self.employer.name=IBM and self.employer.sharePrice=22,45 and self.wife.employer.numberOfEmployees=4000

Testfall 10: test10.vocl

Es können auch mehrere Constraints in einer Datei modelliert werden, wie folgender Testfall zeigt . Sie werden bei der Konvertierung „verundet“.
Auch wird gezeigt, dass die AssociationRole-Verbindungsline mit Hilfe von Bendpoints(Hinzufügen, Verschieben, Löschen) verändert werden kann.

OCL-String:
context Person inv : self=self.child.parent
and
context Person inv : self=self.wife.husband

Testfall 11: test11.vocl

Dieser Testfall zeigt, dass Attributvariablen über das Hinzufügen einer Condition näher spezifiziert werden können.
Bei der Definition von Attributen wird zwischen Attributwerten und - variablen unterschieden. Attributvariablen stehen über eine DropDown-Liste als Operand für eine Vergleichsoperation, die eine Bedingung über die Variable ausdrückt, zur Auswahl. Bei der Konvertierung wird die Beziehung zum Attribut wieder aufgelöst.

OCL-String:
context c : Company inv : c.numberOfEmployees > 50

Testfall 12: test12.vocl

Dieser Testfall zeigt, dass zwei Attribute über ihre zugehörigen Variablen miteinander verglichen werden. Für den Vergleich werden in der Condition die Variablen für die linke und rechte Seite der Bedingung sowie ein entsprechender Operator ausgewählt.

OCL-String:
context p : Person inv ParentElderThanChildren: p.age > p.child.age

Testfall 13: test13.vocl

Dieser Testfall zeigt die Navigation zu einer SetClassifierRole, für die die Methode isEmpty() definiert ist. Die Konvertierung sieht bei der Navigation zu SetClassifierRoles mit Operationen vor, die jeweilige Operation auf dem bisherigen Navigationsausdruck aufzurufen.

OCL-String:
context Person inv CompanyIsEmpty: self.employer->isEmpty()

Testfall 14: test14.vocl

Dieser Testfall zeigt analog zum letzten Testfall die Navigation zu einer SetClassifierRole mit Operation notEmpty().
Dabei können Navigationspfade auch entlang von AssociationEndRoles von/zu SetClassifierRoles ohne Operationen folgen, wie dieses Besipiel zeigt.

OCL-String:
context Person inv AllChildrenEmployed: self.children.employer->notEmpty()

Testfall 15: test15.vocl

Dieser Testfall zeigt die Navigation zu einer SetClassifierRole mit Operation size(). Es kann eine beliebige Variable stellvertretend für die Collection-Size definiert werden. Diese Variable kann dann in der Condition dazu verwendet werden dafür eine einschränkende Bedingung zu formulieren.

OCL-String:
context Person inv : self.employer->size() < 3

Testfall 16: test16.vocl

Dieser Testfall verdeutlicht noch mal die letzte Aussage:
Es werden 2 Variablen x und y für die Operation size() auf unterschiedlichen definiert, die in der Condition miteinander verglichen werden können.
(z.B. eine Firma soll über mehr Angestellte als Aktionäre verfügen)

OCL-String:
context c : Company inv BusinessConstraint: c.employees->size() > c.shareholders->size()

Testfall 17: test17.vocl

Dieser Testfall zeigt die Konvertierung für Methdonaufrufe auf ClassifierRoles (hier eine Methode ohne Parameter).
Für den möglichen Rückgabewert einer Methode kann eine beliebige Variable definiert werden, die in der Condition über Vergleichsoperatoren weiter spezifiziert werden kann.
Dabei kann bei Methodenaufrufen die Navigaton auch von ClassifierRoles beginnen, die im Context über den Namen als StartClassifierRole angegeben wurden.

OCL-String:
context p : Person inv positiveAge: p.getAge( ) > 0

Testfall 18: test18.vocl

Hier werden ebenfalls Methodenaufrufe getestet. Dieses Mal hat die Methode mehrere Parameter und Kind des Constraints ist post. Auf diese Weise kann der Rückgabewert nach der Methodenausführung in einer Condition näher spezifiziert werden.

OCL-String:
context Person::income(a, b, c) post : self.income(a,b,c) = 500

Testfall 19: test19.vocl

Dieses Beispiel zeigt, dass Methoden auch nach Navigation über AssociationEndRoles erreicht und aufgerufen werden. Die Behandlung der ReturnVariable des Methodenaufrus ist analog zum letzten Testfall.

OCL-String:
context son : Person inv : son.father.getAge( ) > 10

Testfall 20: test20.vocl

Dieses Beispiel zeigt, dass im selben Constraint mehrere Methoden definiert werden können.
Die Resultate der beiden Methodenaufrufe können in der Condition über ein Vergleichsoperator miteinander in Beziehung gesetzt werden.

OCL-String:
context p : Person inv : p.getAge( ) = p.getNumberOfCars( )

Testfall 21: test21.vocl

Dieser Testfall zeigt, dass in der Condition auch definierte Parameter-Variablen für die Formulierung von Bedingungen verwendet werden können.
So kann in diesem Bsp. für Methodenaufrufe ausgedrückt werden, dass der Restwert eines Autos niedriger ist als sein Kaufpreis.

OCL-String:
context Car inv : self.getValueOfCar(purchpase_price) < purchpase_price

Testfall 22: test22.vocl

Dieser Testfall zeigt, dass man sich für Variablen in der Condition über @pre auf den Wert der Variablen vor der jeweiligen Operation beziehen kann.
Dazu kann man im ConditionsDialog, in dem man bereits definierte Variablen über eine DropdownList als Operand auswählen kann, die Variable mittels einer CheckBox als @pre-Variable kennzeichnen.

OCL-String:
context Person::birthdayHappens( ) post : self.birthdayHappens( ) > self.age@pre

Testfall 23: test23.vocl

Dieser Testfall dient dazu zu zeigen, dass alle beschrieben Konstrukte kombiniert innerhalb eines Constraints angewendet werden können und ggbf. „verundet“ werden.
Für die Konvertierung ist lediglich erforderlich eine korrekte Startnavigation zu gewährleisten.

OCL-String:
context p : Person inv : p.calculateSalary(age,motivation) = p.mother.salary and p.children->isEmpty()

Testfall 24: test24.vocl

Der Testfall zeigt die die Verwendung des Boolschen Ausdruckes "implies". Dabei wird innerhalb der Teilausdrücke von "implies" Attributnavigation verwendet.
Für Teilausdrücke wird stets eine Klammerung angewendet, die die mögliche Rekursion der Teilausdrücke berücksichtigt.
In unserem Editor besteht die Möglichkeit allgemein Expressions entweder horizontal oder vertikal zu splitten, je nachdem wie es für die Sruktur der Teilausdrücke günstig erscheint (dabei kann man das Resultat einer Veränderung des Splits schon während des Bearbeitens im Dialog beobachten (Preview)).

OCL-String:
context Person inv : (self.isMarried=true implies self.isSingle=false)

Testfall 25: test25.vocl

Dieser Testfall zeigt die Verwendung des Boolschen Ausdrucks "xor". Innerhalb der Teilausdrücke wird eine Naviagation zu Attributen sowie zu einer SetClassifierRole mit Operation "not Empty" verwendet. Dies soll als Beispiel dafür dienen, dass die VOCL-Konstrukte aus der Ausbaustufen I und II nun innerhalb der Subexpression beliebig miteinander kombiniert werden.
Der Testfall zeigt auch, dass VOCL-Konstrukte, die auf der gleichen Rekursionsebene liegen, miteinander verundet werden (->self.isDead=true and self.isUnemployed=true). Auf diese Weise können implizite AND - Expressions definiert wird.

OCL-String:
context Person inv : ((self.isUnemployed=true and self.isDead=true) xor self.employer->notEmpty())

Testfall 26: test26.vocl

Dieser Testfall zeigt, dass Boolsche Ausdrucke auch nebeneinander existieren können. Resultat ist eine And-Expression der Boolschen Sub-Expressions. Dafür wurde die Verundung der letzten beiden Testfälle modelliert(letzter leicht modifiziert).

OCL-String:
context Person inv : ((self.isUnemployed=true xor self.employer->notEmpty()) and (self.isMarried=true implies self.isUnemployed=false))

Testfall 27: test27.vocl

Dieser Tesfall zeigt, dass mehrere Constraints (mit Boolschen Ausdrücken) im selben File modelliert werden können.
Alle vorhandenen Constraints werden getrennt konvertiert und anschliesend verundet.

OCL-String:
context Person inv Constraint1: (self.isMarried=true implies self.isSingle=false)
and
context Person inv Constraint2: (self.isUnemployed=true xor self.employer->notEmpty())

Testfall 28: test28.vocl

Dieser Testfall zeigt das Summieren von Attributen in Collections. Es existiert eine Variable, die das zu summierende Attribut repräsentiert als auch eine Variable für die eigentliche Summe. Sie kann in der Condition näher spezifiziert werden.

OCL-String:
context Person inv MoneyEarning: self.child.job.salary->sum() > 2000$

Testfall 29: test29.vocl

Dieser Testfall zeigt die Verwendung einer If-Then-Else-Expression. Im "If"-Teil wird eine Bedingungen über Attributnaviagtion ausgedrückt. Gleichzeitig wird eine Returnvariable für eine Methode deklariert, die in der "Then"- bzw. "Else"-Expression verwendet wird, um die unterschiedlichen Rückgabbereiche der Method zu spezifizieren.

OCL-String:
context Person inv : (if (self.isMarried=true and self.isUnemployed=true) then self.getIncome( ) >= 400 else self.getIncome( ) < 400)

Testfall 30: test30.vocl

Dieser Testfall zeigt, dass die in Ausbaustufe 3 dazugekommenen Expressions auch verschachtelt angewendet werden können. Dafür wird eine rekursive Auswertungsstrategie verwendet, die auch die erforderliche Klammerstruktur berücksichtigt.
Auch zeigt der Testfall, dass mit Hilfe einer "let-Expression" Variablen definiert werden können, die dann in den zugehörigen Expressions referenziert und in Conditions verwendet werden können.

OCL-String:
context Person inv : (let income:double = self.job.salary in (if self.isUnemployed=true then income >= 400 else income < 400))

Testfall 31: test31.vocl

Dieser Testfall zeigt die Anwendung eines etwas komplexere Beispiels für eine BoolExpression ("implies").
Hier ist ersichtlich, dass es möglich ist für jede Expression eine eigene Condition zu definieren.

OCL-String:
context Person inv GermanLaw: ((self.husband->notEmpty() implies self.husband.age >= 18) and (self.wife->isEmpty() implies self.wife.age >= 18))

Testfall 32: test32.vocl

Dieser Testfall zeigt nochmal die Konvertierung für eine verschachtelte BoolExpression.
Die Ebenen der Verschachtelung werden durch eine abwechselnd unterschiedliche Hintergrundsfarbe gekennzeichnet. Es ist möglich, die Trennlinie für die Expressions, die die Verhältniss für die Subexpressions angibt, über den Dialog einzustellen.
Dieses Beispiel zeigt auch, dass die Variablen in unserer bisherigen Implementierung bezüglich der Konvertieurng als global gültig betrachtet werden. Würde man für das Attribut „age“ jeweils die gleiche Variable x wählen, würden sich noch ein zusätzlicher Kontext für die Variable erschliessen und man würde dieses Resultat für den OCL-String erhalten:
context self : Person inv : ((self.isMarried=true) implies ((self.husband.age and self.wife.age >= 18) or (self.wife.age and self.husband.age >= 18))

OCL-String:
context Person inv : (self.isMarried=true implies (self.wife.age >= 18 or self.husband.age >= 18))

Testfall 33: test33.vocl

Dieser Testfall zeigt die Verwendung eines Negations-Ausdruckes (Not-Exressions). Dazu wird der letzte Testfall modifiziert: bei veränderter Ausgangsbedingung soll die implizierte Aussage in negierter Form vorliegen. Der resultierende OCL-String kennzeichnet die neue negierte Aussage mit „not (...)“.

OCL-String:
context Person inv : (self.isMarried=false implies not(self.wife.age >= 18 or self.husband.age >= 18))

Testfall 34: test34.vocl

Dieses Beispiel dient dem letzten Testeines komplexen Constraints, in dem alle möglichen Expressions verwendet werden. Das Constraint macht inhaltlich zwar keinen Sinn, jedoch lässt sich explizit die Korrektheit der Verschachtelung und Klammerung gut nachvollziehen.

OCL-String:
context Person inv : (if (self.1=1 or (self.getIncome( ) > 500 and self.3=3 and self.2=2)) then (if self.4=4 then self.5=5 else self.6=6) else (self.7=7 xor not(self.8=8 implies self.9=9)))