PES
Technische Universität Berlin
Software Engineering for Embedded Systems Group – Prof. Dr. Sabine Glesner
www.pes.tu-berlin.de Secr. TEL 12-4 Ernst-Reuter-Platz 7 10587 Berlin

Softwaretechnik und Programmierparadigmen WiSe 2014/2015

Prof. Dr. Sabine Glesner
Joachim Fellmuth joachim.fellmuth@tu-berlin.de
Dr. Thomas Göthel thomas.goethel@tu-berlin.de
Lydia Mattick lydia.mattick@tu-berlin.de
Tutoren

Übungsblatt 11 **Ausgabe: 08.01. (Besprechung: 12.01. und 13.01.)**

Model Checking mit GEAR

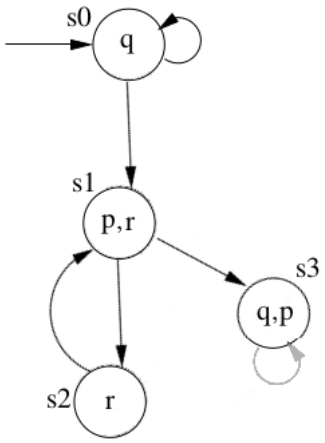
Das Plugin **GEAR** erweitert **jABC** um ein Verifikationstool zur Qualitätsicherung. Dazu interpretiert GEAR das Modell als eine Kripke Struktur und erlaubt darauf die Formulierung von CTL-Formeln.

Achtung: Das Plugin ist noch nicht vollkommen ausgefreift. So betrachtet es bei der Verifizierung eurer logischen Formel ausschließlich die von euch definierten APs (atomic propositions) an den SIBs und deren Transitionen. Der Zustand von Variablen wird nicht betrachtet.

jABC bietet zwei verschiedene Arten die CTL-Formeln anzugeben. Die Syntax, die im Tutorium genutzt wird kann so eingestellt werden: Im Menü unter Plugins → GEAR → Preferred syntax → Default (Mixfix) auswählen.

1. Kripke Struktur von Übungsblatt 10

Gegeben sei folgende Kripke Struktur als Graph.



q,p,r = Propositions

Überprüfe mit GEAR für welche der nachfolgend aufgelisteten Formeln das abgebildete Zustandsübergangssystem ein Modell darstellt.

a) **AF** $(q \wedge p)$ gilt nicht

b) **AG** $(p \Rightarrow$ **AF** $(p \wedge r))$ gilt nicht

c) **A** $[q$ **U** $r]$

d) **A** $[q$ **W** $r]$

e) **AG** $(p \Rightarrow$ **AG** $(p \vee q))$

f) **AG**(**EF**$(\neg r)$)

g) **AG**$((p \wedge q) \Rightarrow$ **AG**$(r))$

AF = Always Future
AG = Always Globally
AWU = Always weak until

A - Allquantor
E - Existenzquantor

Gear Syntax:
& - und
| - oder
! - not

2. Verifikation eines Werkstatt-Workflows

Betrachtet auf dem vorliegenden Modell, das einen Workflow in der Werkstatt Gomez beschreibt, mit Hilfe von CTL-Formeln die folgenden Aussagen. Überlegt euch dazu im Vorfeld eine geeignete Beschriftungsfunktion L. Verifiziert die Aussagen mit GEAR oder begründet warum sie nicht erfüllt werden können.

a) Ein Mitarbeiter kann nur einem bereits erstellten Auftrag zugewiesen werden.

b) Einem Auftrag kann nur ein Mitarbeiter zugewiesen werden.
AG(employeeAssigned => EF(assignEmployee))

c) Ein Auftrag kann beliebig viele Tasks umfassen.
EG(EF(addTask) & EF(orderFinished)) —> geht nicht wegen unendlicher Pfade

d) Ein Auftrag wird in jedem Fall irgendwann beendet sein.
AG(AF(orderFinished)) —> Problem: unendliche Pfade, d.h. gilt nicht!

e) Solange der Auftrag nicht beendet ist, kann ein Task hinzugefügt werden.
AWU(EF(addTask), orderFinished) —> impliziert auch c)