

Komposition von Benutzungsmodellen: Operatoren und Anwendungsbeispiele

Autoren:

Thomas Bauer
Robert Eschbach
Tanvir Hussain
Johannes Kloos
Fabian Zimmermann

IESE-Report Nr. 112.09/D
Version 1.0
Januar 2009

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer-Gesellschaft. Das Institut transferiert innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach (geschäftsführend)
Prof. Dr. Peter Liggesmeyer
Fraunhofer-Platz 1
67663 Kaiserslautern

Abstract

Beim modellbasierten Testen komplexer Systeme stößt man häufig auf das Problem, dass die Modellierung des Benutzer- und Umgebungsverhaltens in einem einzigen Modell nur sehr umständlich durchzuführen ist und dass die Möglichkeiten zur Wiederverwendung und Modellevolution sehr gering sind. Aus diesem Grund untersuchen wir, in welcher Form man Benutzungsmodelle mit Hilfe kompositionaler Methoden aus einfacheren Modellen zusammensetzen kann.

Schlagworte: Statistical Testing, Model-based testing, compositional approach, DNT, reliability

Inhaltsverzeichnis

1	Einige Konzepte für Operatoren und Modelle	1
1.1	Eigenschaften von Modellen	1
1.1.1	Terminierende und nicht-terminierende Modelle	1
1.2	Fairness	2
1.3	Eigenschaften von Kompositionsoperatoren	2
1.3.1	Erhalt der Markov-Eigenschaft, starke Trace-Erhaltung	2
1.3.2	Freie Auswahl	3
1.3.3	Erhalt von sinnvollen Wahrscheinlichkeiten	3
1.3.4	Terminierende und nicht-terminierende Eingabemodelle	4
1.3.5	Terminierende und nicht-terminierende Ausgabemodelle	4
2	Beschreibung der Kompositionsoperatoren	5
2.1	SWITCH Operator	5
2.2	Parallelkomposition	7
2.3	transformation der Modellstruktur	9
2.4	Situationsverfeinerung	11
2.5	Stimulusverfeinerung	14
2.6	Zufällige mutation der Modelstruktur	16
2.7	Stimulusumbenennung	19
2.8	Situationen Entfernen	20
2.9	Stimulus Entfernen	22
2.10	Synchronisation	24
2.11	Fehlbenutzung	26
2.12	ROBUSTheitstest	28
2.13	Terminierend Machen	29
2.14	Nicht-terminierend Machen	30
2.15	BARRIER	32
2.16	SUSPEND	33

1 Einige Konzepte für Operatoren und Modelle

Die entwickelten Operatoren ermöglichen die Erstellung von Stimulationsmodellen aus existierenden Stimulationsmodellen, wie in [AP2-Modelle] und [AP2-Operatoren] beschrieben wurde.

Die Operatoren müssen die folgenden Eigenschaften erfüllen:

Die resultierenden Stimulationsmodelle müssen die Markov-Eigenschaft erfüllen (genaueres dazu in 1.3.1).

Erhaltung des Benutzungsprofils (so weit wie möglich)

Generierung von sinnvollen Testskripten (falls Eingabemodelle Testskripte liefern).

Zunächst müssen wir einige Eigenschaften von Markov-Modellen betrachten, was in Kapitel 2 geschehen wird. Damit ist es möglich, verschiedene Eigenschaften von Kompositionsoperatoren zu beschreiben. Dies geschieht in Kapitel 3. Die konkreten Operatoren werden in Kapitel 2 beschrieben.

1.1 Eigenschaften von Modellen

Um das Verhalten der Kompositionsoperatoren genau beschreiben zu können, müssen wir zunächst einige Eigenschaften von Modellen beschreiben.

1.1.1 Terminierende und nicht-terminierende Modelle

Üblicherweise verlangt man von Markov-Benutzungsmodellen, dass die erzeugten Testfälle niemals unendlich lang werden. Als stochastische Aussage verkleidet können wir diese Bedingung so formulieren: Jeder Random Walk durch das Markov-Modell muss mit Wahrscheinlichkeit 1 den Endzustand erreichen. Solche Markov-Modelle nennen wir terminierend. *Das Markov-Modell muss einen Zustand ohne ausgehende Transitionen (bzw. nur mit END-Selbsttransition) besitzen, der aus allen anderen Zuständen (mit Wahrscheinlichkeit > 0) erreichbar ist.*

In einigen Fällen ist es notwendig Modelle ohne Endzustand zu verwenden. Will man beispielsweise bestimmte Eingaben eines Fahrers mit bestimmten Wetterdaten kombinieren, ist es nicht ersichtlich, weshalb das Wetter irgendwann konstant bleiben soll. Daher bietet es sich häufig an, bestimmte Umwelteinflüs-

se als nicht-terminierende Markov-Modelle zu modellieren. Diese Modelle haben die Eigenschaft, dass sie keinen Endzustand besitzen, und dass sie damit immer unendliche Stimulus-Folgen erzeugen können.

Bei der Komposition verschiedener Markov-Modelle treten bezüglich Terminierung einige Fragestellungen auf. Beispielsweise ist nicht unmittelbar klar, was passieren soll, wenn eines von zwei parallel laufenden Markov-Modellen den Endzustand erreicht: Soll der Testfall an diesem Punkt zu Ende sein, oder soll das andere Modell noch zu Ende laufen?

Die Frage, wann die Testfall-Erzeugung bei mehreren Eingabemodellen zu Ende ist, wird weiter unten als Eigenschaften von Operatoren behandelt.

1.2 Fairness

Wenn wir ein Modell S benutzen, um zwischen mehreren Eingabemodellen M_1 bis M_n auszuwählen, so erwarten wir, dass jedes der Eingabemodelle die Möglichkeit haben sollte, eine beliebig lange Stimulus-Sequenz auszugeben. Dazu muss S jedes der M_i immer wieder auswählen, egal wie S durchlaufen wird.

Wir bezeichnen diese Eigenschaft von S als Fairness und definieren sie wie folgt: Für jedes M_i muss S mit Wahrscheinlichkeit 1 eine Stimulusfolge erzeugen, so dass M_i unendlich oft zum Zug kommt.

1.3 Eigenschaften von Kompositionsoperatoren

In diesem Kapitel sollen einige Eigenschaften von Kompositionsoperatoren beschrieben werden. Wir werden uns hier mit einfachen Charakterisierungen dieser Eigenschaften begnügen; die Details finden sich in [AP5].

1.3.1 Erhalt der Markov-Eigenschaft, starke Trace-Erhaltung

Eine der Forderungen an die Kompositionsoperatoren war, dass diese „die Markov-Eigenschaft erhalten sollten“. Wie im Rahmen des Projektes festgestellt wurde, ist diese Forderungen mehrdeutig. Auf einer sehr einfachen Ebene kann man diese Aussage wie folgt interpretieren: Das durch die Komposition erzeugt Modell muss wiederum eine Markov-Kette sein.

Es hat sich allerdings gezeigt, dass diese Eigenschaften wesentlich zu schwach ist: Wir könnten beispielsweise einen Operator konstruieren, der mit einem Markov-Modell verbunden ist, aber von diesem nur ganz bestimmte Stimuli einliest und alle anderen Stimuli nicht zulässt. Auf diese Art und Weise werden die möglichen Verhalten eines Teilmodells stark eingeschränkt; insbesondere kann nicht mehr jeder Testfall des Teilmodells zur Ableitung eines Gesamt-

Testfalls verwendet werden. Mehr noch, es ist sogar möglich, solche Operatoren zu konstruieren, die in für das Teilmodell unkontrollierbarer Weise bestimmte Transitionen zulassen oder auch nicht. Solche Operatoren werden als nicht gewünscht betrachtet.

Aus diesem Grund haben wir uns für eine Charakterisierung dieser Eigenschaft entschieden, die wir als „starke Trace-Erhaltung“ bezeichnen. Kurz gesagt beschreibt diese Eigenschaft die folgende Situation: Wenn man einen beliebigen Trace eines beliebigen Teil-Modells auswählen kann, dann ist die Wahrscheinlichkeit, dass dieses Teilmodell diesen Trace erzeugt, genau so groß wie die Wahrscheinlichkeit, dass das Gesamtmodell einen Gesamt-Trace erzeugt, an dessen Entstehung genau der betrachtete Trace des Teilmodells beteiligt war.

1.3.2 Freie Auswahl

Wir sagen, ein Operator hat freie Auswahl bezüglich eines Eingabemodells, wenn für alle Stimulus-Sequenzen des Eingabemodells folgendes gilt: Wird diese Stimulus-Sequenz in den Operator eingespeist und das Eingabe-Modell liefert einen weiteren Stimulus, so wird der Operator, egal welche Eingaben von anderen Modellen bisher aufgetreten sind, diesen Stimulus akzeptieren.

Hat ein Operator diese Eigenschaft nicht, so können verschiedene Probleme auftreten. Beispielsweise könnte man auf diese Art erzwingen, dass eine bestimmte Transition des Teilmodells gewählt wird, da alle anderen Kanten mit Stimuli beschriftet sind, die vom Kompositionsoperator nicht akzeptiert werden. Ein Beispiel eines solchen Operators könnte wie folgt aussehen: Es werden zwei Eingabemodelle benutzt, M und S. Immer, wenn S „ja“ ausgibt, akzeptieren wir einen beliebigen Stimuli von M (und geben ihn aus), sonst akzeptieren wir ausschließlich den Stimulus „x“. Wenn nun M immer zwei mögliche Ausgaben hat, nämlich „x“ und „y“, so ändert sich die Wahrscheinlichkeit, „y“ auszugeben, je nachdem, was S ausgibt.

1.3.3 Erhalt von sinnvollen Wahrscheinlichkeiten

Manche Operatoren erlauben es, zusätzliche Transitionen in einem Modell einzuführen. Solche Operatoren werden normalerweise keine starke Trace-Erhaltung ermöglichen. Allerdings ist es normalerweise möglich, immer noch sinnvolle Aussagen über die Wahrscheinlichkeitsverteilungen der entstandenen Traces zu machen. Andererseits muss man die entstandenen Testfälle mit Vorsicht behandeln, da nicht jeder Testfall sich auf eine Usage des Eingabe-Modells zurückführen lässt.

1.3.4 Terminierende und nicht-terminierende Eingabemodelle

Verschiedene Operatoren stellen an die Eingabemodelle bestimmte Bedingungen. So verlangen einige Operatoren von bestimmten Eingabemodellen terminierend bzw. nicht-terminierend und/oder fair zu sein, um bestimmte Eigenschaften zu erhalten.

1.3.5 Terminierende und nicht-terminierende Ausgabemodelle

Zusätzlich zu den bisherigen Bedingungen können wir im Allgemeinen Aussagen darüber treffen, ob das Ergebnis einer Komposition ein terminierendes oder ein nicht-terminierendes Markov-Modell sein wird. Allerdings müssen, um diese Aussage treffen zu können, oft verschiedene Vorbedingungen an die Eingabemodelle erfüllt sein.

Um bei der Komposition mehrerer terminierender Benutzungsmodelle (beispielsweise durch Parallelkomposition) festzulegen, wann das komposite Modell terminiert, gibt es mehrere Ansätze:

1. Das komposite Modell terminiert, wenn alle Teilmodelle terminiert haben. Dieser Ansatz kann in zwei Varianten unterteilt werden:
 - a. Terminierte Teilmodelle kommen nicht mehr zum Zug.
 - b. Man legt fest, dass nur ein Modell terminierend sein darf (weitere terminierende Modelle werden in nicht-terminierende umgewandelt), und ernennt dieses Modell zum Terminations-Master. Das komposite Modell terminiert genau dann, wenn
3. Das komposite Modell terminiert, wenn eines der Teilmodelle terminiert. Dieser Ansatz zerstört die starke Trace-Erhaltung.

Unserer Einschätzung nach ist der STUTTER-Ansatz der beste: In vielen Fällen ergibt sich dadurch automatisch starke Trace-Erhaltung, der Usage-Begriff bleibt erhalten und die Implementierung ist einfach. Der zweitbeste Ansatz ist der Terminations-Master-Ansatz – er hat zwar eine sehr klare Semantik, garantiert Trace-Erhaltung, aber nur für den Terminationsmaster. Bei Operatoren, für die beide Ansätze sinnvoll sind, wählen wir deshalb den STUTTER-Ansatz.

2 Beschreibung der Kompositionsoperatoren

2.1 SWITCH Operator

ID: SWITCH

Alternative Bezeichnungen:

Steuerung von Benutzungsmodellen, Usage Model Control, simple synchronize, model switch

Beschreibung:

Synchronisiere die Stimulation durch mehrere parallel laufende Benutzungsmodelle P_1, \dots, P_n durch ein Steuerungsmodell S . Jeweils nur ein Benutzungsmodell P_i kann aktiv sein und Stimulationssequenzen erzeugen.

Superoperatoren:

-

Suboperatoren:

WEIGHTED_PARALLEL

Eingaben:

Benutzungsmodelle P_1, \dots, P_n

Steuerungsmodell S

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

S muss fair sein.

Eigenschaften:

R ist terminierend, wenn mindestens eines der Benutzungsmodele P_i terminierend ist. Falls alle Benutzungsmodele nicht-terminierend sind, ist auch R nicht-terminierend.

Bei fairem Steuermodell S besitzt dieser Operator die starke Trace-Erhaltung für alle terminierenden Benutzungsmodele P_i .

Semantik:

Das komposite Modell führt immer wieder folgende Schritte durch:

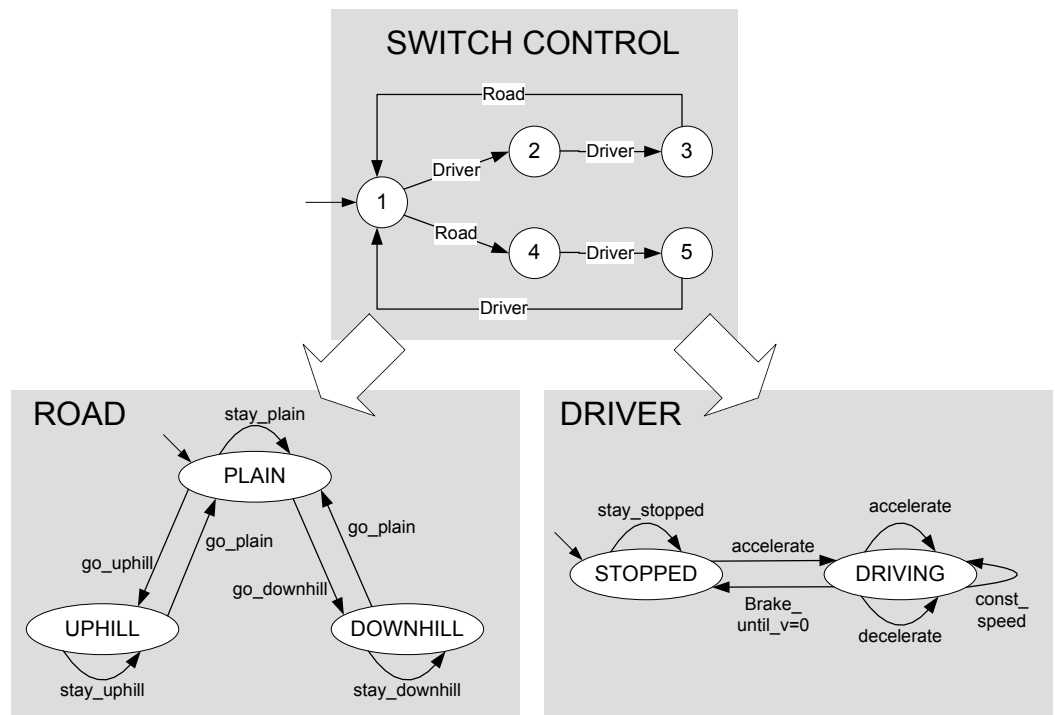
Falls alle terminierenden Teilmodelle terminiert sind, gibt END aus und terminiere. Sonst führe einen Schritt im Steuerungsmodell aus, um zu entscheiden, welches Modell als nächstes am Zug ist.

Führe einen Schritt dieses Modells aus. Falls das Ergebnis STUTTER oder END ist, markiere dieses Modell als „terminiert“, sonst gib das Ergebnis aus.

Wenn alle terminierenden Teilmodelle als „terminiert“ markiert sind, gibt das komposite Modell END aus.

Spezifikation in π :

Beispiel:



Im Beispiel werden die Benutzungsmodelle ROAD und DRIVER gesteuert durch das Steuerungsmodell SWITCH CONTROL. Nur jeweils ein Benutzungsmodell kann aktiv sein und Stimuli generieren. Das Steuerungsmodell legt fest, welches Benutzungsmodell aktiv ist. Hier wechseln sich ROAD und DRIVER nach den in SWITCH CONTROL definierten Regeln ab.

Parallelkomposition

ID: WEIGHTED_PARALLEL

Alternative Bezeichnungen:

Gewichtetes Produkt

Beschreibung:

Gewichtete Komposition von mehreren parallel laufenden Benutzungsmodellen gemäß einer konstanten Wahrscheinlichkeitsverteilung, die beschreibt, wie hoch die Gewichtung der einzelnen Benutzungsmodelle ist.

Superoperatoren:

Dieser Operator ist eine Spezialisierung des Switch-Operators. Dieser Operator besitzt implizit ein sehr einfaches Steuermodell. Dieses Steuermodell besitzt hier genau einen Zustand und für jedes Subbenutzungsmodell eine Selbstkante.

Suboperatoren:

-

Eingaben:

Markov-Ketten der Benutzungsmodelle P_1, \dots, P_n

Gewichtung der einzelnen Benutzungsmodelle als konstante Wahrscheinlichkeitsverteilung: w_1, \dots, w_n

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

Die Gewichtungen w_i müssen alle größer 0 sein.

R ist terminierend, wenn eines der Eingabemodelle terminierend ist.

Eigenschaften:

R ist terminierend, wenn mindestens eines der Benutzungsmodelle P_i terminierend ist. Falls alle Benutzungsmodelle nicht-terminierend sind, ist auch R nicht-terminierend.

Dieser Operator besitzt die starke Trace-Erhaltung für alle terminierenden Benutzungsmodelle P_i .

Semantik:

Das komposite Modell führt immer wieder folgende Schritte durch:

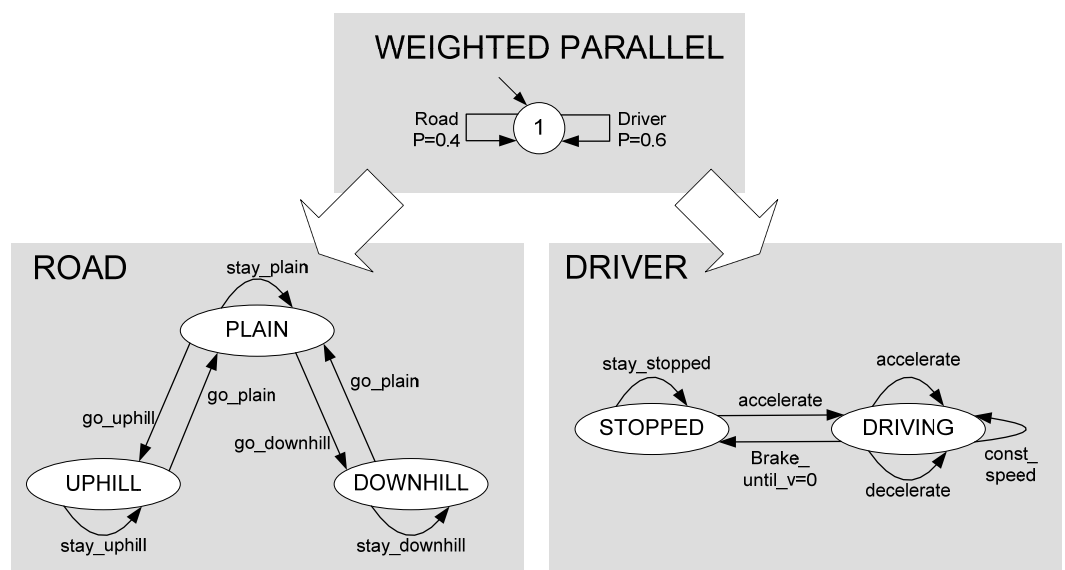
Falls alle terminierenden Teilmodelle terminiert sind, gibt END aus und terminiere. Sonst ermittle aufgrund der Gewichtung stochastisch, welches Modell als nächstes am Zug ist.

Führe einen Schritt dieses Modells aus. Falls das Ergebnis STUTTER oder END ist, markiere dieses Modell als „terminiert“, sonst gib das Ergebnis aus.

Wenn alle terminierende Teilmodelle als „terminiert“ markiert sind, gibt das Modell END aus.

Im π -Kalkül:

Beispiel:



Im Beispiel werden die beiden Benutzungsmodelle ROAD für die Straßenumgebung und DRIVER für die Fahrereingaben parallel komponentiert, wobei das Straßenmodell mit 40% gewichtet und das Fahrermodell mit 60%. Aus dem resultierenden Benutzungsmodell werden zu 60% Eingaben für den Fahrer und zu 40% Eingaben für die Straße generiert.

transformation der Modellstruktur

Name: *TRANSFORM*

Alternative Bezeichnungen:

Post Processing

Beschreibung:

Transformation eines Benutzungsmodells P durch Verarbeitung der erzeugten Stimulussequenzen durch einen Mealy Automaten. Das resultierende Benutzungsmodell erzeugt gültige Sequenzen bezüglich des Mealy Automaten M. Das Benutzungsprofil basiert auf dem P.

Superoperatoren:

-

Suboperatoren:

-

Eingaben:

Benutzungsmodell P

Deterministischer vollständiger Mealy Zustandsautomat, der die Transformationsregeln beschreibt: M

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

M muss total und deterministisch sein, jede Transition von M muss eine nicht-leere Ausgabe besitzen (für leere Ausgaben kann STUTTER verwendet werden)

Eigenschaften:

R ist terminierend wenn P terminierend ist.

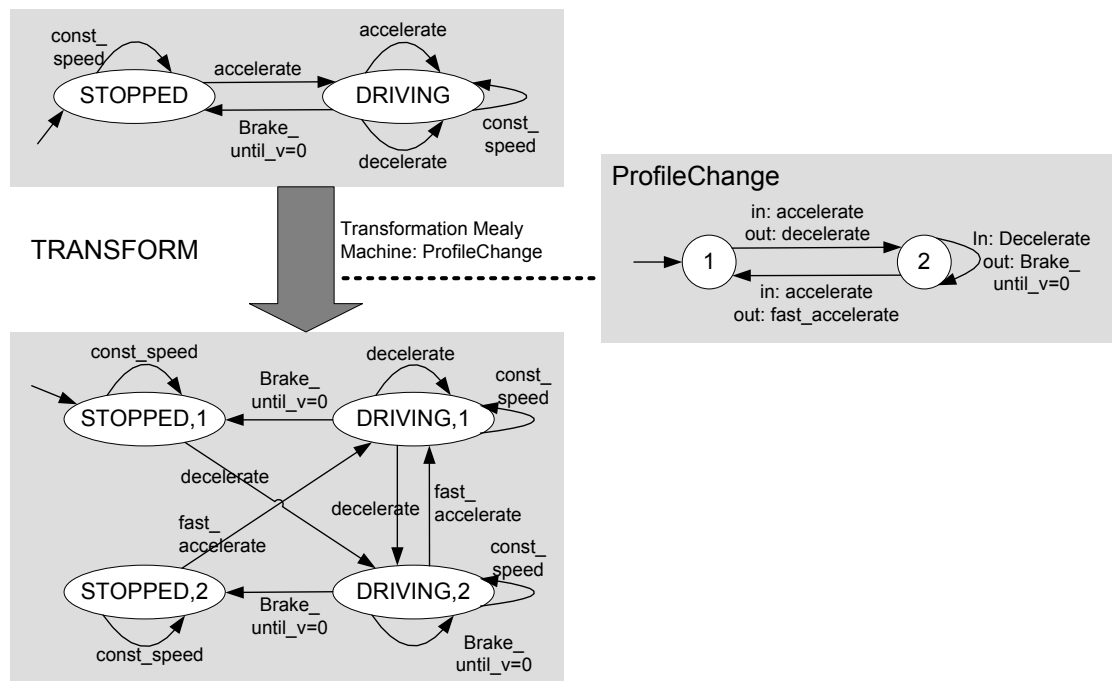
Operator besitzt starke Trace-Erhaltung.

Semantik:

Jeder Stimulus der von P erzeugt wird, wird als Eingabe von M verwendet. Die Ausgaben von M werden ausgegeben. Wenn P END ausgibt, wird kein weiterer Schritt ausgeführt. R gibt dann END aus.

Spezifikation in π :

Beispiel:



Im Beispiel werden die Stimulussequenzen aus dem Benutzungsmodell des Fahrers durch den Mealy Automaten ProfileChange verarbeitet. Einige Stimuli ändern sich. Das resultierende Benutzungsmodell ist abgebildet, es ist der Produktautomat von DRIVER und ProfileChange.

2.2 Situationsverfeinerung

Name: REFINE_SITUATIONS

Alternative Bezeichnungen:

Kantenverfeinerung, Verfeinerung von Transitionshistorien, Stimulussequenzverfeinerung

Beschreibung:

Eine Stimulussequenz ist ein Teilpfad eines Benutzungsmodells und beschreibt eine so genannte Situation. Die Stimulussequenz kann dabei eine Transition mehrfach traversieren. Der letzte Schritt (und damit der letzte Stimulus) wird durch Anwendung des Operators durch Einsetzen eines anderen Be-

nutzungsmodells verfeinert. Die Transitionsverfeinerung ist ein Sonderfall der Situationsverfeinerung, wo alle Traversierungen der gleichen Transition durch ein Modell verfeinert werden.

Superoperatoren:

-

Suboperatoren:

Stimulusverfeinerung

Zufällige mutation der Modelstruktur

Eingaben:

Benutzungsmodell, dessen Teil verfeinert werden sollen: P

Benutzungsmodell, das die Verfeinerung beschreibt: Q

Zu ersetzende Situation von P (die Situation „Transition: S“ beschreibt dabei jedes Auftreten der Transition, die vom Startzustand mit der Sequenz S erreicht werden kann)

Eingabe-Abbildung in

Default-Ausgabe out

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

Q muss terminierend sein

Bild(in) muss Teilmenge der erzwingbaren Traces von Q sein

Eigenschaften:

Wenn P terminierend ist, ist auch R terminierend.

Starke Trace-Erhaltung für P.

Semantik:

Anfangs wird das Modell P Schritt für Schritt ausgeführt. So lange keine Situation aus S erreicht ist, gib die von P erzeugten Stimuli aus. Wird aber durch einen letzten Stimulus s eine Situation aus S erreicht, so passiert folgendes:

Es wird $\text{in}(s)$ ermittelt. Die so ermittelte Stimulusfolge wird benutzt, um einen Zustand von Q zu ermitteln.

Eine Instanz von Q wird in dem eben ermittelten Zustand gestartet.

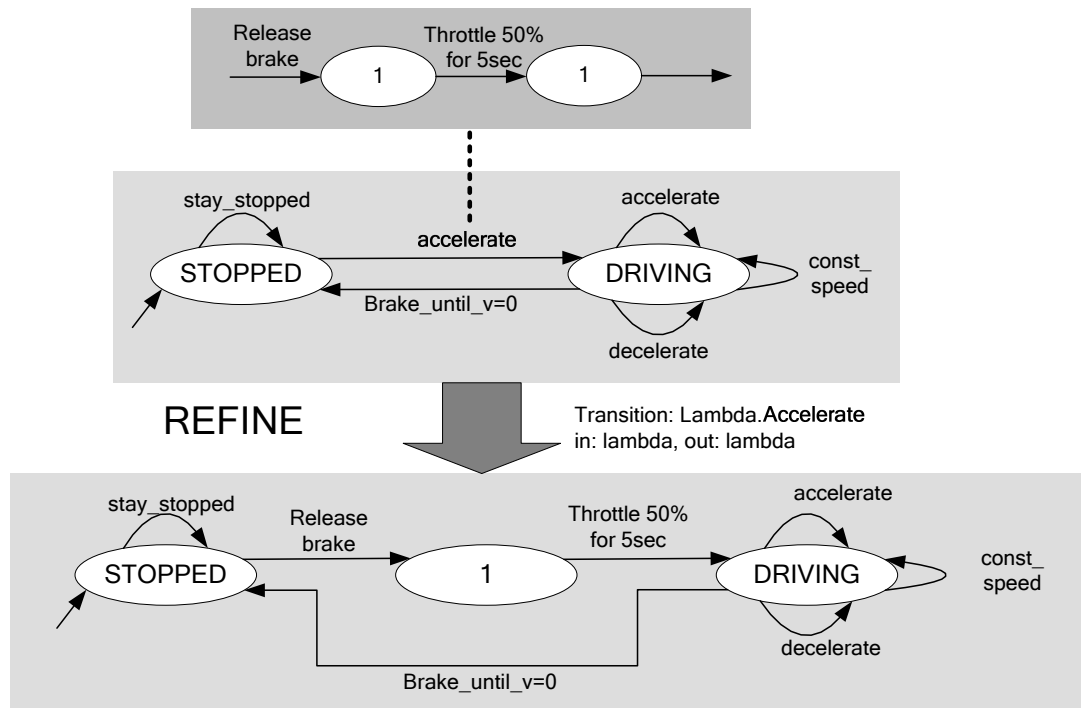
So lange sich Q nicht in den Endzustand begibt, wird immer wieder ein Stimulus von Q erzeugt und ausgegeben.

Sobald sich Q in den Endzustand begibt, wird der letzte erzeugte Stimulus s' verwendet, um einen Zustand in P zu ermitteln. Dazu suchen wir eine Transition, die von dem Zustand, in dem sich P befindet, ausgeht und mit s' beschriftet ist. Der Zielzustand dieser Transition ist der gesuchte Zustand. Gibt es keine solche Transition, wird stattdessen die Transition verwendet, die mit out beschriftet ist; hat out den speziellen Wert „epsilon“, so ist der gesuchte Zustand derjenige, in dem sich P gerade befindet.

P wird, wie am Anfang beschrieben, in dem eben bestimmten Zustand weiterlaufen gelassen.

Spezifikation in π :

Beispiel:



Das Beispiel zeigt einen Sonderfall des Operators. Hier wird die Transition für Beschleunigen „Accelerate“ des Zustands „STOPPED“ bei jedem Durchlauf verfeinert. Damit wird in diesem Beispiel nicht unterschieden, wie oft diese Transition schon traversiert wurde. Das einzusetzende Modell ist sehr einfach und besteht nur aus zwei Transitionen „Brake Release“ und „Throttle 50% for 5sec“. Bei der Anwendung des Operators wird die genannte Transition durch die Sequenz „Brake Release“ und „Throttle 50% for 5sec“ ersetzt.

2.3 Stimulusverfeinerung

Name: REFINES_STIMULUS

Alternative Bezeichnungen:

Stimulus refinement, Event refinement

Beschreibung:

Verfeinert alle Transitionen mit einem definierten Stimulus eines Benutzungsmodells durch Einsetzen eines anderen Benutzungsmodells.

Superoperatoren:

REFINE_SITUATIONS

Suboperatoren:

RANDOM_MUTATION

Eingaben:

Benutzungsmodell, dessen Stimulus verfeinert werden sollen: P

Benutzungsmodell, das die Verfeinerung beschreibt: Q

Zu ersetzenden Stimulus s

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

Q muss terminierend sein

Eigenschaften:

Wenn P terminierend ist, ist auch R terminierend.

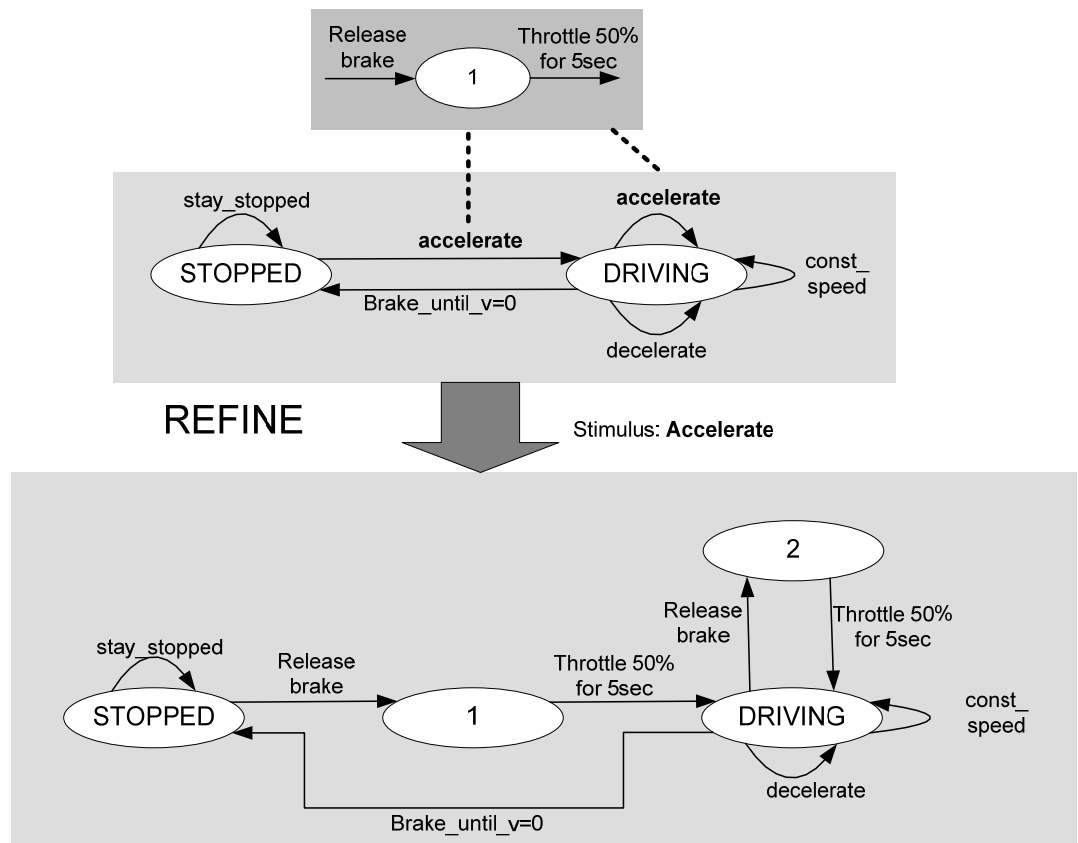
Starke Trace-Erhaltung für P und Q (für Trace-Erhaltung: Q muss mindestens einmal ausgeführt werden, mehrere Läufe von Q möglich).

Semantik:

Es wird das Benutzungsmodell P laufen lassen. Wird ein Stimulus außer s empfangen, so wird dieser ausgegeben; ansonsten wird das Ergebnis eines Laufes von Q ausgegeben.

Spezifikation in π :

Beispiel:



Im Beispiel wird der Stimulus für Beschleunigen „Accelerate“ verfeinert in die Sequenz „Brake Release“ und „Throttle 50% for 5sec“, welche verantwortlich ist um die Bremse zu lösen und das Gaspedal für 5 Sekunden zu 50% zu betätigen. Dazu werden alle Transition, die den Stimulus „Accelerate“ referenzieren, verfeinert.

2.4 Zufällige mutation der Modelstruktur

Name: RANDOM_MUTATION

Alternative Bezeichnungen:

Random mutation of the model structure

Beschreibung:

Zufällige Änderung von definierten Stimuli. Bei jeder Traversierung einer Transition mit dem definierten Stimuli wird zufällig entschieden, ob die Transition mutiert oder nicht. Durch die Mutation erhält die Transition einen neuen definierten Stimulus.

Superoperatoren:

REFINE_SITUATIONS

REFINE_STIMULUS

Dieser Operator lässt sich mit Hilfe des *REFINE_STIMULUS* Operators implementieren. Der zu mutierende Stimulus wird mit einem Modell verfeinert, das die Auswahl zwischen dem Originalstimulus und den möglichen Mutationen bietet.

Suboperatoren:

-

Eingabedaten:

Benutzungsmodell P

Stimulus s von P

Mutierter Stimulus: t

Mutationsrate: p

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

-

Eigenschaften:

Wenn Modell P terminierend ist, so ist auch Modell R terminierend.

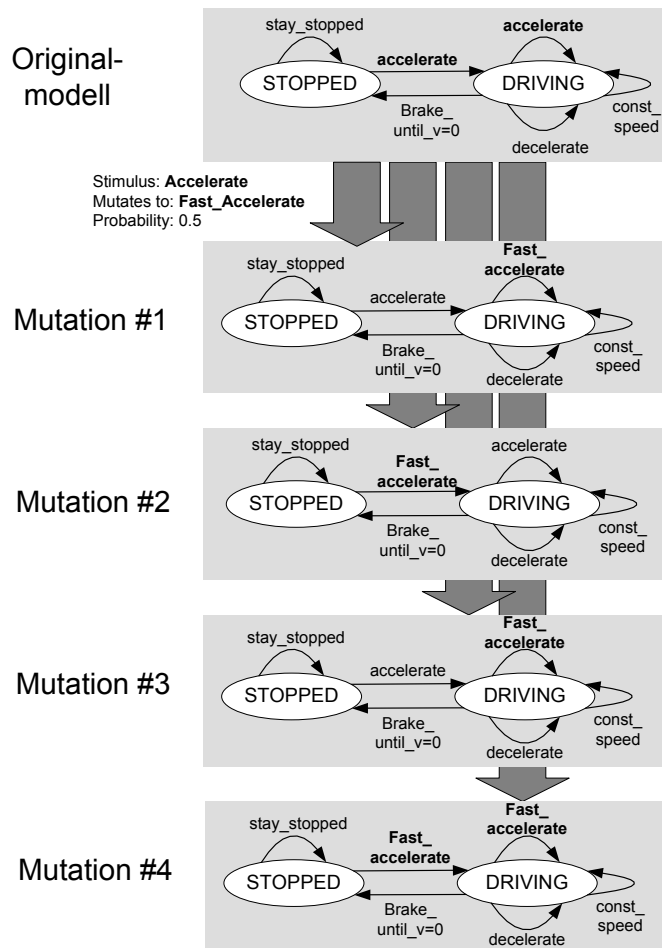
Dieser Operator besitzt die starke Trace-Erhaltung.

Semantik:

Die Stimuli werden im Originalmodell erzeugt. Sobald allerdings der zu mutierende Stimuli erzeugt wird, wird dieser nicht ausgegeben. Stattdessen wird mit Hilfe des Mutationsmodells entschieden, welcher Stimulus stattdessen erzeugt wird.

Spezifikation in π :

Beispiel:



Im Beispiel wird der Stimulus „accelerate“ zufällig mutiert. In jedem Schritt wird zufällig entschieden gemäß der gewählten Mutationsrate von 0.5 entschieden, ob die entsprechenden Transitionen mit dem Stimulus „accelerate“ mutieren. Hier sind 4 Mutationen dargestellt, wo verschiedene Transitionen im Benutzungmodell betroffen sind.

2.5 Stimulusumbenennung

Name: RENAME_STIMULUS

Alternative Bezeichnungen:

Event renaming

Beschreibung:

Änderung des Namens eines Stimulus und damit auch Änderung der Bezeichnung aller Transitionen, die diesen Stimulus verwenden.

Superoperatoren:

REFINE_SITUATIONS

REFINE_STIMULUS

Dieser Operator stellt die einfachste Form der Stimulusverfeinerung dar. Die Verfeinerung besteht einfach darin, einen bestimmten Stimulus umzubenennen.

Suboperatoren:

-

Eingaben:

Benutzungsmodell, dessen Stimulus verfeinert werden sollen: P

Zu ersetzenden Stimulus s

Ersatz: t

Ausgabe:

Benutzungsmodell R

Vor- und Nachbedingungen:

-

Eigenschaften:

Wenn Modell P terminierend ist, so ist auch Modell R terminierend.

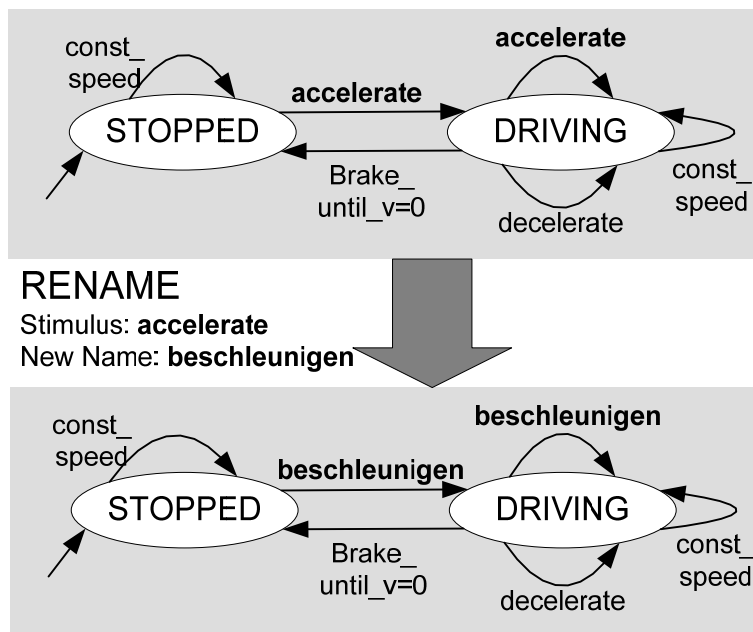
Dieser Operator besitzt die starke Trace-Erhaltung.

Semantik:

Es wird das Benutzungsmodell P laufen lassen. Wird ein Stimulus außer s empfangen, so wird dieser ausgegeben; ansonsten wird t ausgegeben.

Spezifikation in π :

Beispiel



Im Beispiel wird der Stimulus "accelerate" umbenannt in „beschleunigen“. Die entsprechenden Transitionen werden dann aktualisiert und mit den neuen Stimulusnamen versehen.

2.6 Situationen Entfernen

Name: REMOVE_SITUATIONS

Alternative Bezeichnungen:

Entfernen von Transitionshistorien, Entfernen von Stimulussequenzen

Beschreibung:

Eine Stimulussequenz ist ein Teilpfad eines Benutzungsmodells. Die Stimulussequenz kann dabei eine Transition mehrfach traversieren. Der letzte Schritt (und damit der letzte Stimulus) der Sequenz wird durch Anwendung des Operators

entfernt. Das Löschen von Transitionen ist ein Sonderfall der Situationsverfeinerung, wo die eine Transition des Benutzungsmodells entfernt wird.

Superoperatoren:

-

Suboperatoren:

REMOVE_STIMULUS

Eingaben:

Benutzungsmodell, dessen Teile modifiziert werden sollen: P

Zu ersetzende Situation / Stimulussequenz von P

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

-

Eigenschaften:

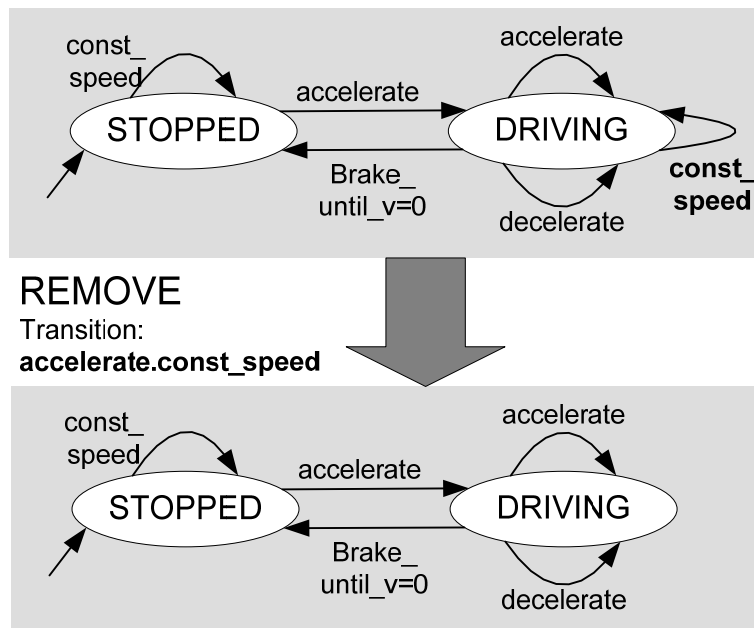
Dieser Operator kann nicht garantieren das R ein gültiges Modell ist und sollte deshalb nur mit größter Vorsicht verwendet werden.

Semantik:

Bestimmte Transitionen werden entfernt. Im schlimmsten Fall ist das Ergebnis kein gültiges Markov-Modell.

Spezifikation in π :

Beispiel:



Im Beispiel wird der Spezialfall beschrieben, bei einer Transition aus dem Benutzungsmodell entfernt wird. In dem Fall ist es die Transition „DRIVING.const_speed“, die durch die

2.7 Stimulus Entfernen

Name: REMOVE_STIMULUS

Alternative Bezeichnungen:

Stimulus löschen, remove event

Beschreibung:

Entfernung eines Stimulus aus der Schnittstelle des Benutzungsmodells und damit Entfernung aller Transitionen, die den Stimulus verwenden.

Superoperatoren:

REMOVE_SITUATION

Suboperatoren:

-

Eingaben:

Benutzungsmodell P

Stimulus von P: $\sigma \in \Sigma_p$

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

-

Eigenschaften:

Dieser Operator kann nicht garantieren das R ein gültiges Modell ist und sollte deshalb nur mit größter Vorsicht verwendet werden.

Semantik:

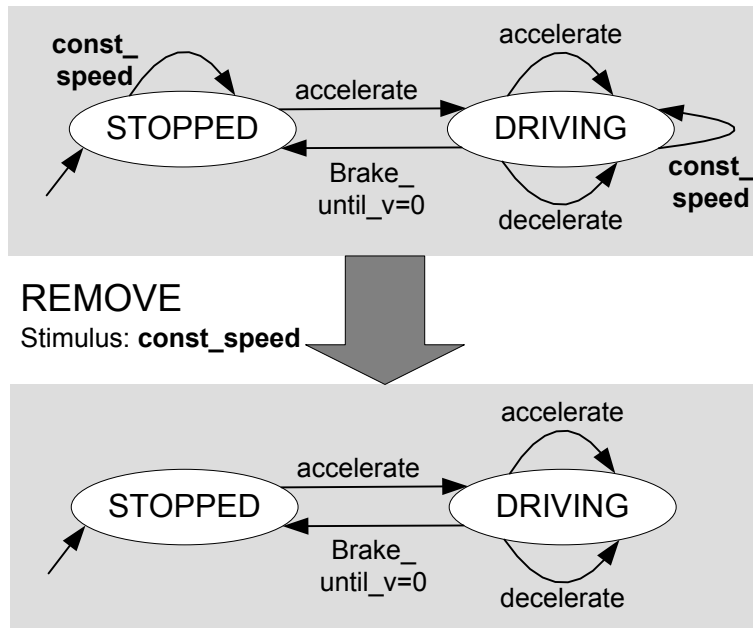
Bestimmte Transitionen werden entfernt. Im schlimmsten Fall ist das Ergebnis kein gültiges Markov-Modell.

Semantik:

Es wird das Benutzungsmodell P laufen lassen. Es werden alle Stimuli von P ausser σ vom Operator akzeptiert; die Annahme von σ wird verweigert.

Spezifikation in π :

Beispiel:



Im Beispiel wird der Stimulus für die konstante Geschwindigkeit "const_speed" entfernt. Die 2 Transitionen, die diesen Stimulus referenzieren werden daraufhin gelöscht.

Stimulussequenz „accelerate“ + „const_speed“ beschrieben wird.

2.8 Synchronisation

Name: SYNCHRONIZE

Alternative Bezeichnungen:

Steuerung von Benutzungsmodellen, Usage Model Control

Wird überarbeitet. Vergleiche Barrier und Suspend!

Beschreibung:

Synchronisiere die Stimulation durch mehrere parallel laufende Benutzungsmodelle durch einen Mealy Automaten. Dadurch können widersprüchliche, unmögliche oder fehlerträchtige Eingabesequenzen vermieden werden.

Eingaben:

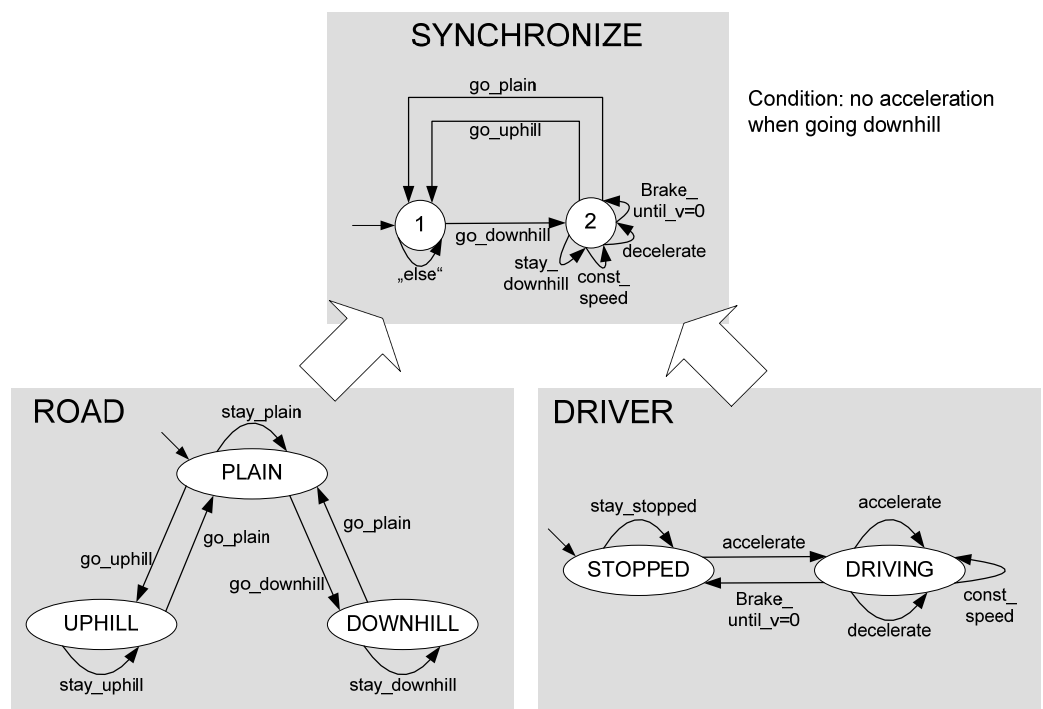
Benutzungsmodelle P_1, \dots, P_n

Mealy Zustandsautomat M (beschreibt Synchronisation)

Ausgaben:

Benutzungsmodell R

Beispiel



Im Beispiel werden die beiden Benutzungsmodelle ROAD und DRIVER durch den Automaten SYNCHRONIZE synchronisiert. Im dargestellten Fall ist es unmöglich zu beschleunigen wenn die Straße bergab geht. Dadurch ist nach den Stimuli „go_downhill“ und „stay_downhill“ kein „accelerate“ möglich. Erst wenn die Straße gerade oder steigend ist, kann das Fahrzeug beschleunigen. In SYNCHRONIZE wird sichergestellt, dass der Stimulus „accelerate“ nur generiert werden kann, wenn die beschriebene Bedingung erfüllt ist.

ab.

2.9 Fehlbenutzung

Name: FAIL_USE

Alternative Bezeichnungen:

Mis-use

Beschreibung:

Hinzufügen von bestimmten Transitionen mit Fehlerstimuli, die in spezielle Fehlerzustände führen.

Superoperatoren:

-

Suboperatoren:

-

Eingabedaten:

Benutzungsmodell P

Fehlerstimuli s

Mutationsrate p

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

-

Eigenschaften:

R ist terminierend.

Keine starke Trace-Erhaltung.

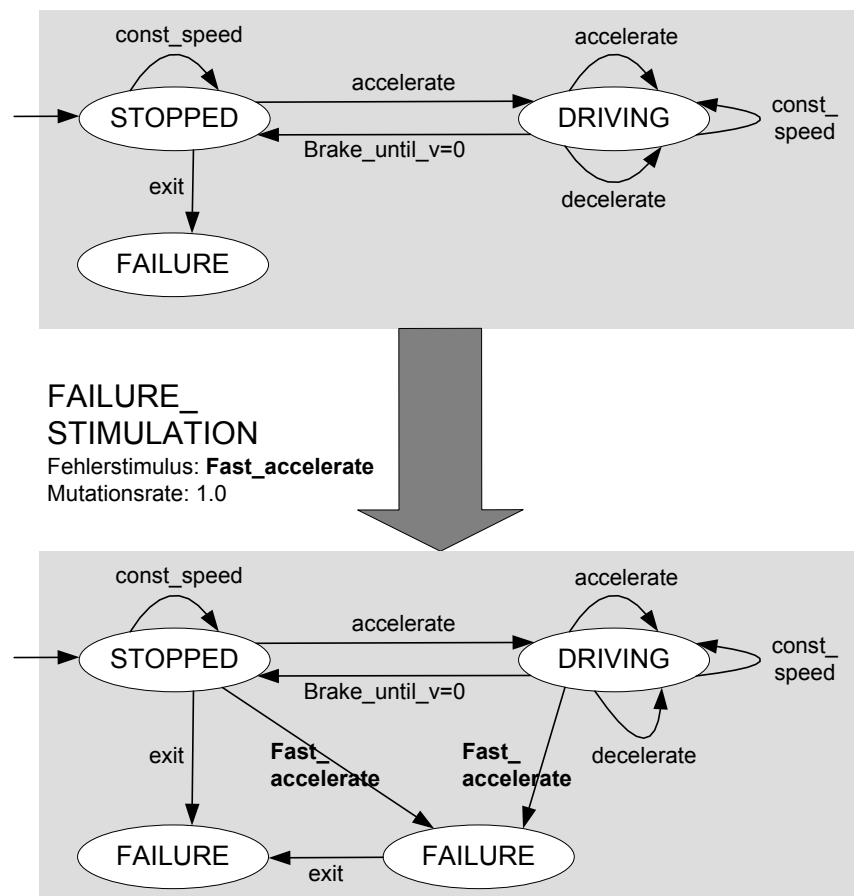
Semantik:

Bei diesem Operator werden zusätzliche Transitionen mit den Fehlerstimuli ins Modell aufgenommen, die alle in den Fehlerzustand führen. Von diesem

Fehlerzustand führt die einzige Kante in den Exit-Zustand. Durch die Mutationsrate wird die Wahrscheinlichkeit der Fehlertransitionen festgelegt.

Spezifikation in π :

Beispiel:



Im Beispiel wird der Fehlerstimulus "fast_accelerate" dem Benutzungsmodell hinzugefügt. Bei der Mutation werden zusätzliche Transitionen ins Modell aufgenommen, die alle in den Fehlerzustand führen. Der Fehlerzustand heißt hier „FAILURE“, er führt in den Endzustand des Benutzungsmodells. Die Mutationsrate im Beispiel ist 1.0, damit mutiert das Benutzungsmodell in jedem Zustand, in unserem Fall in „STOPPED“ und „DRIVING“. Die später daraus generierten Sequenzen stimulieren explizit die Fehlbenutzung, die durch die schnelle Beschleunigung verursacht wird.

2.10 ROBUSTheitstest

Name: ROBUSTNESS

Alternative Bezeichnungen:

-

Beschreibung:

Hinzufügen von Selbsttransitionen mit bestimmtem Stimulus an alle Zustände.

Superoperatoren:

-

Suboperatoren:

-

Eingabedaten:

Benutzungsmodell P

Robustheitsstimulus s

Rate p

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

-

Eigenschaften:

Wenn Modell P terminierend ist, so ist auch Modell R terminierend.

Dieser Operator besitzt die starke Trace-Erhaltung.

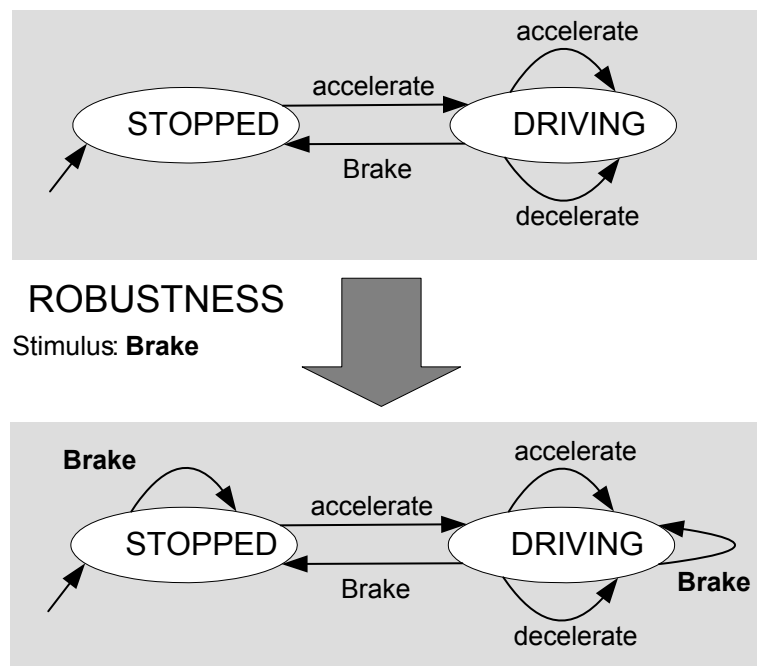
Semantik:

Bei diesem Operator werden zusätzliche Selbsttransitionen mit dem Robustheitsstimulus ins Modell aufgenommen. Diese Transitionen haben eine

Wahrscheinlichkeit von p . Alle anderen Wahrscheinlichkeiten werden angepasst.

Spezifikation in π :

Beispiel:



TODO: Beschreibung

2.11 Terminierend Machen

Name: MAKE_TERMINATING

Alternative Bezeichnungen:

-

Beschreibung:

Hinzufügen von Kanten zu einem Endzustand

Superoperatoren:

-

Suboperatoren:

-

Eingabedaten:

Benutzungsmodell P

Rate p

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

$p > 0$

Eigenschaften:

R ist terminierend.

Keine starke Trace-Erhaltung!

Semantik:

Bei diesem Operator werden zusätzliche Transitionen zu einem Endzustand ins Modell aufgenommen. Diese Transitionen haben eine Wahrscheinlichkeit von p. Alle anderen Wahrscheinlichkeiten werden angepasst.

Spezifikation in π :

Beispiel:

TODO: Beschreibung

2.12 Nicht-terminierend Machen

Name: MAKE_NONTERMINATING

Alternative Bezeichnungen:

-

Beschreibung:

Hinzufügen von Selbstkanten im Endzustand

Superoperatoren:

-

Suboperatoren:

-

Eingabedaten:

Benutzungsmodell P

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

-

Eigenschaften:

R ist nicht-terminierend.

Keine starke Trace-Erhaltung!

Semantik:

Bei diesem Operator wird im Endzustand die END-Transition durch STUTTER ersetzt. Falls P bereits nicht-terminierend ist, tut dieser Operator nichts.

Spezifikation in π :

Beispiel:

TODO: Beschreibung

TODO: Barrier und Suspend einfügen.

2.13 BARRIER

Name: BARRIER

Alternative Bezeichnungen:

-

Beschreibung:

Hinzufügen von Selbstkanten im Endzustand

Superoperatoren:

-

Suboperatoren:

-

Eingabedaten:

Eingabemodelle M_1, \dots, M_n

Menge an Stimuluspaaren $\{s_1 \Rightarrow t_1, \dots, s_n \Rightarrow t_n\}$

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

-

Eigenschaften:

R ist terminierend, wenn mind. ein Eingabemodell terminierend ist

Deadlock-Gefahr, gefahrlos bei einem fairen Mastermodell, dass nur t_i 's erzeugt, aber nicht angehalten wird.

Deadlock-Freiheit impliziert starke Trace-Erhaltung

Semantik:

Führe M_1, \dots, M_n parallel aus. Wenn M_i sendet, halte M_i so lange an, bis ein anderes Modell t_i sendet. Abbruch, wenn alle terminierenden Eingabemodelle END erreicht haben.

Spezifikation in π :

Beispiel:

TODO: Beschreibung

2.14 SUSPEND

Name: SUSPEND

Alternative Bezeichnungen:

-

Beschreibung:

Hinzufügen von Selbstkanten im Endzustand

Superoperatoren:

-

Suboperatoren:

-

Eingabedaten:

Eingabemodelle M_1, \dots, M_n

Stimuli *stop, resume*

Ausgaben:

Benutzungsmodell R

Vor- und Nachbedingungen:

kein Monopol

Eigenschaften:

R ist terminierend, wenn mind. ein Eingabemodell terminierend ist

Monopol-Gefahr,

Monopol-Freiheit impliziert starke Trace-Erhaltung

Semantik:

Führe M_1, \dots, M_n parallel aus. Wenn M_i *stop* sendet, halte alle anderen Modelle so lange an, bis M_i *resume* sendet. Abbruch, wenn alle terminierenden Eingabemodelle END erreicht haben.

Spezifikation in π :

3 Anwendungsbeispiel Editor

Als Anwendungsbeispiel für komposite Modelle soll ein Stimulationsmodell für den systematischen Test eines fiktiven Texteditors verwendet werden. Ein Editor eignet sich dabei besonders für die Verwendung von kompositen Modelle, da zum einen viele verschiedene Funktionalitäten getestet werden müssen und zum anderen in bestimmten Situationen nur bestimmte Eingaben überhaupt möglich sind.

In Abschnitt 3.1 wird zunächst der fiktive Editor, der als Beispiel dienen soll, vorgestellt. In Abschnitt 3.2 werden Stimulationsmodelle für einzelne Komponenten oder Funktionalitäten des Editors beschrieben. Diese einzelnen Stimulationsmodelle werden dann mit Hilfe verschiedener Kompositionsoperatoren zusammengesetzt. Dies ist in Abschnitt 3.3 genauer beschrieben. Dadurch erhält man ein Stimulationsmodell für alle Funktionalitäten des Editors.

3.1 Funktionen des Editors

Bei dem zu testenden Editor handelt es sich um einen gewöhnlichen Texteditor. Zur besseren Anschauung wurde auf viele Funktionalitäten verzichtet.

Folgende Aktionen sind möglich:

Start

Der Editor wird gestartet. Nach dem Start ist zunächst kein Dokument geöffnet. Daher muss erst ein Dokument neu erstellt oder geöffnet werden, bevor ein Editieren möglich wird.

Neues Dokument

Ein leeres Dokument in neuem Fenster wird erstellt.

Dokument Öffnen

Über den File-Browser-Dialog lässt sich eine Datei auswählen. Diese Datei wird in einem neuen Fenster geöffnet. Während des Öffnungsvorgangs sind keine anderen Aktionen möglich. Es ist aber möglich den Vorgang abubrechen.

Dokument speichern

Falls das Dokument schon gespeichert war, kann es wieder unter dem gleichen Dateinamen gespeichert werden. Falls nicht ist diese Aktion nicht möglich.

Dokument speichern unter

Über den File-Browser-Dialog kann eine Datei in einem beliebigen Verzeichnis ausgewählt werden. Als diese Datei wird das Dokument gespeichert. Es ist möglich neue Verzeichnis und eine neue Datei anzulegen. Während dieses Vorgangs sind keine anderen Aktionen möglich.

Programm beenden

Hierfür müssen alle Dokumente geschlossen werden.

Dokument editieren

Falls der Fokus auf dem Dokument liegt, kann es editiert werden.

Dokument schließen

Falls Dokument noch nicht gespeichert ist, wird nachgefragt, ob es gespeichert werden soll. Ggf. wird die Aktion *Speichern unter* ausgeführt.

3.2 Teilmodelle des Editors

Einzelne Komponenten oder Funktionalitäten des Editors können sich durch spezielle Stimulationsmodelle angesprochen werden.

Dokument bearbeiten

Um ein einzelnes Dokument zu bearbeiten, muss entweder ein neues Dokument erstellt werden oder ein bestehendes Dokument geöffnet werden. Das Dokument kann anschließend bearbeitet und gespeichert werden. Ein Schließen des Dokuments ist nur möglich, wenn Änderungen gespeichert wurden.

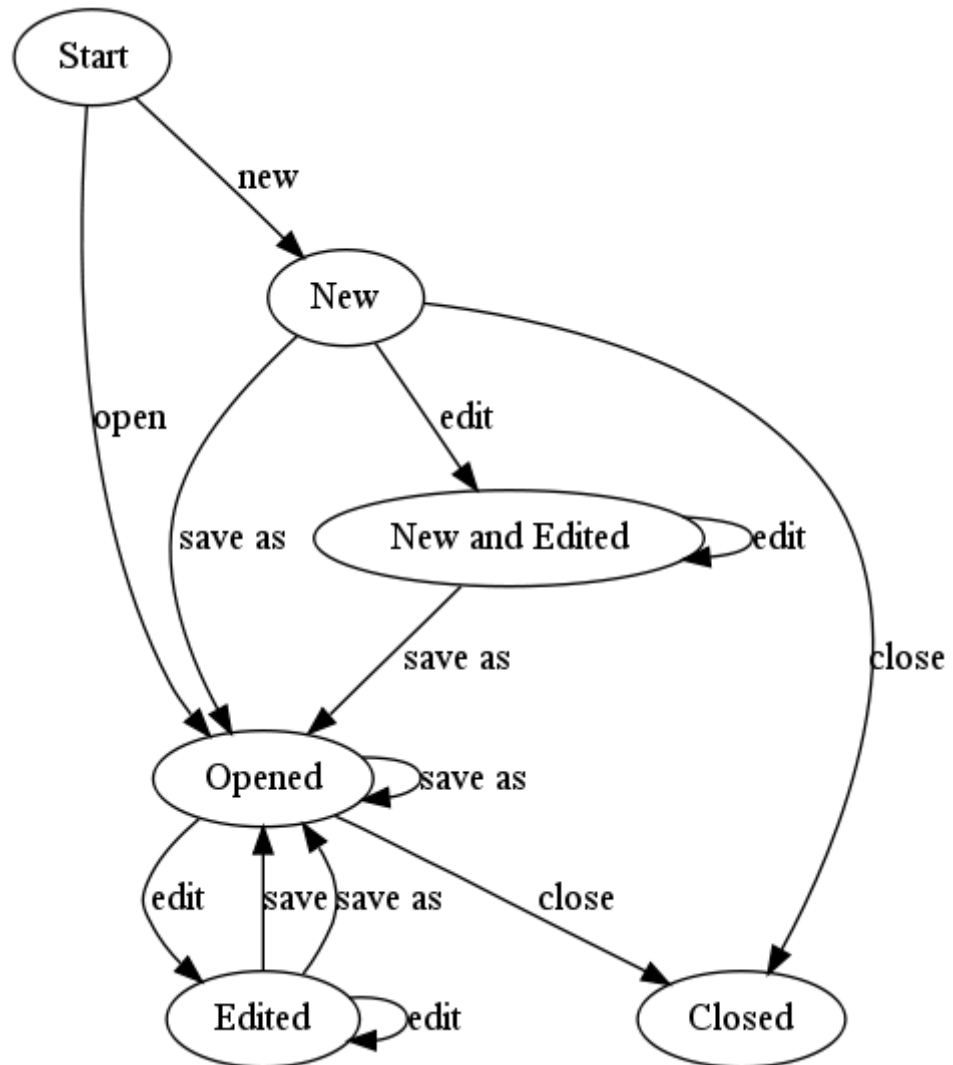


Abbildung 1:

Editieren

Der edit-Stimulus lässt sich weiter verfeinern. So können alle Zeichen eingegeben werden. Außerdem ist es möglich eine bestimmte Anzahl an Änderungen wieder rückgängig zu machen.

Bild

Beenden-Dialog

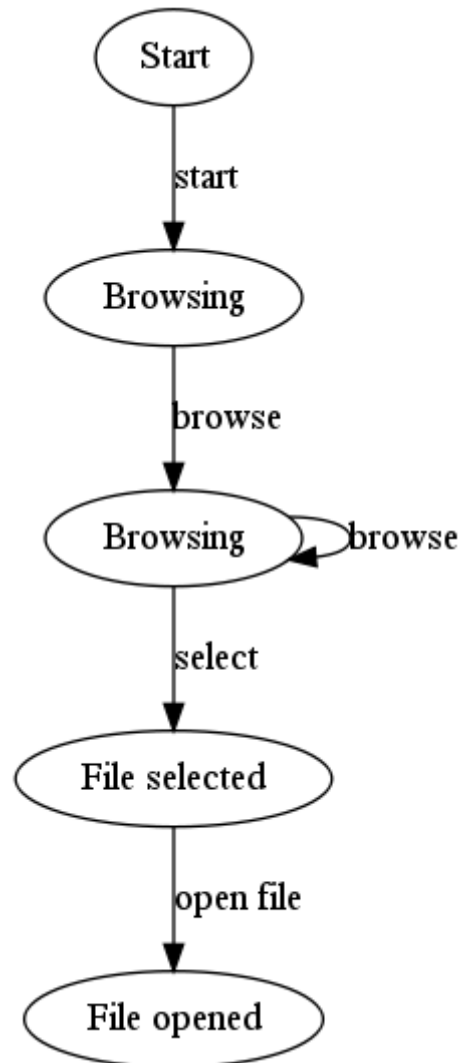
Sobald das Programm beendet wird, wird überprüft, ob editierte Dokumente noch nicht gespeichert sind. Ist dies nicht der Fall wird das Programm beendet. Sind Änderungen noch nicht gespeichert, ist es möglich zwischen Speichern (falls es schon mal gespeichert war), Speichern unter, Abbrechen und Verwerfen zu wählen.

Als einfache Realisierung

Hier lässt sich der BARRIER-Operator verwenden. Erst wenn alle anderen Modelle den save Stimulus

Öffnen-Dialog

Zum Öffnen wird der Filebrowser geöffnet. Dort kann eine Datei zum Öffnen ausgewählt werden. Diese Datei wird geöffnet. Spätere Verfeinerungen: Dateityp überprüfen



Speichern-Dialog

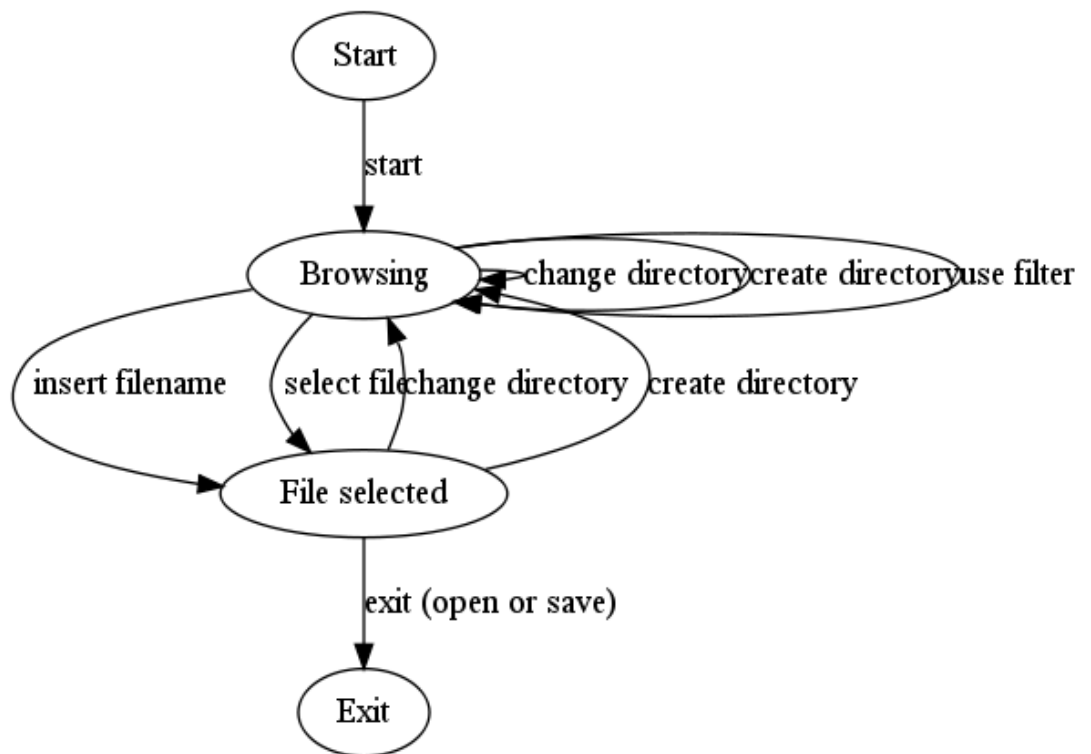
Dokument wird unter aktuellem Dateinamen abgespeichert.

Speichern unter-Dialog

Der Speichern unter-Dialog ähnelt sehr dem Öffnen Dialog. Der Unterschied besteht darin, dass statt **open file**, der Stimulus **save file** erzeugt wird. Auch hier kann das File-Browser-Modell verwendet werden.

File-Browser

Beim Speichern unter einem anderen Namen und beim Öffnen von Dateien wird der Filebrowser verwendet. Wir bauen hier ein Modell, das alle Stimuli für den Filebrowser erzeugen kann. Dabei ist zu beachten, dass während der Filebrowser aktiv ist, keine anderen Eingaben erzeugt werden können. Um dies zu Modellieren verwenden wir später beim Zusammensetzen mit anderen Modellen den SUSPEND-Operator.



Während des Filebrowsings ist es möglich verschiedene

3.2.1 Schließen-Dialog

3.3 Das Komposites Modell

Beschreibung wie man es mit welchen Operatoren zusammensetzt.

Bild des kompositen Modells, Anzahl der Zustände,

Dokumenten Information

Titel: Komposition von Benutzungs-
modellen:
Operatoren und Anwendungs-
beispiele

Datum: Januar 2009
Report: IESE-112.09/D
Status: Final
Klassifikation: Öffentlich

Copyright 2009, Fraunhofer IESE.
Alle Rechte vorbehalten. Diese Veröffentlichung darf
für kommerzielle Zwecke ohne vorherige schriftliche
Erlaubnis des Herausgebers in keiner Weise, auch
nicht auszugsweise, insbesondere elektronisch oder
mechanisch, als Fotokopie oder als Aufnahme oder
sonstwie vervielfältigt, gespeichert oder übertragen
werden. Eine schriftliche Genehmigung ist nicht erfor-
derlich für die Vervielfältigung oder Verteilung der
Veröffentlichung von bzw. an Personen zu privaten
Zwecken.