

Rheinisch-Westfälische Technische Hochschule Aachen

Lehrstuhl Informatik 2
Prof. Dr. Ir. Joost-Pieter Katoen

Zuverlässigkeitsanalyse dynamischer Fehlerbäume

Silvio De Carolis

Masterarbeit
im Fach Informatik

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Aachen, den 26. September 2011

Danksagungen

Zu Beginn bedanke ich mich bei Herrn Prof. Dr. Ir. Joost-Pieter Katoen für die Vergabe und die gute Betreuung dieser Masterarbeit.

Bei meinem Zweitgutachter Herrn AOR Priv.-Doz. Dr. Thomas Noll bedanke ich mich für die Bereitschaft, diese Arbeit zu bewerten.

Mein weiterer Dank gilt sämtlichen Korrekturlesern.

Inhaltsverzeichnis

1. Einleitung	4
2. Strukturen	7
2.1. Dynamische Fehlerbäume	7
2.1.1. Fehlerbäume	7
2.1.2. Dynamische Erweiterungen	8
2.2. Interaktive Markov-Ketten	12
2.2.1. I/O-IMCs	14
3. Tools	16
3.1. CORAL	16
3.1.1. Eingabeformat	17
3.1.2. Übersetzung von DFTs in I/O-IMCs	18
3.1.3. Parallele Komposition	23
3.1.4. Schwache Bisimulation	24
3.1.5. Nichtdeterminismus in DFTs	25
3.2. IMCA	26
3.2.1. Eingabeformat	27
3.2.2. Timed vs. Time-Abstract Scheduler	27
3.2.3. Berechnung der Erreichbarkeit	28
3.3. MRMC	30
3.4. Output CORAL → Input IMCA	30
4. Analyse	33
4.1. Beispiel DFTs	33
4.2. Ausfallwahrscheinlichkeiten der Beispiele	44
4.2.1. Motivation für den Gebrauch von IMCA	46
4.3. Expected Times	52
4.3.1. Maximaler Erwartungswert	53
4.3.2. Minimaler Erwartungswert	54
4.3.3. Testergebnisse für Erwartungswerte	55
5. Nutzen der Kreislosigkeit	57
5.1. Algorithmenanpassungen	57
5.2. Minimierung des Zustandsraumes	57

6. Fazit	61
A. CD mit Quellcode	62
Literaturverzeichnis	63

Übersicht

Mit Fehlerbäumen (FTs) gibt es einen intuitiven Formalismus, um das Fehlverhalten technischer Systeme zu modellieren und zu untersuchen. Dynamische Fehlerbäume (DFTs) erweitern FTs um dynamische Aspekte, wodurch die modellierbare Komplexität der Systeme steigt. Allerdings gibt es Fälle, in denen eine genaue Verhaltensdefinition des DFTs fehlt und so unterschiedliche Interpretationsmöglichkeiten entstehen.

In dieser Masterarbeit wird mit Hilfe des Tools CORAL die Zuverlässigkeitsanalyse von dynamischen Fehlerbäumen (DFTs) untersucht. Die Analyse der DFTs beruht auf einer Erreichbarkeitsanalyse in Interaktiven Markov-Ketten (IMCs). In CORAL wird ein Ansatz verfolgt, in der das zuvor nicht eindeutig definierte Verhalten mit Nichtdeterminismus gelöst wird. Aus diesem Grund benötigt CORAL Tools, welche mit IMCs bzw. mit CTMDPs umgehen können. Aktuell wird dafür der Modelchecker MRMC benutzt. Im Folgenden wird der Unterschied von timed und time-abstract Scheduling anhand von Beispielen deutlich gemacht und untersucht, inwieweit die Nutzung eines Tools, bei dem die Analyse auf timed Scheduling beruht - hier IMCA - für CORAL sinnvoll ist.

1. Einleitung

In der aktuell schon vorhandenen und sich immer weiter entwickelnden hochtechnologischen Zeit wird es immer wichtiger, die Fehlerraten von technischen Systemen zu kennen, sei es aus ökonomischen Gründen, wie der Ausfall einer Produktionsmaschine, oder aus Sicherheitsgründen, wie der Ausfall der Getriebe eines Flugzeugs. Im Zentrum des Interesses stehen also Antworten auf Fragen wie „Mit welcher Wahrscheinlichkeit geht das Auto innerhalb von 10 Jahren kaputt?“, „Ist die Wahrscheinlichkeit eines Ausfalls des Herzschrittmachers innerhalb der nächsten 2,5 Jahre höher als 5%?“ oder auch „Nach welcher Zeit kann ich einen Absturz meines PCs erwarten?“. Eine verbreitete Möglichkeit zur Modellierung des Fehlverhaltens von Systemen bilden Fehlerbäume (FTs). Im Zusammenhang mit den Risiken eines Atomkraftwerks traten sie erstmals um 1981 herum auf [22]. Ausgehend von dem bekannten Ausfallverhalten der Grundelemente eines Systems lässt sich durch die statische Struktur der Bäume effizient das Ausfallverhalten des gesamten Systems ermitteln [6]. Eine Erweiterung um dynamische Aspekte, welche eine höhere Komplexität der zu modellierenden Systeme ermöglicht, wurde mit dynamischen Fehlerbäumen (DFTs) umgesetzt [7]. Mit ihnen ist es möglich, Abhängigkeiten von Grundelementen untereinander und von der Reihenfolge ihres Ausfalls sowie den Einsatz von Ersatzteilen für ausgefallene Elemente zu beschreiben. Durch diese Änderungen steigt allerdings die Zeitkomplexität der Zuverlässigkeitsanalyse im Vergleich zu FTs. Außerdem wurde es in einigen Fällen verpasst, eine genaue Definition des Verhaltens in den DFTs zu geben, wodurch unterschiedliche Interpretationsmöglichkeiten dessen entstehen.

In [5] wird dieses nichtdeterministische Verhalten dadurch ersetzt, dass den unterschiedlichen Verhaltensmöglichkeiten Wahrscheinlichkeiten zugewiesen werden, mit denen sie auftreten. Einen anderen Ansatz verfolgt das Analysetool CORAL [3]. Hier wird das nichtdeterministische Verhalten als solches belassen und in die Analyse eingebaut. Die Zuverlässigkeitsanalyse wird in CORAL dabei in eine Erreichbarkeitsanalyse auf interaktiven Markov-Ketten (IMCs) gewandelt. IMCs sind eine Mischung aus *continuous-time Markov chains* (CTMCs) und *labeled transitions systems* (LTSs), das heißt sie haben interaktive und Markov Transitionen. Während erstere die Kommunikation innerhalb des entsprechenden DFTs bzw. den Ausfall des Systems beschreiben, wird durch die zweiten das Ausfallverhalten umgesetzt. Enthält der zugrundeliegende DFT keinen Nichtdeterminismus, so ist die entstehende IMC in eine CTMC wandelbar. Die Analyse von CTMCs wurde bereits ausgiebig untersucht und es sind effiziente Algorithmen für eine Erreichbarkeitsanalyse auf diesen bekannt [2]. Interessanter für diese Arbeit sind DFTs,

in denen Nichtdeterminismus auftritt und dadurch eine Transformation in einen CTMC gegebenenfalls nicht mehr möglich ist. Hier können, je nach nichtdeterministischer Wahl des folgenden Verhaltens, unterschiedliche Analyseergebnisse auftreten. Als Folge wird unterschieden zwischen *maximaler* und *minimaler* Wahrscheinlichkeit. Die nichtdeterministische Wahl wird durch Scheduler getroffen. Bei diesen gibt es die relevante Unterscheidung zwischen *time-abstract* und *timed* Schemulern. Den Letztgenannten stehen sowohl Informationem über die History als auch über die vergangene Zeit zur Verfügung. Time-abstract Scheduler haben im Gegensatz dazu nur Kenntnis über die History. In vielen Fällen reicht eine solche um das optimale Ergebnis zu ermitteln. Es gibt jedoch auch Fälle, in denen die Information über die Zeit für das Finden des optimalen Werts erforderlich ist.

In der aktuellen Version von CORAL (April 2011) wird, im Falle von auftretendem Nichtdeterminismus, eine Umwandlung der IMC in eine *continuous-time Markov Chain* (CTMDPs) vollzogen, welche wiederum mit dem Model Checker MRMC [25] auf die Erreichbarkeit eines passenden Zielzustands hin analysiert wird. MRMC arbeitet bei der Analyse mit time-abstract Schemulern.

Der Kernpunkt dieser Arbeit ist die Untersuchung, ob es sich in diesem Zusammenhang lohnt, MRMC durch IMCA [30], welches die Analyse von IMCs basierend auf timed Schemulern durchführt, zu ersetzen. Dafür wurde zunächst eine Schnittstelle in IMCA zur Anbindung an CORAL geschaffen, wodurch ein direkter Vergleich der Ergebnisse CORALs unter der Benutzung von MRMC bzw. von IMCA ermöglicht wird. Dieser wurde dann anhand mehrerer konstruierter Beispiele durchgeführt und interpretiert.

Die Masterarbeit setzt sich dabei aus sechs Kapiteln wie folgt zusammen:

- Kapitel 1 gibt eine Einleitung in das Thema dieser Arbeit und gibt einen groben Überblick über diese wieder.
- In Kapitel 2 werden dynamische Fehlerbäume und interaktive Markov-Ketten definiert und näher vorgestellt sowie auf wichtige Eigenschaften dieser eingegangen.
- Das dritte Kapitel stellt CORAL und dessen Arbeitsweise vor. Zusätzlich gibt es eine kurze Einführung in für diese Arbeit relevante Teilaspekte der Tools MRMC und IMCA. Zum Schluß des Kapitels wird auf die Einbindung IMCAs in CORAL eingegangen.
- Kapitel 4 stellt einige Beispiele für DFTs vor, die Nichtdeterminismus enthalten. Diese Beispiele wurden analysiert und die Ergebnisse verglichen. Insbesondere wird ein DFT zur Verdeutlichung des Unterschieds zwischen MRMC und IMCA analysiert. Auf diesen Ergebnissen aufbauend soll die Frage geklärt werden, ob sich eine Ersetzung MRMCs durch IMCA für CORAL lohnt. Des Weiteren wird noch auf

die Berechnung der *expected Time* von IMCs in IMCA eingegangen.

- Kapitel 5 stellt die Idee aus [16] für eine mögliche stärkere Zustandsraumminimierung in CORAL vor.
- Abgeschlossen wird die Arbeit durch ein Fazit in Kapitel 6.
- Im Anhang ist eine CD mit dem Code zum Übergang von CORAL zu IMCA zu finden.

2. Strukturen

In diesem Kapitel werden in Abschnitt 2.1 dynamische Fehlerbäume (DFTs) und in Abschnitt 2.2 interaktive Markov-Ketten (IMCs) eingeführt. DFTs stellen die Eingabe von CORAL dar und sind die Struktur, welche es in dieser Arbeit zu untersuchen gilt. Dabei werden die DFTs in Input/Output-IMCs (I/O-IMCs) gewandelt, welche wiederum in allgemeine IMCs übersetzt werden. Die Analyse erfolgt dann mit IMCA auf IMCs bzw. mit MRMC auf CTMDPs (*continuous-time Markov decision processes*) oder auf CTMCs (*continuous-time Markov chains*). Mehr über die Tools und die Umwandlung von DFTs in IMCs ist in Kapitel 3 zu finden.

2.1. Dynamische Fehlerbäume

Fehlerbäume (FTs) [22] sind eine verbreitete Möglichkeit, das Fehlverhalten von einfachen bis hin zu komplexen technischen Systemen zu modellieren. Durch ihre statische Struktur lässt sich die Ausfallwahrscheinlichkeit des Systems effizient berechnen [6]. Dies ist bei der Erweiterung zu dynamischen Fehlerbäumen nicht der Fall. Hier steigt die Abhängigkeit der Untersysteme voneinander, wodurch die Analyse aufwendiger wird.

2.1.1. Fehlerbäume

Fehlerbäume sind, wie der Name schon sagt, eine Art von endlichen Bäumen. Dabei ist es den Knoten erlaubt, mehrere Elternknoten zu haben, wobei zum Beispiel auch ein Geschwisterknoten als Elternknoten auftreten kann. Die Blätter werden dabei als Basisevents (BE) bezeichnet. Sie können mit einer bestimmten Wahrscheinlichkeit ausfallen. Diese Wahrscheinlichkeit kann dabei entweder als Exponentialverteilung oder als Phasentypverteilung angegeben werden. Eine Phasentypverteilung hat endlich viele zu durchlaufende Stufen, in denen sich die Ausfallwahrscheinlichkeiten unterscheiden können. Ein BE kann entweder *aktiv* oder *ausgefallen* sein. Dabei ist nur ein Wechsel von „aktiv“ nach „ausgefallen“ möglich. Die Basisevents können dann durch verschiedene Gates zu einem System verbunden werden. Auch Gates selber können wiederum Eingangselemente für andere Gates sein. Gates haben dabei endlich viele Inputs und, außer der Wurzel, mindestens einen Output. Die bei FTs zur Verfügung stehenden Gates sind dabei

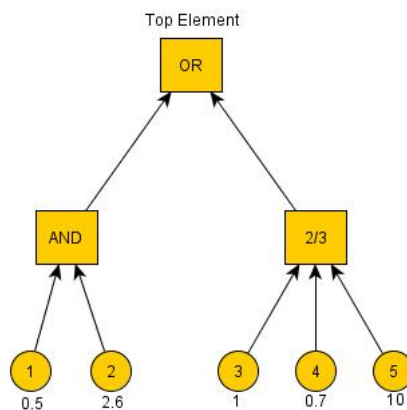
OR-Gate das OR-Gate fällt aus, wenn eines seiner Inputs ausfällt,

AND-Gate das AND-Gate fällt aus, wenn alle seiner Inputs ausfallen,

K/M VOTING-Gate das K/M VOTING-Gate fällt aus, wenn mindestens K seiner M Inputs ausfallen.

Die Wurzel des Baums wird als *Top Element* bezeichnet, dessen Ausfall mit dem gesamten Systemausfall gleichzusetzen ist. Die Analyse von Fehlerbäumen ist effektiv durchführbar, nachzulesen zum Beispiel in [21] und [6]. Ein kleines Beispiel für die graphische Darstellung von Fehlerbäumen ist in Abbildung 2.1 zu sehen. Dabei sind die BEs als Kreise dargestellt und die entsprechenden Ausfallraten λ stehen darunter. Sie sind dabei als exponentialverteilt aufzufassen, sodass sich die Ausfallwahrscheinlichkeit eines BEs aus $1 - e^{-\lambda t}$ ergibt, wobei t die vergangene Zeit darstellt. Bei phasentypverteilten BEs müssten mehrere Raten für die unterschiedlichen Phasen angegeben werden.

Abbildung 2.1.: Beispiel für einen FT



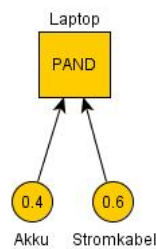
2.1.2. Dynamische Erweiterungen

Dynamische Fehlerbäume ergänzen Fehlerbäume um dynamische Gates, durch welche weitere Abhängigkeiten von Untersystemen oder BEs beschrieben werden können. Dabei handelt es sich um die folgenden Gates.

PAND-Gate Das *priority AND*-Gate fällt aus, sobald alle Inputs in der angegebenen Reihenfolge ausfallen. Ein Beispiel für ein solches Gate liefert Abbildung 2.2. Man stelle sich vor, man arbeite an seinem Laptop an einer Ausarbeitung und hat sowohl

ein volles Akku als auch das Stromkabel angeschlossen. Nun kann es passieren, dass das Stromkabel aus irgendwelchen Gründen kaputt geht. Dann sollte der Laptop automatisch auf Akkubetrieb umschalten und eine entsprechende Meldung ausgeben, sodass man seine Arbeit speichern und sich ein neues Stromkabel besorgen kann. Wenn jedoch zuerst der Akku kaputt geht, man dies, da der Laptop per Strom arbeitet, nicht bemerkt und dann zusätzlich das Stromkabel kaputt geht, schaltet sich der Laptop ab und die Arbeit ist verloren.

Abbildung 2.2.: PAND-Gate

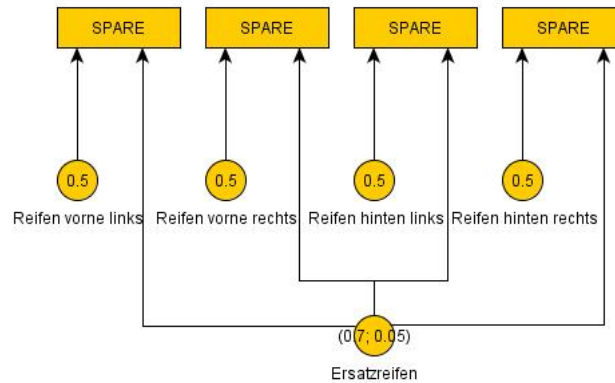


SPARE-Gate Das SPARE-Gate bekommt als Eingabe ein primäres Element und optional weitere Ersatzelemente. Es fällt aus, sobald das primäre Element und alle Ersatzelemente ausfallen bzw. kein Ersatz mehr zur Verfügung steht. Als Beispiel kann man sich ein Auto vorstellen, welches vier Reifen im Einsatz und einen Ersatzreifen im Kofferraum hat. In Abbildung 2.3 ist die Situation der Reifen modelliert. Den vier SPARE-Gates mit jeweils einem Reifen als primärem Element, also dem Hauptreifen, steht insgesamt ein Ersatzelement, der Ersatzreifen, zur Verfügung. Es ist jedoch klar, dass der Ersatzreifen nur einen Hauptreifen ersetzen kann, und nicht mehrere. Dies gilt auch allgemein. Es ist Ersatzelementen möglich, bei mehreren unterschiedlichen SPARE-Gates als Ersatzteil eingetragen zu sein. Tatsächlich kann ein Element jedoch nur in einem SPARE-Gate aktiv sein. Dabei steht das Element jenem Gate zur Verfügung, welches es zuerst benötigt. Die vorstellbare Situation, dass das primäre Element eines SPARE-Gates zugleich ein Ersatzelement eines anderen SPARE-Gates ist, ist in DFTs ausgeschlossen. Anders gesagt heißt das, dass die Mengen der primären Elemente und der Ersatzelemente disjunkt sind.

Für das SPARE-Gate wird ein weiterer Zustand für Basisevents eingeführt. Ein Element, welches als Ersatzelement auftritt, ist solange *schlafend*, bis es benötigt wird. Dabei spielt wieder die Reihenfolge der Ersatzelemente eine Rolle, da sie gemäß ihrer Reihenfolge als Ersatz herangezogen werden. Ist ein BE schlafend, so wird die Ausfallrate um einen Ruhefaktor α verringert, sodass sich die Gesamtausfallrate aus $\mu = \alpha\lambda$ ergibt. Den Sinn dahinter kann man sich wieder an Abbildung 2.3 bewusst machen. Solange der Ersatzreifen nicht im Einsatz ist und nur im Kofferraum liegt, ist die Wahrscheinlichkeit eines Ausfalls dessen erheblich gerin-

ger, als wenn er sich im Einsatz befindet. Mit α , hier 0.05, wird also der Faktor beschrieben, um den sich die *aktive* Ausfallrate, hier 0.7, im *schlafenden* Zustand verringert.

Abbildung 2.3.: SPARE-Gate



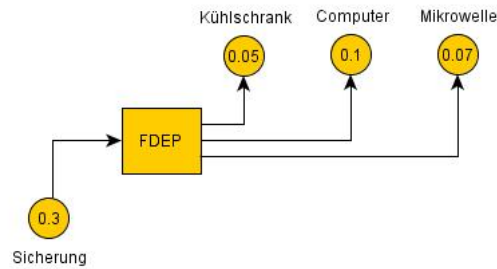
Eingänge für ein SPARE-Gate können sowohl BEs als auch Untersysteme sein. Im letzteren Fall sind dann die zugrunde liegenden BEs solange schlafend, bis sie gebraucht werden (Besonderheit, falls ein SPARE-Gate ein Ersatzelement eines anderen SPARE-Gates ist, siehe 3.1.2).

FDEP-Gate Das *functional dependency*-Gate unterscheidet sich von den anderen Gates, da es im ursprünglichen Sinne keinen Ausfall an ein übergeordnetes System meldet. Als Eingabe hat es einen Trigger und endlich viele von diesem Trigger abhängige Elemente. Fällt der Trigger aus, so fallen auch, ohne zeitliche Verzögerung, die abhängigen Elemente aus. Je nach Definition müssen die abhängigen Elemente BEs (zum Beispiel [7] und [23]), oder können, wie bei CORAL, zusätzlich auch Untersysteme sein. Da formal jedes Gate einen Ausgang haben muss, hat auch das FDEP-Gate einen solchen, welcher jedoch in der späteren Berechnung der Systemausfallwahrscheinlichkeit nicht berücksichtigt wird.

In Abbildung 2.4 ist als Beispiel der Teil eines Haushalts modelliert. Die Basisevents stellen einen Kühlschrank, einen Computer und eine Mikrowelle dar. Zusätzlich gibt es die Sicherung, welche als Trigger dient. Fliegt die Sicherung heraus, so erhalten die drei Geräte keinen Strom mehr und schalten sich sofort ab.

Die Analyse von DFTs ist im Gegensatz zu FTs aufwendig und beim Analysieren tritt das bekannte Problem der *state space explosion* auf. Genauer zur Analyse von DFTs wird in Kapitel 3 behandelt. Ein weiterer wichtiger Punkt bei DFTs ist das mögliche Auftreten von Situationen, in denen das Vorgehen nicht klar definiert ist und somit unterschiedlich

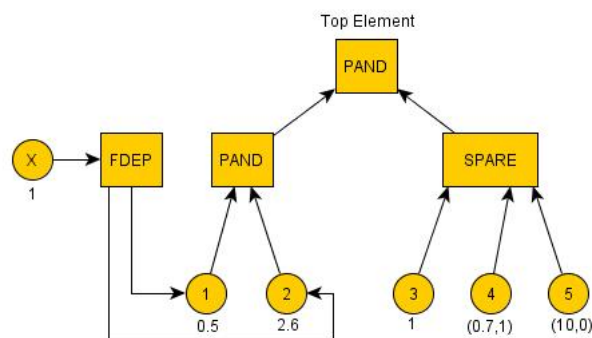
Abbildung 2.4.: FDEP-Gate



behandelbar ist. In CORAL wird dies als nichtdeterministisches Verhalten interpretiert. Näheres dazu ist in Kapitel 3.1.5 zu finden.

Abbildung 2.5 zeigt einen DFT als Beispiel für die dynamischen Gates. Dabei ist unter den BEs mit λ bzw. (λ, α) die Ausfallrate λ und, falls es sich um ein mögliches Ersatzteil handelt, der Ruhefaktor α angegeben. Um beim FDEP-Gate deutlicher zu machen, welches Element der Trigger und welche die abhängigen Elemente sind, geht der Pfeil vom Trigger zum Gate, und vom Gate zu den abhängigen Elementen. Der Pfeil für den Output wird aus Gründen der Übersicht für FDEP-Gates gespart. Bei den Ersatzteilen 4 und 5 sind zwei Extremfälle für den Ruhefaktor zu sehen. So unterscheidet sich die Ausfallrate von BE 4 im aktiven oder schlafenden Zustand nicht, während BE 5 im Ruhezustand nicht ausfallen kann. Weitere Beispiele sind in Kapitel 4 zu finden.

Abbildung 2.5.: Beispiel für einen DFT



2.2. Interaktive Markov-Ketten

Interaktive Markov-Ketten (IMCs) sind eine Kombination markierter Transitionssysteme (LTSs, *labeled transition systems*) und CTMCs. Ihre Transitionen repräsentieren entweder interaktive Aktionen oder eine exponential- oder phasenverteilte Verzögerung. Der letzte Fall beschreibt dabei eine Markov Transition, wo die Wahrscheinlichkeit, dass diese Transition vollzogen wird, von der vergangenen Zeit abhängig ist.

Formal ist eine IMC ein Fünftupel $M = (S, Act, \longrightarrow, \Longrightarrow, s_0)$. Dabei beschreibt S eine nichtleere, endliche Menge von Zuständen, Act eine nichtleere, endliche Menge von Aktionen, $\longrightarrow \subseteq S \times Act \times S$ eine endliche Menge von interaktiven Transitionen, $\Longrightarrow \subseteq S \times \mathbb{R}_{>0} \times S$ eine endliche Menge von Markov Transitionen und s_0 den Startzustand.

Die Aktionsmenge Act kann dabei noch in Act_e für externe beobachtbare Aktionen und Act_i für interne Aktionen unterteilt werden. Dies ist zum Beispiel relevant, wenn man mehrere IMCs synchronisieren möchte. Interne Aktionen werden dabei in dieser Arbeit generell mit i benannt.

Statt $(s, a, s') \in \longrightarrow$ mit $s, s' \in S$ und $a \in Act$ wird $s \xrightarrow{a} s'$ geschrieben. Analog dazu wird statt $(s, \lambda, s') \in \Longrightarrow$ mit $s, s' \in S$ und $\lambda \in \mathbb{R}_{>0}$ $s \xrightarrow{\lambda} s'$ verwendet. Dabei wird λ als *Rate* bezeichnet.

Weiterhin werden $IT(s) = \{s \xrightarrow{a} s'\}$ und $MT(s) = \{s \xrightarrow{\lambda} s'\}$ als Mengen der von s ausgehenden interaktiven bzw. Markov Transitionen definiert. Betrachtet man nun eine IMC M mit $MT(s) = \emptyset$ für alle Zustände $s \in S$, so hat man ein normales LTS. Entsprechend hat man für $IT(s) = \emptyset$ eine normale CTMC. Daraus wird ersichtlich, dass IMCs sowohl eine Erweiterung von LTSs als auch von CTMCs sind.

Während die interaktiven Transitionen selbsterklärend sein dürften, bedarf es für die Markov Transitionen eventuell einer genaueren Erläuterung: Eine Transition $s \xrightarrow{\lambda} s'$ ist so zu verstehen, dass die IMC innerhalb von t Zeiteinheiten mit einer Wahrscheinlichkeit von $1 - e^{-\lambda \cdot t}$ von Zustand s nach Zustand s' wechseln kann. Angenommen es gibt nun $s \xrightarrow{\lambda'} s'$ und $s \xrightarrow{\lambda''} s''$, mit $s' \neq s''$, und s hat keine interaktive Transition. Dann tritt die sogenannte *race condition* auf, die ausgehenden Markov Transitionen konkurrieren also miteinander. Sei jetzt $\mathbf{R}(s, s') = \Sigma\{\lambda | s \xrightarrow{\lambda} s'\}$ als die Summe aller Raten von s nach s' und $E(s) = \Sigma_{s' \in S} \mathbf{R}(s, s')$ als die Summe aller von s ausgehenden Raten definiert. Dann ist die Wahrscheinlichkeit, von s zu einem bestimmten Zustand s' innerhalb von t Zeiteinheiten zu gelangen

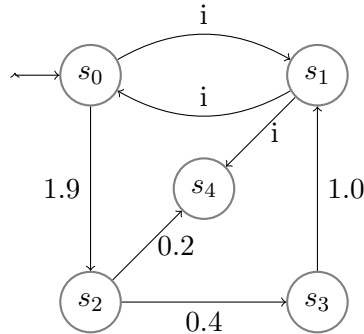
$$\frac{\mathbf{R}(s, s')}{E(s)} \cdot (1 - e^{-E(s) \cdot t}).$$

Der erste Teil, also $\frac{\mathbf{R}(s,s')}{E(s)}$, beschreibt dabei die Wahrscheinlichkeit, dass eine Transition von s nach s' , und nicht etwa nach s'' gewählt wird. Der zweite Teil, also $1 - e^{-E(s) \cdot t}$, beschreibt die Wahrscheinlichkeit, dass überhaupt eine Transition gewählt wird. Näheres zu der Definition ist in [13] und [24] zu finden.

Als Beispiel ist in Abbildung 2.6 eine IMC zu sehen. Dabei sind die Transitionen mit Zahlen als Beschriftung Markov Transitionen und jene mit der Beschriftung „i“ interne Transitionen. Zustand s_1 ist ein *interaktiver Zustand*, da er nur interaktive Transitionen von sich ausgehend hat. Dementsprechend sind die Zustände s_2 und s_3 *Markov Zustände*, Zustand s_0 ein *Hybridzustand* und Zustand s_4 ein *Deadlock*. Markov Zustände verhalten sich wie Zustände in CTMCs, also wie zuvor beschrieben. So wird die Transition $s_3 \xrightarrow{1.0} s_1$ beispielsweise innerhalb von 2 Zeiteinheiten, angefangen zu zählen bei Erreichen von s_3 , mit einer Wahrscheinlichkeit von $1 - e^{-2} \approx 0.865$ vollzogen. Zustand s_2 hat zwei ausgehende Markov Transitionen. Es findet also ein Rennen zwischen den Transitionen statt. Die Wahrscheinlichkeit, dass $s_2 \xrightarrow{0.2} s_4$ innerhalb von 3 Zeiteinheiten gewählt wird, ist $\frac{0.2}{0.6} \cdot (1 - e^{-1.8})$, während sie sich für $s_2 \xrightarrow{0.4} s_3$ aus $\frac{0.4}{0.6} \cdot (1 - e^{-1.8})$ ergibt. Im interaktiven Zustand s_1 findet eine nichtdeterministische Wahl statt, da durch Ausführen einer Aktion zwei verschiedene Folgezustände erreichbar sind. Dies bewirkt, dass die Erreichbarkeitswahrscheinlichkeit, also die Wahrscheinlichkeit, einen bestimmten Zielzustand zu erreichen, im Gegensatz zu den Ergebnissen bei CTMCs nicht eindeutig ist, sondern es einen minimalen und einen maximalen Wert gibt, wodurch die Berechnung schwieriger wird. Mehr zu der Berechnung wird in Kapitel 3.2.3 behandelt.

Unbetrachtet blieb bislang der Startzustand s_0 . Als Hybridzustand kann man im Prinzip auch hier die *race condition* gültig machen. Dabei wird angenommen, dass die interne interaktive Aktion ohne Zeitverlust sofort ausgeführt wird, während die Wahrscheinlichkeit, dass die Markov Transition innerhalb von 0 Sekunden ausgeführt wird, genau 0 beträgt. Es wird also die interne interaktive Aktion gewählt. Diese Eigenschaft wird als *maximal progress assumption* bzw. *maximale Fortschrittsprämisse* beschrieben und sie bewirkt, dass Markov Transitionen in hybriden Zuständen, bei welchen die interaktiven Aktionen intern sind, nie gewählt werden und deshalb der Einfachheit halber gestrichen werden können. Bei externen Aktionen ist dies nicht zwingend der Fall, da sie Verzögerungen ausgesetzt sein können. Da jedoch die im folgenden auftretenden IMCs geschlossen sind, also keiner Synchronisation mehr unterworfen werden, können alle externen Aktionen versteckt und so zu internen Aktionen umgewandelt werden. Wichtige Eigenschaften von IMCs sind die Möglichkeit der parallelen Komposition und der Minimierung mittels schwacher Bisimulation (siehe Kapitel 3.1.4 und [3][13][24][12]), wodurch der Zustandsraum erheblich reduziert werden kann.

Abbildung 2.6.: Beispiel für eine IMC

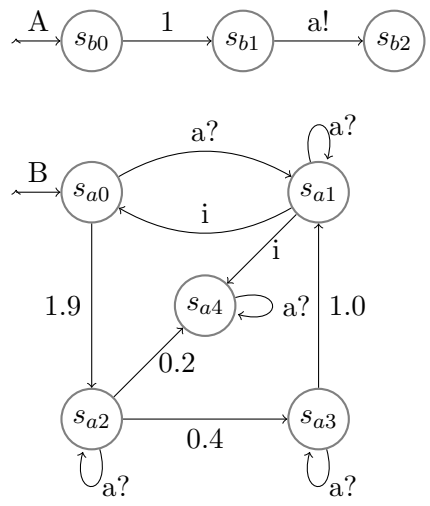


2.2.1. I/O-IMCs

Eine Erweiterung von IMCs sind sogenannte Input/Output-IMCs, kurz I/O-IMCs, bei welchen die Menge der externen Aktionen Act_e nochmal in eine Menge Act_{in} von *input Aktionen* und eine Menge Act_{out} von *output Aktionen* unterteilt wird. Input Aktionen sind durch ein Fragezeichen hinter der Aktion gekennzeichnet, zum Beispiel $a?$. Input Aktionen können nur durchgeführt werden, wenn eine andere I/O-IMC eine entsprechende Output Aktion, zum Beispiel $a!$, liefert. Das heißt, hybride Zustände mit einer Transition mit Input Aktion und einer Markov Transition, lösen ein Rennen zwischen diesen Transitionen aus. Wird der Input geliefert, bevor die Markov Transition ausgeführt wurde, gewinnt die interaktive Transition, im anderen Fall die Markov Transition. Im Gegensatz dazu werden Output Aktionen analog zu internen Aktionen sofort ausgeführt und gewinnen jedes Rennen gegen Markov Transitionen. I/O-IMCs sind *input-enabled*. Das heißt, der IMC muss jederzeit auf jedes seiner möglichen Inputs $a? \in Act_{in}$ reagieren können. Formal ausgedrückt heißt dies: $\forall s \in S$ und $\forall a? \in Act_{in}$ gilt $\exists s' \in S$ mit $s \xrightarrow{a?} s'$.

In Abbildung 2.7 sind zwei Beispiele für I/O-IMCs zu sehen. Die I/O-IMC A besteht dabei aus drei Zuständen. Nach einer Verzögerung mit Rate 1 wird die Output Aktion $a!$ durchgeführt. B besteht aus 5 Zuständen und ist ähnlich zu Abbildung 2.6 aufgebaut. Ein Unterschied ist in Zustand s_{a0} zu sehen. Hier findet jetzt ein Rennen zwischen einer Markov Transition und einer interaktiven Transition mit Input Aktion statt. Gibt A $a!$ zurück bevor die Markov Transition genommen wurde, wird zu Zustand s_{a1} übergegangen. Ein weiterer Unterschied sind die Selfloops in den restlichen Zuständen. Durch sie wird gewährleistet, dass die I/O-IMC input-enabled ist. Im Folgenden werden diese Selfloops übersichtshalber weggelassen.

Abbildung 2.7.: Beispiel für eine I/O-IMC



3. Tools

In diesem Kapitel werden die drei Tools *CORAL*, *MRMC* und *IMCA* vorgestellt. *CORAL* (Kapitel 3.1) ist dabei ein Analysetool für dynamische Fehlerbäume, welches diese in I/O-IMCs übersetzt, welche dann wiederum in das richtige Eingabeformat für *MRMC* (Kapitel 3.3) gewandelt und dort analysiert werden. *MRMC* ist dabei ein Tool zur Analyse von CTMCs, CTMDPs und anderen Strukturen. Im Gegensatz zu *MRMC* ist *IMCA* (Kapitel 3.2) direkt auf die Erreichbarkeitsanalyse von IMCs ausgerichtet. Dafür werden aktuelle Algorithmen zur Berechnung herangezogen (Kapitel 3.2.3). Da in dieser Arbeit untersucht werden soll, inwieweit eine Nutzung IMCAs durch *CORAL* sinnvoll ist, *CORAL* jedoch I/O-IMCs liefert, während *IMCA* IMCs in einem bestimmten Format erwartet, ist eine Umwandlung nötig, auf welche in Kapitel 3.4 eingegangen wird.

3.1. CORAL

CORAL [3] ist ein noch unveröffentlichtes Tool zur Zuverlässigkeitsanalyse von dynamischen Fehlerbäumen. Es soll also untersucht werden, mit welcher Wahrscheinlichkeit ein System, modelliert als DFT, innerhalb einer bestimmten Zeit ausfällt. Als Eingabe erhält *CORAL* dabei (nach Stand April 2011) einen DFT im Galileo Format (Abschnitt 3.1.1), übersetzt diesen erst in mehrere I/O-IMCs (Abschnitt 3.1.2), welche dann durch parallele Komposition (Abschnitt 3.1.3) zu einem zusammengefasst werden und ermittelt an diesem die Ausfallwahrscheinlichkeit des Systems. Dabei wird zum einen das Toolset CADP [26][9] für die Umwandlung und Analyse für CTMCs und zum anderen *MRMC* [25][14] für die Analyse von CTMDPs eingebunden. Genauer stellt sich das Ganze dann wie folgt dar:

1. Übersetzung des dynamischen Fehlerbaums, bzw. seiner einzelnen Elemente, in mehrere I/O-IMCs.
2. Wahl zweier dieser I/O-IMCs und Durchführung einer parallelen Komposition.
3. Wandlung von nicht mehr zur Synchronisation benötigten Output Aktionen in interne Aktionen.

4. Minimierung des in Schritt 2 entstandenen I/O-IMCs mittels schwacher Bisimulation.
5. Wahl des weiteren Vorgehens: Falls nur noch eine I/O-IMC übrig ist, weiter zu Schritt 6. Andernfalls zurück zu Schritt 2.
6. Falls das Produkt ein CTMDP ist: Ergänzung der Zustände um eine Markov Self-loop, sodass alle Zustände dieselbe Gesamtausgangsrate haben. Das Ergebnis ist ein uniformierter CTMDP. Es folgt die Analyse.

Während Schritt 1 in Abschnitt 3.1.2 näher erläutert wird, sind besonders Schritt 2 und die Schritte 4-6 interessant. Es fällt auf, dass nach jeder parallelen Komposition zweier I/O-IMCs die resultierende I/O-IMC minimiert wird, anstatt zunächst die gesamte Komposition aller einzelnen I/O-IMCs zu bilden und das Ergebnis danach zu minimieren. Dadurch wird die maximale Anzahl der während der Komposition vorkommenden Zustände verringert, wobei die in Schritt 2 stattfindende Wahl der zwei I/O-IMCs einen wichtigen Einfluss auf diese Größe hat (auf den minimierten I/O-IMC, der in Schritt 6 schließlich analysiert wird, jedoch nicht, da hier kein Unterschied besteht, wann minimiert wird). Die Minimierung der I/O-IMCs basiert auf schwacher Bisimulation, welche in Abschnitt 3.1.4 näher erklärt wird. Nach der letzten Komposition ist dann entweder ein CTMC oder, falls im DFT Nichtdeterminismus auftrat, ein CTMDP vorhanden. In Schritt 6 werden diesem dann Markovian Selfloops hinzugefügt, sodass ein uniformierter CTMDP entsteht. Dies hat enorme Auswirkungen auf die Analyseergebnisse. Nähere dazu wird in Kapitel 4.2.1 behandelt. In welchen Fällen nichtdeterministisches Verhalten in DFTs auftritt, ist in Abschnitt 3.1.5 näher beschrieben.

3.1.1. Eingabeformat

Als Eingabe bekommt CORAL aktuell den zu untersuchenden DFT im Galileo Format sowie die Zeitpunkte, für welche die Analyse ausgeführt werden soll. Während letzteres direkt in der Shell durch aneinanderreihen der Zeitpunkte angegeben wird, muss der DFT in einer .dft Datei gespeichert sein. Als Beispiel für die Beschreibung eines DFTs kann

```

toplevel "A";
"A" or "B" "C";

"B" fdep "X" "D" "E";

"C" pand "D" "E";

```

```

"X" lambda=0.5 dorm=0;
"D" lambda=0.4 dorm=0;
"E" lambda=0.3 dorm=0;

```

dienen, welches den DFT in Abbildung 4.1 beschreibt. Mit „toplevel“ wird dabei die Wurzel beschrieben und hier mit „A“ benannt. In der nächsten Zeile wird dieses dann als OR-Gate mit den zwei Eingängen „B“ und „C“ definiert. „B“ wird wiederum als FDEP-Gate mit dem Trigger X und den abhängigen BEs D und E beschrieben und „C“ als PAND-Gate mit eben jenen BEs als Eingänge. Die letzten drei Zeilen geben die Ausfallwahrscheinlichkeit („lambda“) und den Ruhefaktor („dorm“) der BEs an.

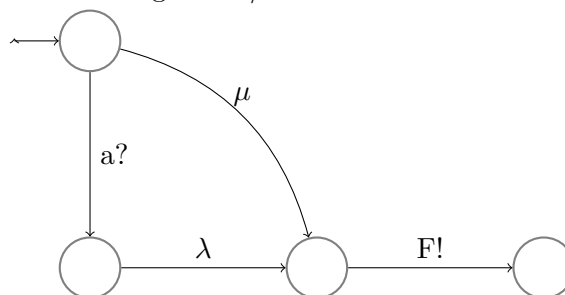
Die in diesem Beispiel nicht vorkommenden AND-, VOTING- und SPARE-Gates werden entsprechend mit „and“, „KofM“ und „csp“ beschrieben. Bei Letzterem steht an erster Stelle das Primärelement gefolgt von den Ersatzelementen, welche in der vorgesehenen Reihenfolge aufgelistet werden.

3.1.2. Übersetzung von DFTs in I/O-IMCs

Wie bereits erwähnt, ist die Umwandlung eines DFTs in eine I/O-IMC ein wichtiger Schritt. Im Folgenden wird für jedes mögliche Element eines DFTs die Struktur des zugehörigen I/O-IMCs angegeben und erläutert. Die folgenden Grafiken sind [3] entnommen.

Basisevents

Abbildung 3.1.: I/O-IMC für Basisevents



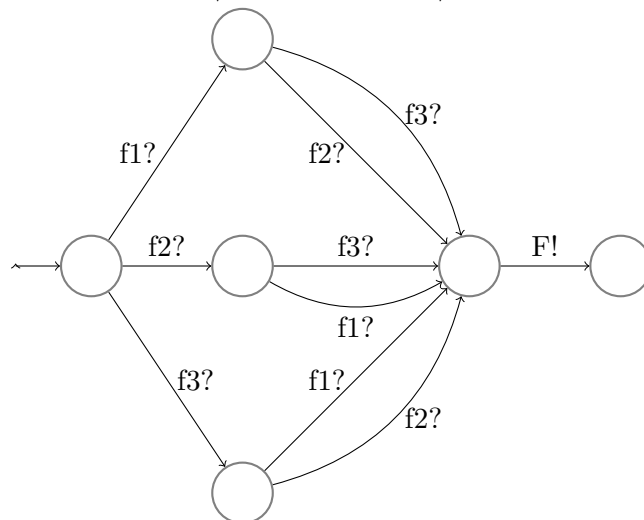
Als erstes betrachten wir die Struktur der I/O-IMCs, welche die Basisevents des DFTs repräsentieren. In Abbildung 3.1 ist die entsprechende I/O-IMC für ein Basisevent mit

exponentialverteilter Ausfallwahrscheinlichkeit mit der Rate λ in aktivem und μ in schlafendem Zustand zu sehen. Initial ist das BE schlafend, wartet also entweder darauf geweckt zu werden ($a?$) oder mit Rate μ auszufallen. Sobald das BE aktiv ist, beträgt die Ausfallrate λ . Für phasentypverteilte BEs (PHBEs) ergeben sich größere I/O-IMCs. Sie sind so aufgebaut, dass es für jede aktive und passive Phase einen Zustand gibt. Durch $a?$ kann von einem passiven Zustand in Phase i in einen aktiven Zustand in Phase i gewechselt werden. Dazu gibt es noch Transitionen zwischen den einzelnen aktiven bzw. zwischen den einzelnen passiven Zuständen, welche den Übergang von einer Phase in eine andere realisieren. Ihre Raten hängen von der jeweiligen Phasentypverteilung ab. Von jedem Zustand aus gibt es dann noch eine Transition mit einer entsprechenden Rate (abhängig von der Phase und ob aktiv oder schlafend) zu einem Zustand, welcher den Ausfall des PHBE markiert. Genauereres dazu ist wieder in [3] zu finden.

K/M VOTING-Gates

In Abbildung 3.2 ist eine I/O-IMC für ein 2/3 VOTING-Gate zu sehen. Dabei wird mittels $f1?$, $f2?$ und $f3?$ auf den Ausfall der entsprechenden Eingänge gewartet und F verkündet den Ausfall des VOTING-Gates.

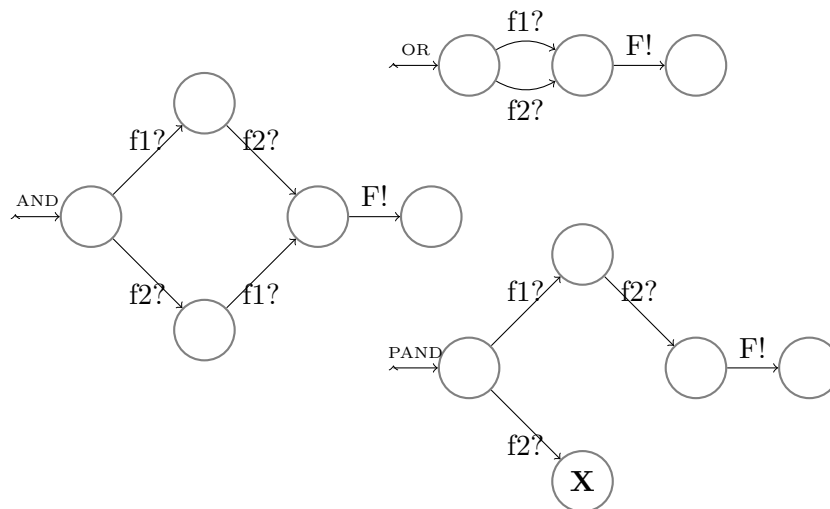
Abbildung 3.2.: I/O-IMC für ein 2/3 VOTING-Gate



AND-, PAND- und OR-Gates

Abbildung 3.3 zeigt die I/O-IMCs für AND-, PAND- und OR-Gates mit jeweils zwei Eingängen. Das AND-Gate kann dabei generell als Sonderfall für ein K/M VOTING-Gate mit $K = M$ angesehen werden. $f1?$ und $f2?$ zeigen dabei wieder das Warten auf den Ausfall der Eingänge an und $F!$ verkündet den Ausfall des jeweiligen Gates. Das AND- und das OR-Gate sind selbsterklärend, bei dem PAND-Gate spielt die Reihenfolge des Ausfalls eine Rolle. So wird in dem Beispiel verlangt, dass zuerst $f1?$ und dann $f2?$ erfüllt wird. Wird zuerst in der entsprechenden I/O-IMC $f2!$ gefeuert, so gelangt man in einen Deadlock und $F!$ wird niemals ausgeführt, das PAND-Gate fällt also niemals aus.

Abbildung 3.3.: I/O-IMCs für AND-, PAND- und OR-Gate

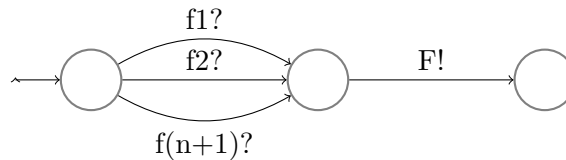


FDEP-Gates

Die Umsetzung des FDEP-Gates findet nicht als simple Übersetzung eines Gates in eine I/O-IMC statt, sondern es wird für jedes abhängige Element eine eigene I/O-IMC gebildet. Diese sieht dann so aus wie in Abbildung 3.4. Hierbei ist das Element von n Triggern abhängig, welche durch das Warten auf $f2$ bis $f(n + 1)$ dargestellt sind. $f1?$ ist das Warten auf den Ausfall des abhängigen Elements selber, welches, wie bereits erwähnt, durch eine andere I/O-IMC (siehe Abbildung 3.1) beschrieben wird. Erhält die I/O-IMC also die Nachricht des Ausfalls eines Triggers (oder des Elements selber durch die ursprüngliche I/O-IMC), so verkündet es den Ausfall des abhängigen Elements per $F!$. Der Aufbau ist also analog zu dem eines OR-Gates mit $n + 1$ Eingängen. Anhand

der I/O-IMC wird ersichtlich, dass sowohl Trigger als auch abhängige Elemente keine BEs sein müssen, sondern auch andere Gates sein können.

Abbildung 3.4.: I/O-IMC für abhängige Elemente



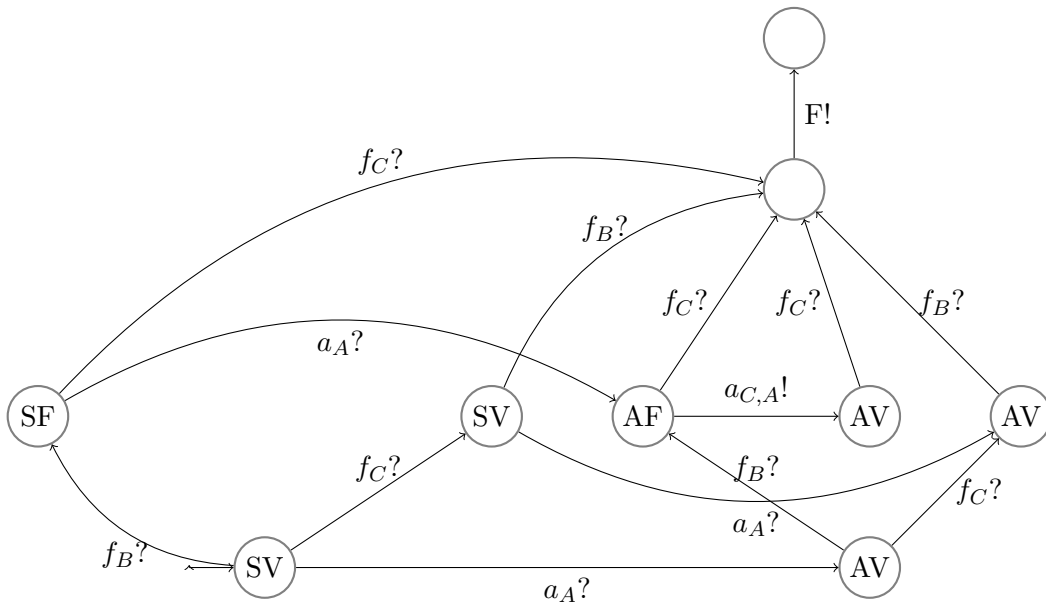
SPARE-Gates

Als Letztes wird die Umwandlung von SPARE-Gates betrachtet. Hierbei gibt es mehrere Punkte zu beachten. Zum einen muss berücksichtigt werden, dass Ersatzelemente mehreren unterschiedlichen SPARE-Gates zur Verfügung stehen können. Zum anderen gilt weiterhin, dass weder primär- noch Ersatzelemente BEs sein müssen. Ersatzelemente schlafen nach Definition so lange, bis sie gebraucht werden. Im Normalfall wird dies direkt auf die zugrundeliegenden BEs (anders gesagt auf alle Blätter des Baums mit dem Ersatzelement als Wurzel) übertragen. Bei der Interpretation der bisherigen Gates spielt dies bei der Umwandlung keine Rolle. Ist jedoch ein SPARE-Gate $SP2$ ein Ersatzelement eines anderen SPARE-Gates $SP1$, so ist es wichtig, bei $SP2$ zwischen schlafendem und aktivem Zustand zu unterscheiden. Angenommen $SP2$, und damit auch sein primäres und seine Ersatzelemente, schläft und sein primäres Element fällt aus. Dann wird das erste Ersatzelement von $SP2$ nur dann aktiviert, wenn $SP2$ aktiv ist. Andernfalls schläft es weiter. Zur Bildung der I/O-IMC für ein SPARE-Gate müssen also vier Fälle unterschieden werden: Das SPARE-Gate kann schlafen (S) oder aktiv (A) sein und das primäre Element kann verwendbar (V) oder ausgefallen (F) sein. Je nach Zustand des Gates und des primären Elements, kann der Zustand der I/O-IMC einer Menge SV , SF , AV oder AF zugezählt werden. Dadurch können die möglichen Transitionen wie folgt eingeteilt werden. Ist der aktuelle Zustand in SV , so kann entweder das Gate geweckt werden, also in einen entsprechenden Zustand in AV gewechselt werden, das primäre Element ausfallen (Wechsel in entsprechenden Zustand in SF) oder eines der Ersatzelemente von einem anderen SPARE-Gate in Anspruch genommen werden (Zustandswechsel innerhalb von SV). Ist der aktuelle Zustand in SF , so kann aus gleichen Gründen wie zuvor ein Zustandswechsel in die Menge AF oder innerhalb von SF stattfinden. Zusätzlich besteht die Möglichkeit, dass kein Ersatzelement mehr vorhanden ist und $SP2$ somit seinen Ausfall melden kann. Bei einem Zustand in AV besteht die Möglichkeit des Wechsels nach AF oder innerhalb von AV (gleiche Begründung wie zuvor). Für Zustände in AF gibt es nun die Möglichkeit, das erste verfügbare Ersatzelement anzufordern, und so wieder

in AV zu wechseln. Weiterhin kann auch wieder mit der gleichen Begründung wie bisher ein Zustandswechsel innerhalb von AF stattfinden.

In Abbildung 3.5 ist als Beispiel eine I/O-IMC für ein simples SPARE-Gate, bei dem sowohl primäres als auch Ersatzelement BEs sind, gegeben. $f_B!$ und $f_C!$ kennzeichnen dabei den Ausfall der BEs, $a_A!$ kennzeichnet die Aktivierung des SPARE-Gates, $a_{C,A}!$ kennzeichnet die Inanspruchnahme von Ersatzelement C durch dieses SPARE-Gate und $F!$ kennzeichnet den Ausfall des Gates. Falls sich mehrere SPARE-Gates ein oder mehrere Ersatzelemente teilen, muss die Umwandlung so angepasst werden, dass eine Abfrage zur Verfügbarkeit des Ersatzelements vorhanden ist. Ein Beispiel dazu ist in [3] zu finden.

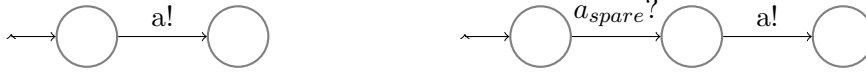
Abbildung 3.5.: I/O-IMC für SPARE-Gate mit BE B als primäres- und BE C als Ersatzelement



IMCs zur Aktivierung

Wie in Abbildung 3.1 und 3.5 zu sehen ist, fehlen noch I/O-IMCs für die Aktivierung der BEs und der SPARE-Gates. Diese sehen, abhängig davon, ob sie Ersatzelemente sind (links) oder nicht (rechts), wie in Abbildung 3.6 aus. Dabei kennzeichnet a_{spare} die Aktivierung des Ersatzelements.

Abbildung 3.6.: I/O-IMCs zur Aktivierung von BEs und SPARE-Gates



3.1.3. Parallele Komposition

Nachdem im vorherigen Abschnitt die Übersetzung der DFT Elemente in einzelne I/O-IMCs behandelt wurde, wird in diesem Abschnitt erklärt, wie zwei I/O-IMCs zu einer I/O-IMC zusammengefasst werden können. Dafür wird zunächst der Operator \parallel für die parallele Komposition eingeführt. Für zwei I/O-IMCs Q und R ist durch $P = Q \parallel R$ deren Vereinigung angegeben. Die Vereinigung setzt sich dabei so zusammen, dass Markov Transitionen und voneinander unabhängige interaktive Transitionen der ursprünglichen I/O-IMCs auch in $Q \parallel R$ unabhängig ausführbar sind, und interaktive Transitionen, über welche synchronisiert werden muss, so nachvollzogen werden, als ob die Aktion in beiden I/O-IMCs durchgeführt wird. Synchronisiert werden zum einen Input-Aktionen, die in beiden I/O-IMCs vorkommen, und zum anderen passende Input- und Output-Aktionen (also z.B. $a?$ und $a!$). Dabei gilt zu beachten, dass die Synchronisation nur dann stattfinden kann, wenn die Aktionen zur selben Zeit durchführbar sind. Das Ergebnis der Synchronisation einer Input- und Output-Aktion ist wiederum eine Output-Aktion. Aktionen, über welche keine Synchronisation mehr statt findet, werden versteckt und somit wie interne Aktionen behandelt. Die [3] entnommene formale Definition der parallelen Komposition ist wie folgt.

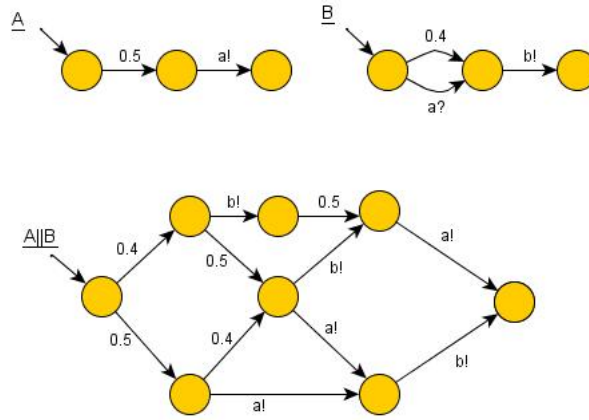
Definition 1. Seien Q und R zwei I/O-IMCs.

1. Q und R sind komponierbar, wenn $A_{out}^Q \cap A_{in}^R = A_i^Q \cap A^R = A_{out}^Q \cap A_i^R = \emptyset$.
2. Wenn Q und R komponierbar sind, dann ist ihre Komposition $Q \parallel R$ der I/O-IMC $(S_Q \times S_R, (s_0^Q, s_0^R), ((A_{in}^Q \cup A_{in}^R) \setminus (A_{out}^Q \cup A_{out}^R), (A_{out}^Q \cup A_{out}^R), (A_i^Q \cup A_i^R)), \rightarrow_{Q \parallel R}, \Rightarrow_{Q \parallel R})$ mit

$$\begin{aligned} \rightarrow_{Q \parallel R} = & \{(s, t) \rightarrow_{Q \parallel R}^a (s', t) \mid s \xrightarrow{a} s' \wedge a \in A_Q \setminus A_R\} \\ & \cup \{(s, t) \rightarrow_{Q \parallel R}^a (s, t') \mid t \xrightarrow{a} t' \wedge a \in A_R \setminus A_Q\} \\ & \cup \{(s, t) \rightarrow_{Q \parallel R}^a (s', t') \mid s \xrightarrow{a} s' \wedge t \xrightarrow{a} t' \wedge a \in A_Q \cup A_R\} \\ \Rightarrow_{Q \parallel R} = & \{(s, t) \xrightarrow{\lambda}_{Q \parallel R} (s', t) \mid s \xrightarrow{\lambda} s'\} \\ & \cup \{(s, t) \xrightarrow{\lambda}_{Q \parallel R} (s, t') \mid t \xrightarrow{\lambda} t'\}. \end{aligned}$$

Ein Beispiel für eine parallele Komposition zweier I/O-IMCs ist in Abbildung 3.7 zu sehen. Dabei findet eine Synchronisation über die Aktion x statt, die betreffenden Tran-

Abbildung 3.7.: Beispiel für parallele Komposition



sitionen können also nur parallel durchgeführt werden. Die restlichen Transitionen sind unabhängig voneinander durchführbar.

3.1.4. Schwache Bisimulation

Um die Größe von I/O-IMCs zu verringern, werden Zustandsäquivalenzen benötigt. In diesem Abschnitt wird die schwache Bisimulation Relation, welche in CORAL zum Einsatz kommt, um das state Explosion Problem abzuschwächen, bezüglich I/O-IMCs eingeführt.

Die Idee bei der Bisimulation im Allgemeinen ist es, Zustände, welche sich gleich verhalten, zu einem einzelnen Zustand zusammenzufassen. Bei der schwachen Bisimulation wird weiterhin von internen Aktionen abstrahiert, wodurch eine stärkere Reduzierung möglich wird. Die folgende Definition ist aus [3] und basiert auf jener aus [12], passt diese jedoch in wenigen Punkten den hier benötigten Ansprüchen an.

Definition 2. Sei $M = (S, Act, \rightarrow, \Rightarrow, s_0)$ eine I/O-IMC und $R \subseteq S \times S$ eine Äquivalenzrelation. Dann ist R genau dann eine schwache Bisimulation, wenn für alle $(s, t) \in R$, $a \in Act$ gilt:

1. wenn $s \xrightarrow{i}^* \xrightarrow{a} \xrightarrow{i}^* s'$, $s' \in S$, dann $t \xrightarrow{i}^* \xrightarrow{a} \xrightarrow{i}^* t'$ für ein $t' \in S$ mit $(s', t') \in R$.
2. wenn $s \xrightarrow{i}^* s'$, $s' \in S$, und s' hat keine ausgehende interne oder output Transition, dann gibt es ein ebensolches $t' \in S$, sodass $t \xrightarrow{i}^* t'$ und $\mathbf{R}(s', Pre_i(C)) =$

$\mathbf{R}(t', \text{Pre}_i(C))$ für alle Äquivalenzklassen $C \in (S/R) \setminus \{[s']_R\}$, wobei $\text{Pre}_i(C) = \{s' \mid \exists s \in C : s' \xrightarrow{i}^* s\}$ die Menge aller Zustände mit Pfaden aus internen Transitionen in C ist.

Zwei Zustände $s, t \in S$ in M sind genau dann schwach bisimilar, geschrieben $s \approx_M t$, wenn es eine schwache Bisimulation R mit $(s, t) \in R$ gibt.

$\approx_M = \bigcup \{R \mid R \text{ ist eine schwache Bisimulation auf } M\}$ ist die schwache Bisimilarität einer I/O-IMC M .

Der erste Punkt ist analog zur schwachen Bisimulation auf LTSs aufgebaut. Der zweite Punkt beschreibt die Bedingungen für Zustände mit Markov Transitionen. Die Grundidee ist, dass bisimilare Zustände die gleiche Ausgangsrate beim Übergang in eine bestimmte andere Äquivalenzklasse haben. Dabei ist zu beachten, dass Transitionen, welche innerhalb der Äquivalenzklasse verlaufen, anders als in [12], nicht beachtet werden. Dies kann hier angewendet werden, da nicht nur, wie zuvor erwähnt, CORAL vor der Analyse der I/O-IMCs im Fall von vorkommendem Nichtdeterminismus eine Uniformierung durchführt, sondern auch im anderen Fall der benutzte Algorithmus von CADP eine Uniformierung durchführt [29]. Dadurch hat jeder Zustand dieselbe Gesamtausgangsrate. Da die Gesamtausgangsrate in andere Äquivalenzklassen für bisimilare Zustände übereinstimmen muss, stimmt damit automatisch auch die Gesamtausgangsrate innerhalb der eigenen Äquivalenzklasse überein. Dies kann bereits jetzt, also vor der Uniformierung, ausgenutzt werden. Das Ergebnis ist eine stärkere Minimierung. Außerdem ist anzumerken, dass \approx_M die größte schwache Bisimulationsrelation auf M ist und sowohl parallele Komposition als auch das Verstecken interaktiver Transitionen die schwache Bisimilarität nicht beeinflussen.

3.1.5. Nichtdeterminismus in DFTs

Um die Präsenz von Nichtdeterminismus in dynamischen Fehlerbäumen aufzuzeigen, betrachte man zunächst die Gates *priority AND* (*PAND*) und *SPARE*. Man nehme bei einem PAND-Gate mit zwei Eingangselementen an, dass beide genau gleichzeitig ausfallen. Die Frage ist nun, wie das PAND-Gate zu reagieren hat. Die gleiche Frage tritt auf, wenn zwei SPARE-Gates sich ein Ersatzelement teilen und die primären Elemente der beiden Gates gleichzeitig ausfallen. Welches Gate soll nun das Ersatzelement bekommen? Da die Wahrscheinlichkeit dieser Fälle jedoch bei Null liegt, wird zur weiteren Argumentation noch das *functional dependency* (*FDEP*)-Gate benötigt. Erweitert man die zuvor beschriebene Situation darum, dass die beiden Eingänge des PAND-Gates (siehe Abbildung 4.1) bzw. die Primärelemente des SPARE-Gates (siehe z.B. Abbildung 4.3) abhängig von dem Triggerevent X des FDEP-Gates sind, wird die Relevanz bei einem Ausfall von X der zuvor erwähnten Frage sichtbar. Als Lösung gibt es zum einen die

Möglichkeit, dass mit Hilfe einer Wahrscheinlichkeitsverteilung entschieden wird, welches Element zuerst ausfällt, wie zum Beispiel in [5]. In CORAL wird jedoch genau dieses Verhalten als Nichtdeterminismus aufgefasst. In den von CORAL ausgegebenen I/O-IMCs wird der Nichtdeterminismus durch Transitionen mit dem Label i kenntlich gemacht. Dabei kann es, wie auch eines der Beispiele in Kapitel 4.1 zeigen wird, dazu kommen, dass zwar eine der obigen Situationen auftritt, der Nichtdeterminismus aber keine Rolle bei der abschließenden Analyse spielt. Dort können noch die unterschiedlichen Fälle auftreten, dass der Nichtdeterminismus in dem entstandenen I/O-IMC noch zu sehen ist, und im Gegensatz dazu, dass er durch die Minimierung nicht mehr ersichtlich ist.

3.2. IMCA

Der *Interactive Markov Chain Analyzer* (IMCA) [30] ist ein Tool zur Analyse von IMCs. Dabei unterstützt es die Analyse der

Time-Bounded Reachability die Berechnung der maximalen und minimalen Wahrscheinlichkeit eines jeden Zustands, eine bestimmte Zielmenge innerhalb von t Zeiteinheiten zu erreichen,

Interval-Bounded Reachability die Berechnung der maximalen und minimalen Wahrscheinlichkeit eines jeden Zustands, eine bestimmte Zielmenge innerhalb des Zeitintervalls $[a, b]$ zu erreichen,

Unbounded Reachability die Berechnung der maximalen und minimalen Wahrscheinlichkeit eines jeden Zustands, eine bestimmte Zielmenge zu erreichen,

Expected Time die Berechnung der maximalen und minimalen erwarteten Zeit eines jeden Zustands, eine bestimmte Zielmenge zu erreichen,

Expected Steps die Berechnung der maximalen und minimalen erwarteten Anzahl von Schritten eines jeden Zustands, eine bestimmte Zielmenge zu erreichen.

Zusätzlich ist die explizite Berechnung der Scheduler für die maximale *time-bounded* und *interval-bounded Reachability* möglich. Innerhalb IMCAs findet eine Minimierung mittels starker Bisimulation statt. Da IMCA jedoch im Kontext mit CORAL untersucht wird - und die Ergebnisse, die CORAL liefert, bereits mittels schwacher Bisimulation in mindestens demselben Ausmaße minimiert wurden - spielt die starke Bisimulation im Folgenden keine Rolle.

Von den zuvor aufgezählten Punkten werden in dieser Arbeit die interval-bounded Reachability in Abschnitt 3.2.3 und die Expected Time in Kapitel 4.3 behandelt.

Ein weiterer, wichtiger Punkt und der Unterschied zu anderen Model Checkern, ist die Berechnung optimaler Ergebnisse mit Verwendung von timed Schemulern im Gegensatz zur Verwendung von time-abstract Schemulern (Abschnitt 3.2.2).

3.2.1. Eingabeformat

Als Eingabe erhalt IMCA eine IMC in einer .imc Datei oder eine IPC (*Interactive Probability Chain*) in einer .ipc Datei. Die Form der Eingabe ist dabei wie folgt festgelegt: Zunachst, in der ersten Zeile, die Auflistung der Startzustande, getrennt durch Leerzeichen und endend mit „//“. In der zweiten Zeile folgt die ebenfalls durch Leerzeichen und mit „//“ endende Auflistung der Zielzustande. Die weiteren Zeilen beinhalten dann die Transitionen der IMC oder IPC in der Form „Ausgangszustand Folgezustand Wahrscheinlichkeit/Rate/Aktion“.

Die Eingabe fur die IMC aus Abbildung 2.6 mit s_3 und s_4 als Zielzustande wird demnach mit

```
s0 //
s3 s4 //
s0 s1 i
s0 s2 1.9
s1 s0 i
s1 s4 i
s2 s3 0.4
s2 s4 0.2
s3 s1 1
```

dargestellt.

3.2.2. Timed vs. Time-Abstract Scheduler

Um die Bedeutung der Unterschiede zwischen timed und time-abstract Schemulern zu verstehen, sollte zunachst einmal die Berechnung der maximalen Wahrscheinlichkeit zum Erreichen eines Zielzustandes *goal* in dem Intervall I von Zustand s aus betrachtet werden. Die Formel hierfur ist

$$p^{max}(s, I) = \sup_{D \in S, s, D} \Pr(\diamond^I goal).$$

Dabei ist D ein Scheduler aus einer Menge S von Schemulern. Durch die verschiedenen Definitionen dieser Menge S kann sich die berechnete Wahrscheinlichkeit unterscheiden.

Eine Möglichkeit ist es zum Beispiel, S als die Menge aller time-abstract Scheduler zu definieren. Das heißt, die Scheduler können zwar wissen auf welchem Weg sie zu dem aktuellen Zustand gekommen sind, allerdings nicht wie viel Zeit dabei vergangen ist, was im Gegensatz dazu die Besonderheit der timed Scheduler ist. Offensichtlich gilt, dass die Wahrscheinlichkeit, die wir mit time-abstract Schemulern erhalten, in keinem Fall größer sein kann, als die entsprechende Wahrscheinlichkeit über timed Scheduler. Dass die andere Richtung nicht gilt, wird in Kapitel 4 an einigen Beispielen zu sehen sein, da MRMC mit time-abstract Schemulern arbeitet, während IMCA timed Scheduler benutzt. Auf die Vor- und Nachteile wird wiederum in Kapitel 4 anhand von Testergebnissen eingegangen. Es ist jedoch bereits festzuhalten, dass folgender Satz gilt.

Theorem 1. *Für alle Schemulermengen S gilt: $p^{max}(s, I) \leq p'^{max}(s, I)$, falls bei p' zur Berechnung die Menge aller timed Scheduler gewählt wurde.*

3.2.3. Berechnung der Erreichbarkeit

In diesem Abschnitt wird näher beschrieben, wie die Berechnung der Wahrscheinlichkeit der Erreichbarkeit einer Zielmenge unter Berücksichtigung von timed Schemulern funktioniert.

Da ein möglicher Ansatz, die Wahrscheinlichkeit direkt über IMCs zu berechnen, aufgrund des nötigen Lösen von integralen Gleichungssystemen über das Maximum von Funktionen nicht effizient lösbar und die iterative Integration numerisch instabil ist [24][2], wird ein Umweg über IPCs gegangen. Eine IPC ist dabei das diskrete Gegenstück zu einer IMC. Statt der Transitionsfunktion \Rightarrow für Markov Transitions hat eine IPC eine Transitionsfunktion $\mathbf{P} : S \times S \rightarrow [0, 1]$, welche die Wahrscheinlichkeit des Übergangs in den Folgezustand angibt. Dabei gilt $\forall s \in S : \mathbf{P}(s, S) \in \{0, 1\}$. Die *maximal progress assumption* gilt auch weiterhin. Des Weiteren beschreibe PS die Menge von Zuständen, für die $\mathbf{P}(s, S) = 1$ und $IT(s) = \emptyset$ gilt, und IS die Menge von Zuständen, für die $IS \neq \emptyset$ und $(P)(s, S) = 0$ gilt. Während IMCs mit kontinuierlichem Zeitfortschritt arbeiten, liegen IPCs diskrete Zeitschritte zugrunde. Die Idee aus [24] ist nun, durch Teilen der Zeit in kleine Zeitstücke und durch entsprechende Umwandlung der IMCs in IPCs, eine step-bounded Erreichbarkeitsanalyse auf IPCs durchzuführen.

Zur Diskretisierung einer IMC $M = (S, Act, \rightarrow, \Rightarrow, s_0)$ in eine IPC $\mathcal{P}_\delta = (S, Act, \rightarrow, \mathbf{P}', s_0)$ wird mittels einer zuvor festgelegten Schrittgröße $\delta \in \mathbb{R}_{>0}$ die Transitionsfunktion \mathbf{P}' mit $\mathbf{P}'(s, s') = \frac{E(s)}{R(s, s')}$ wie folgt definiert:

$$\mathbf{P}'(s, s') = \begin{cases} (1 - e^{-E(s) \cdot \delta}) \cdot \mathbf{P}(s, s'), & \text{wenn } s \neq s', \\ (1 - e^{-E(s) \cdot \delta}) \cdot \mathbf{P}(s, s') + e^{-E(s) \cdot \delta}, & \text{wenn } s = s'. \end{cases}$$

Dabei beschreibt $1 - e^{-E(s) \cdot \delta} \cdot \mathbf{P}(s, s')$ die Wahrscheinlichkeit, dass in M innerhalb von δ

Zeiteinheiten, also innerhalb eines diskreten Schritts, die Transition (s, s') durchgeführt wird, und $e^{-E(s)\cdot\delta}$ die Wahrscheinlichkeit, dass keine Transition vollzogen wird.

Das berechnete Ergebnis der step-bounded Erreichbarkeitswahrscheinlichkeit ist dabei zwar nur eine Approximation der interval-bounded Erreichbarkeitswahrscheinlichkeit, der Fehler ϵ kann aber, je nach Wahl der Schrittgröße δ , beliebig verkleinert werden. Es gilt nach [4]

$$|p_{max}^{\mathcal{P}\delta}(s, (k_a, k_b)) - p_{max}^M(s, (a, b))| \leq (1 - e^{-\lambda\cdot b\cdot\delta}) \cdot (1 - e^{-\lambda\cdot b}) < \epsilon,$$

mit $a = k_a \cdot \delta$ und $b = k_b \cdot \delta$ für $k_a, k_b \in \mathbb{N}_{>0}$ und mit λ als größte Ausgangsrate in M .

IMCA setzt die Berechnung mittels einer modifizierten *value iteration* um, welche für die Berechnung von $p_{max}^{\mathcal{P}}(s, [0, b])$ kurz vorgestellt wird [24].

Es werden die zwei Vektoren $\vec{v}_i \in [0, 1]^{|S|}$ und $\vec{u}_i \in [0, 1]^{|S|}$, wobei $i = 0, 1, \dots, b$ den aktuellen Iterationsschritt darstellt, verwendet. $\vec{v}_i(s)$ gibt die Wahrscheinlichkeit nach i diskreten Schritten wieder. Dabei wird pro Iterationsschritt, ausgehend von $\vec{u}_{i-1}(s)$, die Auswirkung eines diskreten Schritts betrachtet. $\vec{u}_i(s)$ ermittelt den maximalen (bzw. minimalen) Wert über $\vec{v}_i(s')$ für die s' , welche durch interne Transitionen von s aus erreicht werden können. Für die Berechnung des Maximums gilt initial $\vec{v}_0(s) = 1$ für $s \in G$ und $\vec{v}_0 = 0$ für $s \in S \setminus G$ und für die weiteren k Schritte gilt

$$\vec{v}_i(s) = \begin{cases} \sum_{s' \in S} \mathbf{P}(s, s') \cdot \vec{u}_{i-1}(s'), & \text{wenn } s \in PS, \\ 1, & \text{wenn } s \in PS \cap G, \\ \vec{u}_{i-1}(s), & \text{wenn } s \in IS. \end{cases}$$

Für die Berechnung der \vec{u}_i gilt $\vec{u}_i(s) = \max\{\vec{v}_i(s') \mid s \xrightarrow{i}^* s'\}$, wobei \xrightarrow{i}^* die reflexive, transitive Hülle von \xrightarrow{i} beschreibt. Nach k Iterationsschritten enthält $\vec{u}_k(s)$ dann $p_{max}^{\mathcal{P}}(s, [0, b])$.

Der Algorithmus arbeitet in $\mathcal{O}(n^{2.376} + (m + n^2) \cdot \frac{(\lambda \cdot b)^2}{\epsilon})$ [24].

Bei der Berechnung des Minimums muss eine Anpassung durch mögliche Kreise aus internen Transitionen vorgenommen werden, welche dem in Kapitel 4.3.1 beschriebenen Prinzip folgt und in [24] näher beschrieben wird. Auch die Berechnung für Intervalle $[a, b]$ mit $a \neq 0$ wird ausgelassen, da sie im Kontext dieser Arbeit eher keine Rolle spielt. Auch diese Anpassung ist in [24] nachzulesen.

Es bleibt festzuhalten, dass IMCA, auf Kosten einer relativ hohen Laufzeit, die optimale time-bounded Erreichbarkeitswahrscheinlichkeit über timed Scheduler beliebig approximieren kann.

3.3. MRMC

Der *Markov Reward Model Checker* (MRMC) [25] unterstützt die Logiken PCTL für DTMCs, CSL für CTMCs und - eingeschränkt - für CTMDPIs (wobei das *I* für den Nichtdeterminismus mit internen Aktionen steht), sowie die Reward Erweiterungen dieser Logiken PRCTL für DMRMs und CSRL für CMRMs. Die Einschränkung der Analyse von CTMDPs mit MRMC betrifft *steady-state* und *unbounded-time reachability*.

Wie zuvor erwähnt, wird MRMC von CORAL zur time-bounded Erreichbarkeitsanalyse von CTMDPs, welche nur noch interne Aktionen beinhalten, benutzt. Dabei sind in MRMC unterschiedliche Algorithmen für uniformierte und nicht uniformierte CTMDPs implementiert [14]. Beide arbeiten mit time-abstract Schemulern. Für uniformierte Algorithmen gibt es einen effizienten Algorithmus, welcher in $\mathcal{O}(E \cdot t \cdot |S|^2 \cdot |Act|)$ [1], mit E als Gesamtausgangsrate der Zustände und t als Zeitschranke, arbeitet. Für diesen Algorithmus ist der genaue Pfad irrelevant und lediglich die Länge dessen wird benötigt. Dies reicht für die Analyse von nicht uniformierten CTMDPs nicht. Dadurch steigt die worst-case Zeitkomplexität erheblich an. Sie ist exponential in der größten Ausgangsrate des CTMDP, der Zeitschranke und der Fehlerrate. Dadurch, dass CORAL den CTMDP vor der Analyse uniformiert, kann der schnellere Algorithmus für uniformierte CTMDPs benutzt werden.

Es bleibt also festzuhalten: MRMC kann für uniformierte CTMDPs effizient die time-bounded Erreichbarkeitswahrscheinlichkeit berechnen. Diese ist jedoch nur optimal bezogen auf time-abstract Scheduler.

3.4. Output CORAL \rightarrow Input IMCA

In Abschnitt 3.2.1 ist das Eingabeformat beschrieben, welches IMCA für die IMC erwartet. CORAL speichert die zu analysierende IMC im BCG Format [27]. Dabei gibt es drei verschiedene Varianten der IMCs (siehe Abbildung 3.8): Einmal speichert CORAL die ursprüngliche I/O-IMC mit Aktionen zur Aktivierung in jedem Zustand, wodurch Selfloops mit internen Aktionen entstehen, welche jedoch aufgrund der Geschlossenheit der IMC nicht nötig sind und die Analyse aufgrund der maximal progress Eigenschaft behindern. Außerdem wird eine Variante ohne diese Aktionen gespeichert. Als drittes ist die uniformierte Variante der IMC ohne Aktivierungsaktionen vorhanden, wodurch Selfloops mit hohen Raten entstehen, welche die Analyselaufzeit von IMCA stark ansteigen lassen würde, ohne, aufgrund des timed Schemulers, einen Mehrwert zu bringen. Um also IMCA mit CORAL zu kombinieren, ist eine Anpassung der Darstellung nötig, welche in diesem Kapitel beschrieben wird.

Zunächst gilt es zu betrachten, welche Änderungen an der IMC selber vorgenommen werden müssen. Zunächst können ohne Einschränkung alle Selfloops entfernt werden, da diese entweder die erwähnten Aktivierungsaktionen sind, oder durch die Uniformierung entstanden sind. Die in Abbildung 3.8 (links) zu sehende Aktivierungstransition von Zustand 0 zu Zustand 3 beeinflusst das Ergebnis nicht, da es sich um eine interne Transition handelt.

Eine weitere wichtige Anpassung betrifft die F -Transition, welche den Ausfall des Systems kennzeichnet. Bei IMCA muss ein Zielzustandsmenge angegeben werden, nach welcher sich die Berechnung richtet. In dem Beispiel aus Abbildung 3.8 ist es jedoch nicht möglich eine adäquate Zustandsmenge zu wählen. Mit der Wahl von Zustand 1 (Zustandsbezeichnungen nach mittlerer IMC) würde man nicht das gewünschte Ergebnis erreichen, da nicht unterschieden werden würde, ob Zustand 1 von Zustand 0, Zustand 2 oder Zustand 4 aus erreicht wird, obwohl nur das Erreichen von Zustand 4 aus den Ausfall markiert. Gelöst wird dies, indem ein neuer Zustand s_F eingefügt wird, zu welchem alle F -Transitionen hin umgeleitet werden und welcher als einziger Zielzustand definiert wird.

Außerdem muss aus der Kantenbeschriftung „rate λ “ das „rate“ entfernt werden. Zusätzlich werden die internen Aktionen umbenannt, sodass bei der Betrachtung des Schedulers über IMCA die Wahl unterschieden werden kann. Das Ergebnis der Anpassung des Beispiels ist in Abbildung 3.9 zu sehen.

Abbildung 3.8.: Drei IMC Ausgabevarianten von CORAL

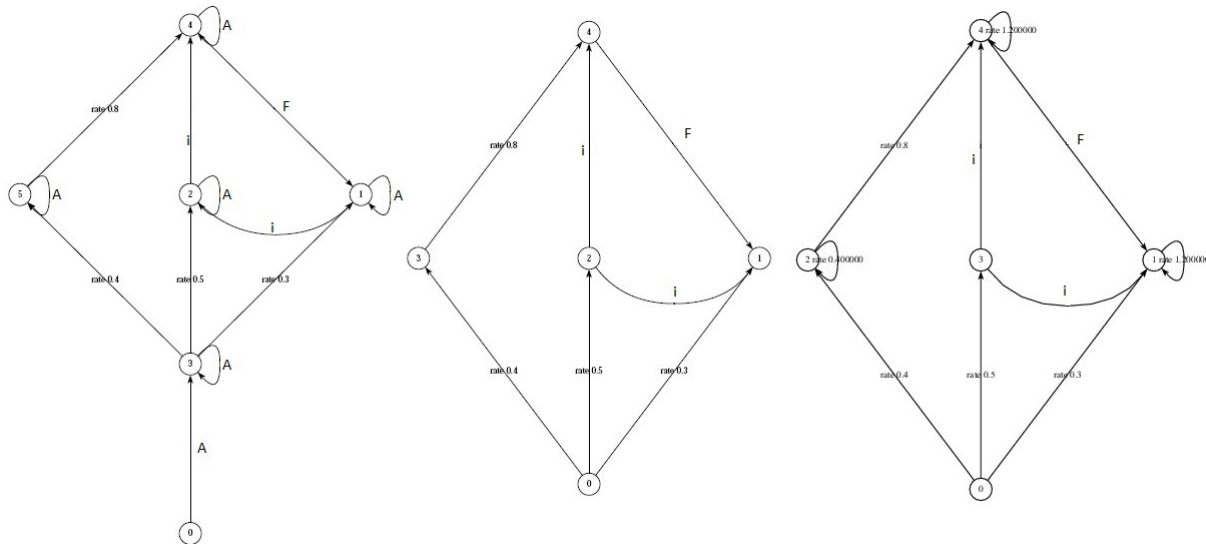
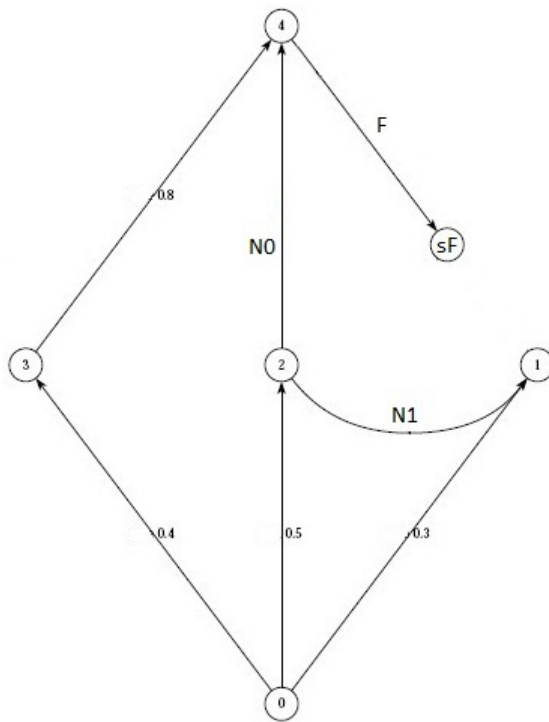


Abbildung 3.9.: Angepasste IMC



Der dokumentierte Quellcode für diese Anpassungen ist im Anhang zu sehen und basiert auf einer Beispielimplementierung von [28].

4. Analyse

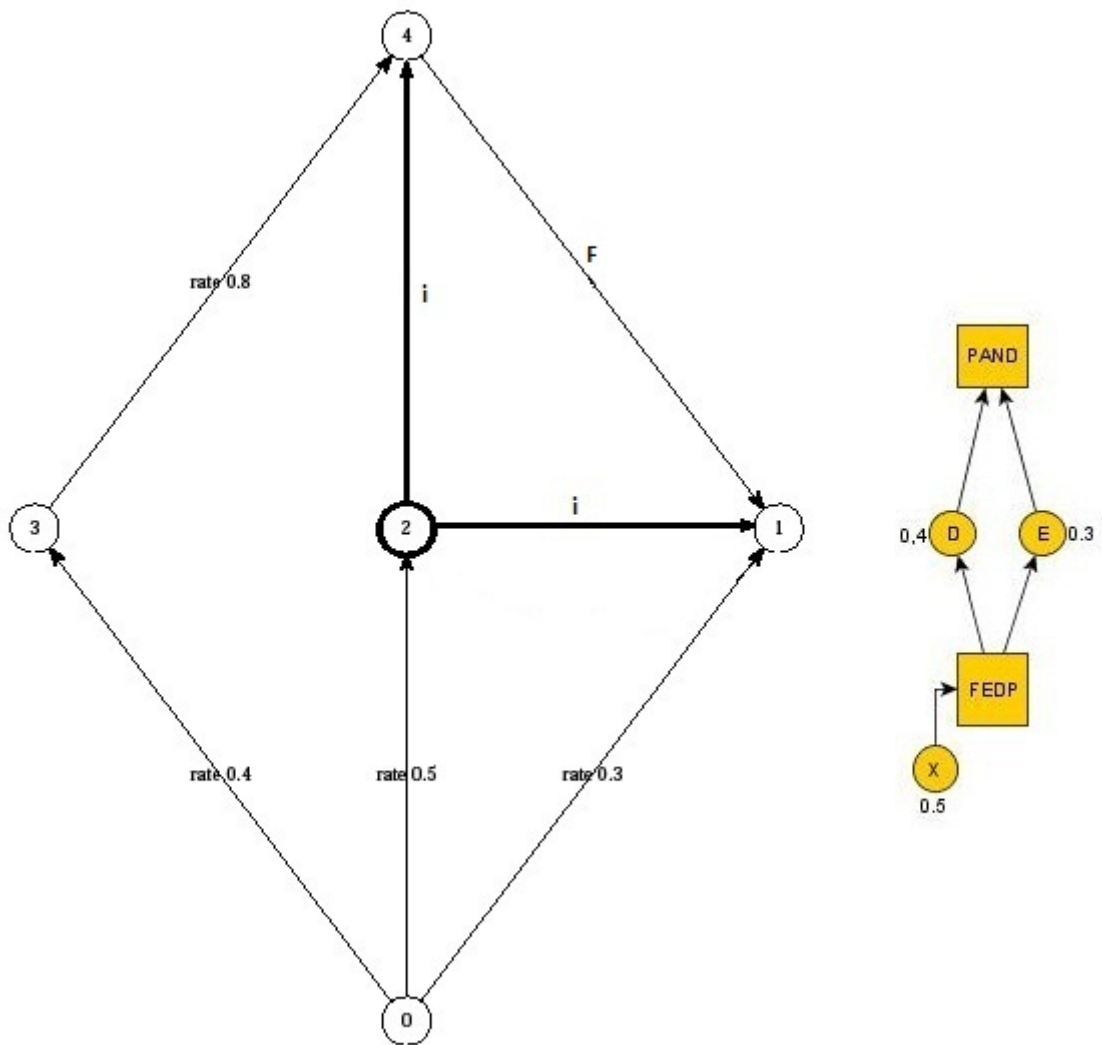
Dieses Kapitel dient zur Motivation der Verknüpfung der Tools CORAL und IMCA. Dazu wird im Folgenden das Auftreten von Nichtdeterminismus aus DFTs entstandener IMCs anhand von acht Beispielen nachgewiesen sowie analysiert. Durch die Analyse wird nicht nur die zwingende Verwendung eines Tools aufgezeigt, welches mit continuous-time Markov decision processes (CTMDPs) umgehen kann, sondern es wird auch ein Vorteil von IMCA gegenüber dem aktuell in CORAL verwendeten Tools MRMC ersichtlich. Dieser liegt in dem Gebrauch von timed Schemulern im Gegensatz zu time-abstract Schemulern, wodurch eine Optimierung der Ergebnisse gewährleistet werden kann.

Das Kapitel ist so aufgebaut, dass zunächst in Abschnitt 4.1 ein paar Beispiele vorgestellt werden. Dabei wird beispielhaft auf die Wandlung von DFTs zu IMCs eingegangen. Des Weiteren wird die Rolle des Nichtdeterminismus in den Beispielen erläutert. In Abschnitt 4.2 werden die aus der Analyse der Beispiele mit CORAL und IMCA erzielten Ergebnisse verglichen und interpretiert sowie die Untersuchung eines DFTs zur Motivierung der Benutzung von IMCA in CORAL. Geschlossen wird dieses Kapitel mit Abschnitt 4.3, in welchem Erwartungswerte für das Erreichen eines Zielzustandes anhand der vorgestellten Beispiele untersucht werden.

4.1. Beispiel DFTs

Das erste Beispiel in Abbildung 4.1 zeigt die zuvor beschriebene Kombination eines PAND- mit einem FDEP-Gate. Die entsprechende I/O-IMC ist daneben abgebildet. Die Fehlerraten sind für das Triggerevent X und die Basisevents D und E $\lambda(X) = 0.5$, $\lambda(D) = 0.4$, $\lambda(E) = 0.3$. Sollte also X ausfallen, bevor D oder E dies tut, muss die Reihenfolge dessen nichtdeterministisch entschieden werden. Dies ist in der entsprechenden I/O-IMC an dem Übergang von Zustand 0 zu Zustand 2 zu sehen. Dort gibt es nun die Möglichkeit, dass zuerst E ausfällt, was gleichbedeutend mit einem False-Output des PAND-Gates ist, und andererseits die Möglichkeit, dass zuerst D und danach E ausfällt, wodurch das PAND-Gate „true“ zurückgibt. Dies ist durch das Erreichen des Zustands 4 gekennzeichnet, welcher durch die Aktion F wiederum den Ausfall des gesamten Systems verkündet.

Abbildung 4.1.: NDT1



Die Entwicklung vom DFT zur IMC ist in Abbildung 4.2 nach den in Kapitel 3 vorgestellten Schritten wiedergegeben. Die IMCs D, E und X beschreiben dabei das Verhalten der entsprechenden BEs, wobei die Komposition mit den IMCs zur Aktivierung der BEs bereits durchgeführt wurde. FDEP-D und FDEP-E beschreiben die aus der Abhängigkeit der BEs entstandenen IMCs. Als erster Schritt wurden jeweils FDEP-D mit D und FDEP-E mit E synchronisiert, sodass es jetzt für jedes BE eine IMC gibt, die das Ausfallverhalten komplett beschreibt. Im Folgenden sind weitere Kompositionen zu sehen, welche jeweils mittels schwacher Bisimulation minimiert sind, bis zuletzt die IMC aus Abbildung 4.1 entsteht.

In Beispiel 2 (Abb. 4.3) wird die zuvor beschriebene Kombination zweier SPARE-Gates mit einem FDEP-Gate benutzt, wobei die beiden SPARE-Gates durch ein OR verbunden sind. Wie zuvor erwähnt tritt hier also nichtdeterministisches Verhalten auf, da entschieden werden muss, welches SPARE-Gate das Ersatzteil G nach Auslösen des Triggers bekommt. In der entsprechenden I/O-IMC ist jedoch kein Nichtdeterminismus zu sehen. Er muss also durch die Minimierung verschwunden sein, woraus direkt folgen muss, dass der Nichtdeterminismus keine Rolle spielen kann. Dies erscheint bei näherer Betrachtung des DFTs auch logisch, da durch das OR-Gate die Relevanz der Zuordnung des Ersatzteils offensichtlich verloren geht.

Die I/O-IMC ist so aufgebaut, dass der Übergang $0 \xrightarrow{0.25} 5$ den Ausfall des Zustands G im schlafenden Zustand beschreibt. Fällt danach X , E oder F aus, so ist mindestens ein SPARE-Gate, und damit das ganze System, ausgefallen. Die Rate von 1.3 beim Übergang $5 \xrightarrow{1.3} 2$ ergibt sich dabei aus der Summe der Ausfallraten ebendieser BEs. Die Übergänge $0 \xrightarrow{0.7} 4$, $0 \xrightarrow{0.4} 3$ und $0 \xrightarrow{0.2} 2$ beschreiben jeweils den Ausfall der BEs E und F bzw. des Triggers X . Durch den Ausfall von X fallen sofort die beiden BEs E und F aus. Da nur ein Ersatzteil vorhanden ist, fällt das System aus. Ist zunächst nur F oder E ausgefallen, so reicht für den Ausfall des Systems das weitere Ausfallen von einem der anderen BEs. Zu beachten ist hier, dass in den Zuständen 3 und 4 das Ersatzteil G aktiv ist.

Abbildung 4.2.: Entstehung des IMC für Beispiel 1

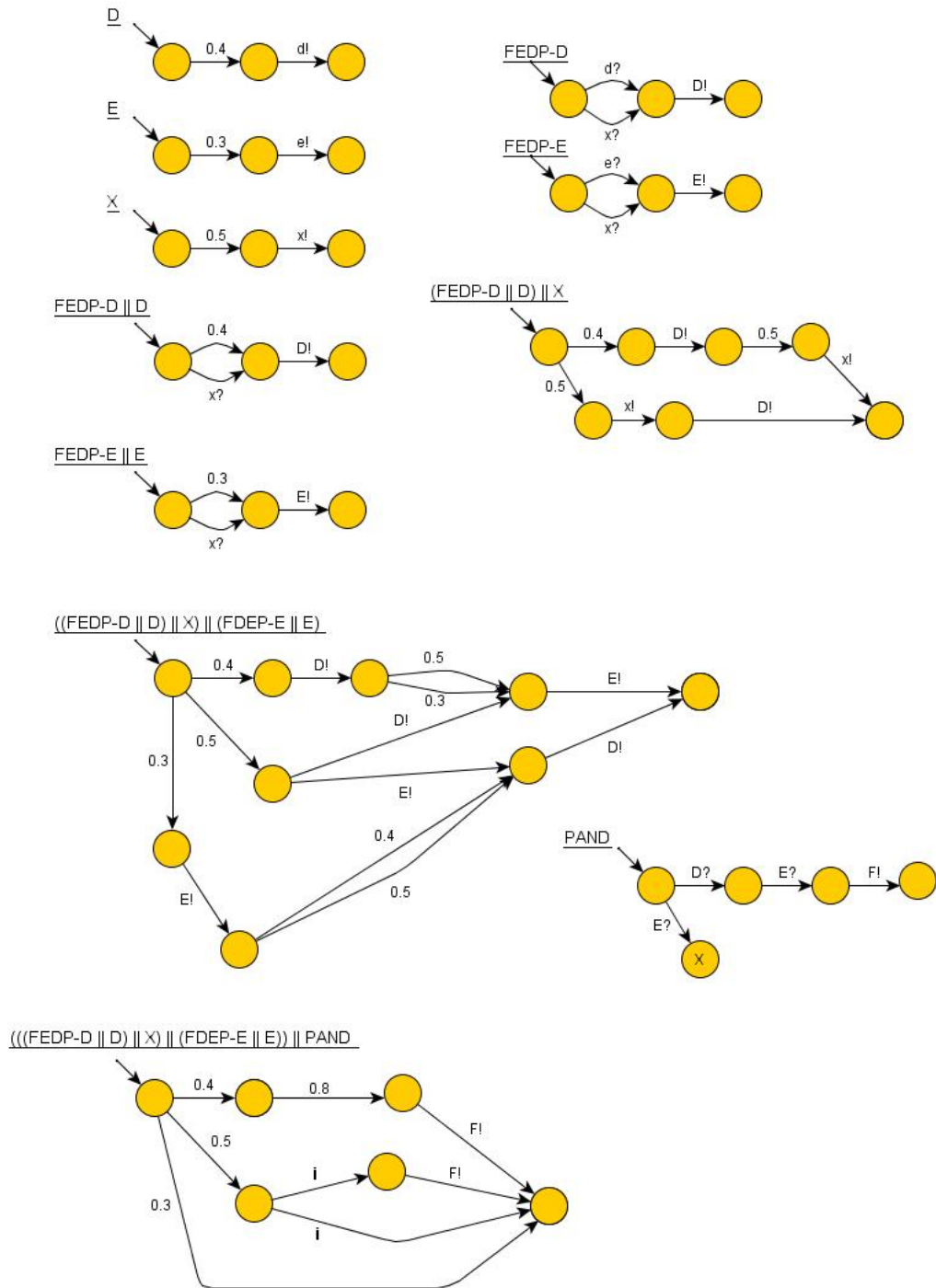


Abbildung 4.3.: NDT2

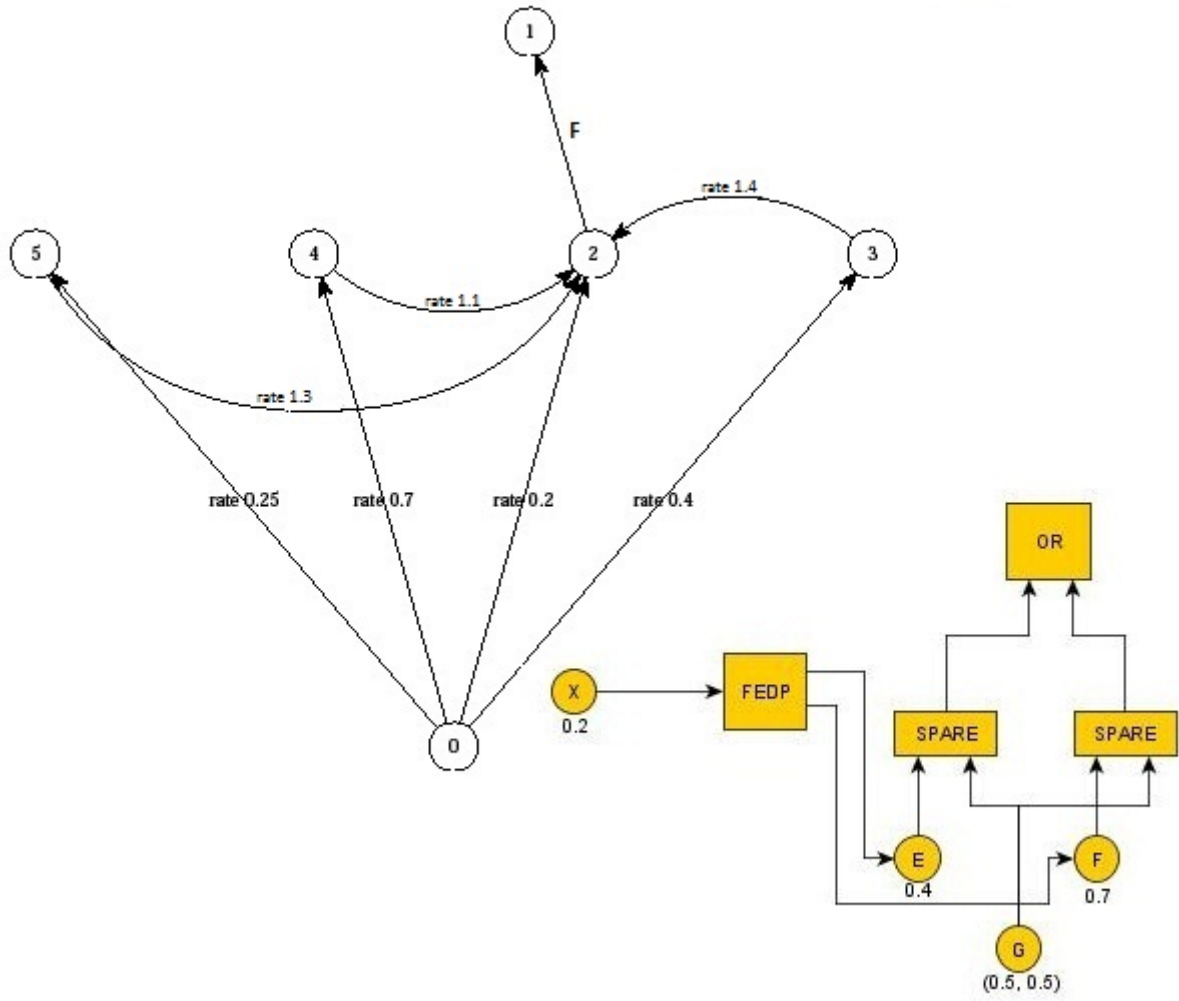
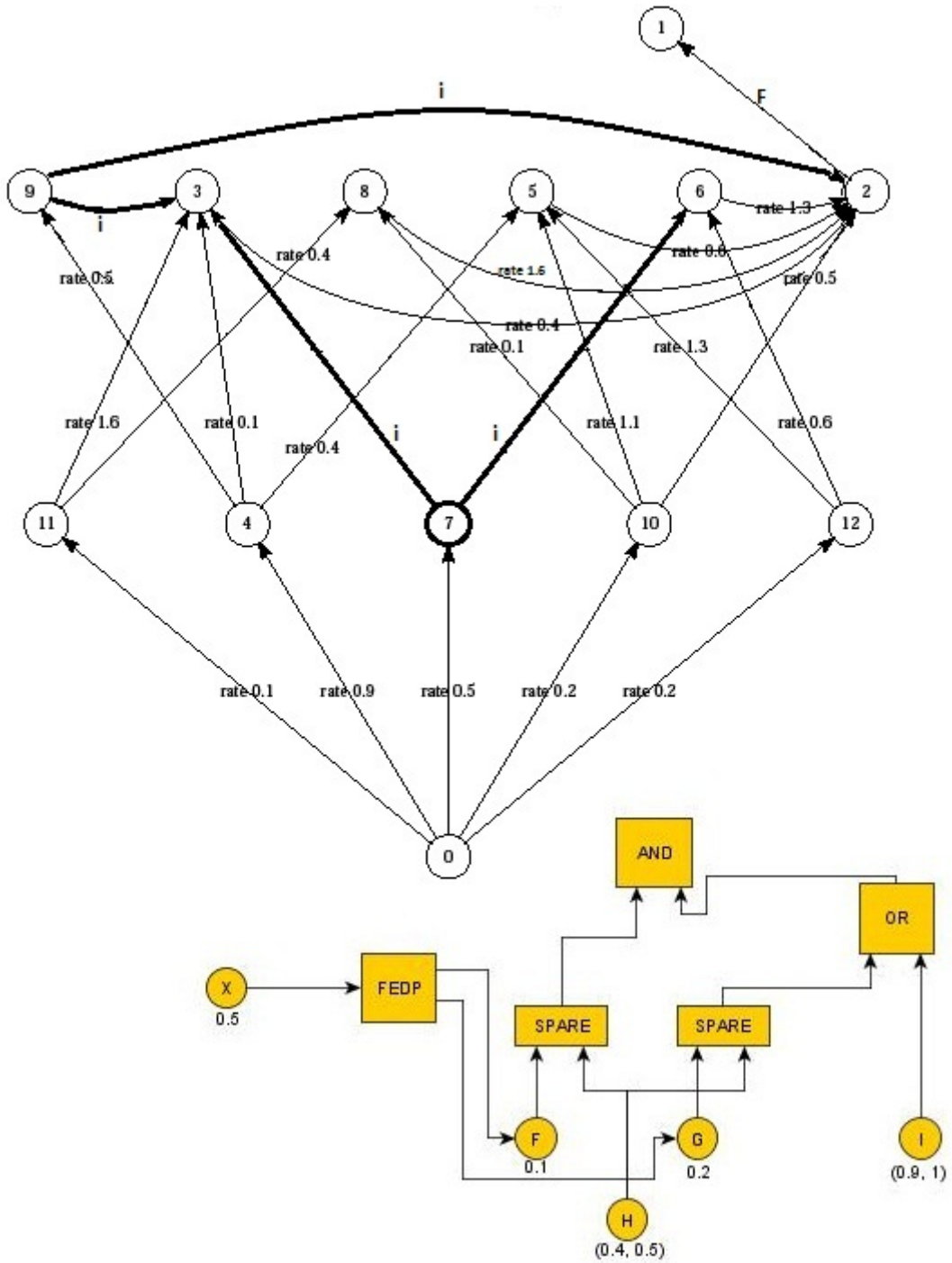


Abbildung 4.4.: NDT3



Im Gegensatz dazu zeigt Beispiel 3 (Abb. 4.4) eine Verwendung dieser Kombination, in der Nichtdeterminismus einen Einfluss auf das Ergebnis hat. Angenommen X fällt aus und das Ersatzteil H ist noch nicht vergeben, dann gibt es zwei Möglichkeiten: Das linke SPARE-Gate bekommt H und es wird jetzt auf den Ausfall von H und I gewartet. Oder das rechte SPARE-Gate bekommt H und es wird auf den Ausfall von H oder I gewartet, wodurch die Wahrscheinlichkeit eines Ausfalls natürlich größer wird. Es ergeben sich also zwei unterschiedlich wahrscheinliche Fälle. Dies ist wiederum auch in Abbildung 4.4 zu sehen, wo in den Zuständen 7 und 9 eine nichtdeterministische Entscheidung getroffen werden muss. Dass der Nichtdeterminismus relevant ist, sieht man in Zustand 7 an den unterschiedlichen Raten der Transitionen der erreichbaren Zustände 3 und 6, welche beide zu Zustand 2 führen, und an Zustand 9, da entweder direkt Zustand 2 erreicht wird, oder durch Zustand 3 das Erreichen von Zustand 2 verzögert wird. Dadurch ist auch direkt ersichtlich, welche Wahl ein Scheduler zum Maximieren bzw. Minimieren der Ausfallwahrscheinlichkeit zu jeder Zeit treffen wird.

Beispiel 4 (Abb. 4.5) zeigt, dass die von einem FDEP-Gate abhängigen Events nicht zwingend Basisevents sein müssen, sondern auch selber Gates sein können. Es handelt sich dabei um ein erweitertes Beispiel aus [3]. Dabei geht es um eine Rundreise, welche dann ausfällt, wenn zuerst das Telefon und danach das Auto ausfällt (ansonsten könnte man zum Beispiel noch den ADAC anrufen, der das Auto repariert). Das Auto wiederum fällt aus, wenn entweder sein Motor oder zwei von fünf Reifen (inklusive Ersatzreifen) ausfallen. Des Weiteren können durch einen Blitzschlag Telefon und Auto auf der Stelle benutzungsunfähig gemacht werden. Da die Reihenfolge des Ausfalls von Telefon und Auto entscheidend ist, wird durch den Blitzschlag in diesem Beispiel die nichtdeterministische Entscheidung relevant. In der entsprechenden I/O-IMC wird dies in Zustand 2 sichtbar. Entweder man geht zu Zustand 6, welcher dann durch die F -Transition den Ausfall des Systems ausgibt, oder man geht zu Zustand 1, wo dies nicht geschieht.

Beispiel 5 (Abb. 4.6) verknüpft SPARE-Gates durch ein PAND-Gate. Die SPARE-Gates teilen sich kein Ersatzteil, sodass der Nichtdeterminismus nicht in dieser Komponente liegt, sondern nur in dem PAND-Gate. Es ist also im Grunde analog zu Beispiel 1 zu sehen. Auch Beispiel 6 (Abb. 4.7) ist nur eine triviale Erweiterung von Beispiel 1.

Abbildung 4.5.: NDT4

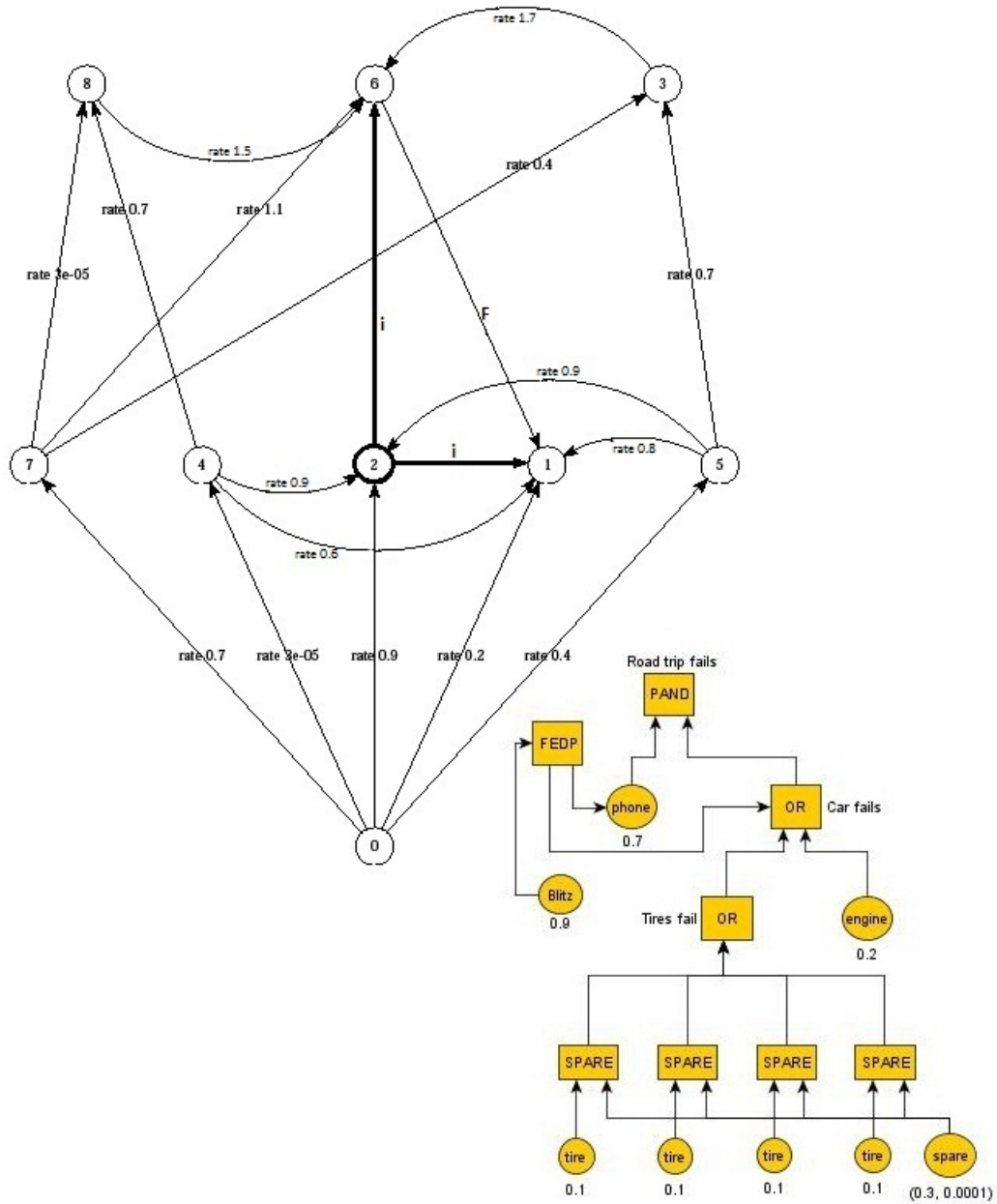


Abbildung 4.6.: NDT5

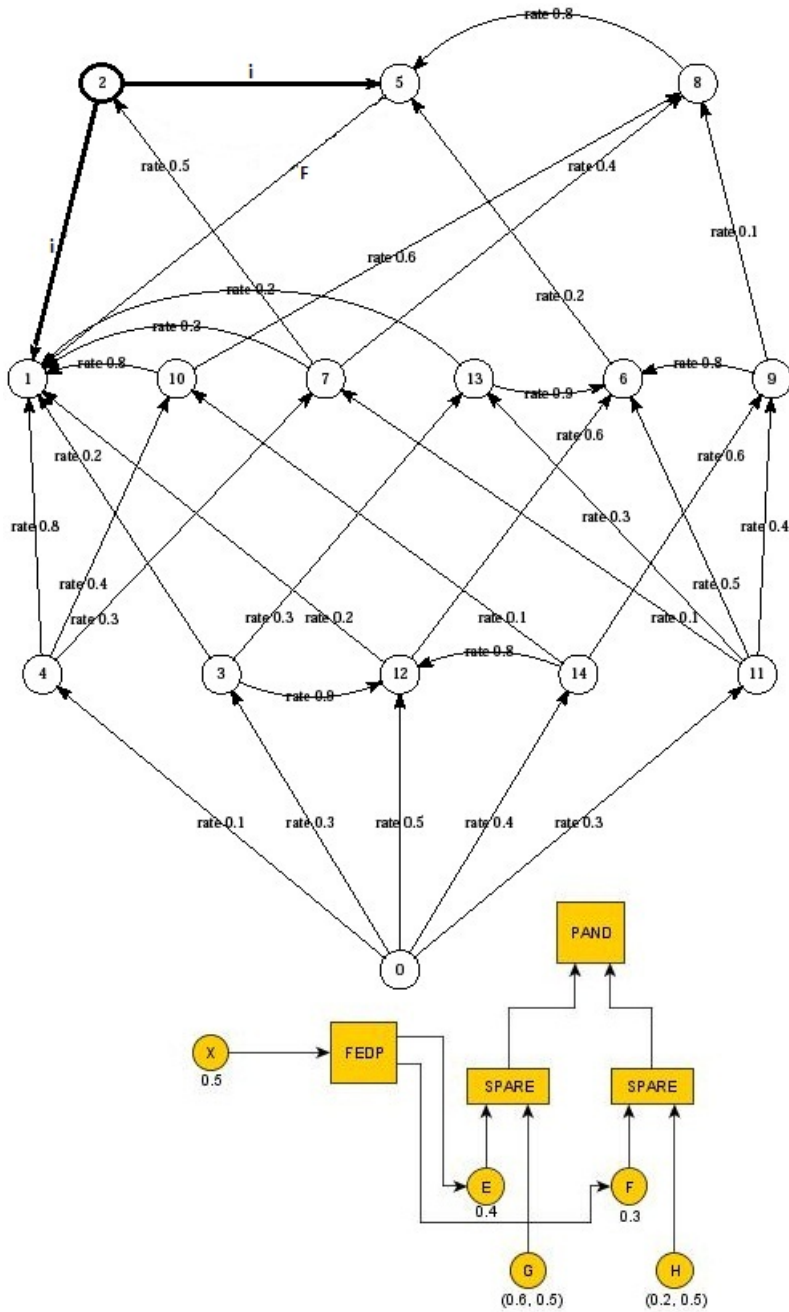
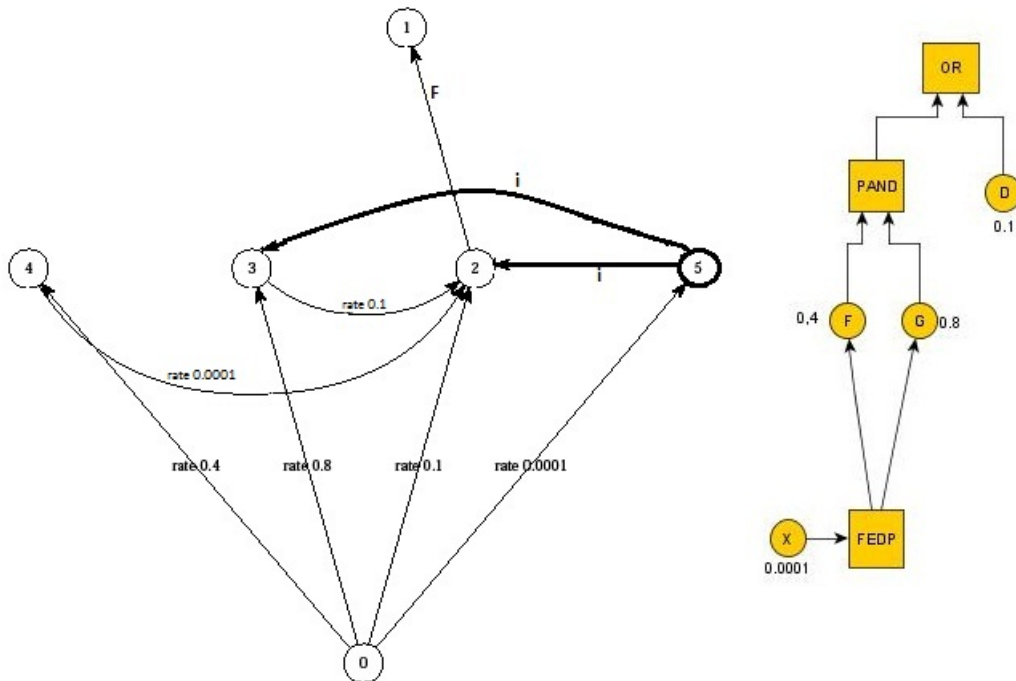


Abbildung 4.7.: NDT6



Etwas interessanter ist wiederum Beispiel 7 (Abb. 4.8). Hierbei handelt es sich um ein Fallbeispiel aus [3]. Es geht um ein einfaches Pumpsystem bestehend aus zwei Hauptpumpen *A* und *B*, zwei Hilfspumpen *C* und *D* und einer Ersatzhilfspumpe *E*. Die beiden Hilfspumpen sind abhängig von einem Trigger *X*, woraus sich der Nichtdeterminismus ergibt.

Das letzte Beispiel (Abb. 4.9), welches jedoch lediglich direkt mit MRMC und IMCA untersucht wurde, wurde [15] entnommen und benutzt eine *Erlang*(30, 10) Verteilung. Der Weg von Zustand 3 führt über 30 Zustände jeweils mit Rate 10 zum Zielzustand 5. Nichtdeterminismus tritt in Zustand 2 auf, von wo aus entweder zu Zustand 3 gegangen werden kann, von welchem man auf jeden Fall nach längerer Zeit Zustand 5 erreicht, oder zu Zustand 4, von wo aus entweder sehr schnell Zustand 5 erreicht wird, oder aber ein Trap-Zustand. Anhand dieses Beispiels wird der Unterschied zwischen timed und time-abstract Schedulern deutlich gemacht.

Abbildung 4.8.: NDPS

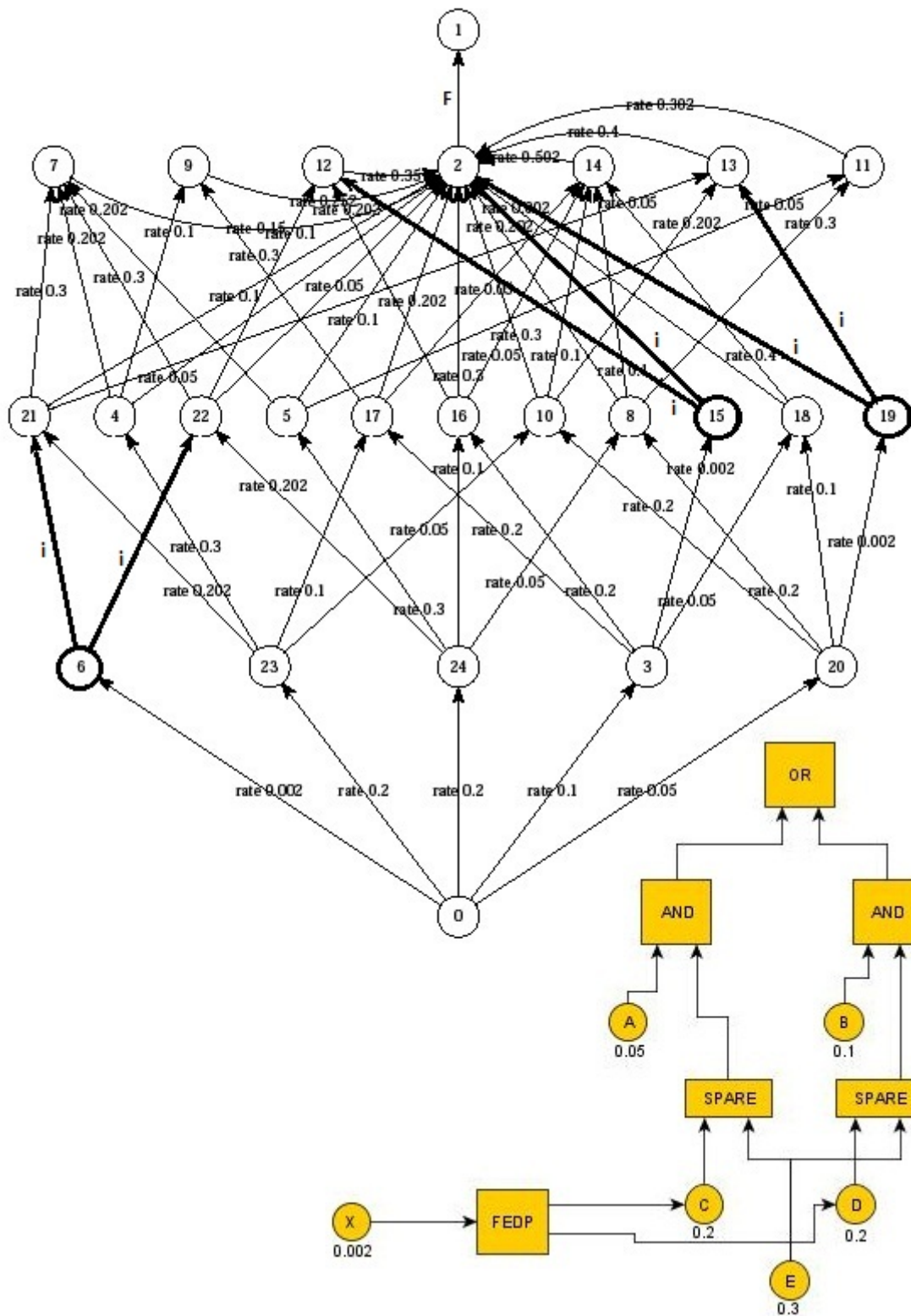
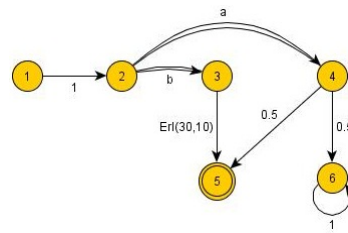


Abbildung 4.9.: Erl(30,10)



4.2. Ausfallwahrscheinlichkeiten der Beispiele

Die Tests wurden alle auf einem Intel Core2 Duo CPU E6750 @ 2,66GHz 2,67GHz mit 2GB RAM und dem Betriebssystem Ubuntu 10.04 64-bit durchgeführt. Des Weiteren wurde für CORAL die CADP Version 2009-h sowie die MRMC Version 1.5 benutzt. Untersucht wurde jeweils die time-bounded Reachability. Mit IMCA wurde jeweils sowohl die maximale als auch die minimale Wahrscheinlichkeit berechnet, mit CORAL nur die maximale Wahrscheinlichkeit, da keine Einstellung zur Berechnung der minimalen Wahrscheinlichkeit zur Verfügung stand.

Die Tabelle in Abb. 4.13 zeigt die Testergebnisse der in Kapitel 4.1 vorgestellten Beispiele, wobei das *Erlang*(30, 10) Beispiel gesondert behandelt wird, da es nicht mit CORAL, sondern direkt mit MRMC untersucht wurde.

Zu Beginn werden die Ergebnisse von CORAL und IMCA verglichen. Dabei fällt im ersten Beispiel auf, dass die Werte sich nur minimal unterscheiden, was auf Rundungsfehler zurückgeführt werden kann. Dass der Nichtdeterminismus Einfluss auf das Ergebnis hat, sieht man, aufgrund der relativ großen Ausfallwahrscheinlichkeit des Triggers deutlich, daran, dass sich die minimale und die maximale Wahrscheinlichkeit unterscheiden.

Auch im zweiten Beispiel liefern CORAL und IMCA das gleiche Ergebnis. Hier fällt jedoch auf, dass die minimale und die maximale Wahrscheinlichkeit übereinstimmt. Dies zeigt, dass der Nichtdeterminismus aus dem DFT irrelevant für das Ergebnis ist, was auch an der minimierten IMC in Abbildung 4.3 zu sehen ist.

Beispiel 3 zeigt wie Beispiel 1, abgesehen von Rundungsfehlern, dieselben Ergebnisse für CORAL und IMCA. Dies liegt daran, dass es für die nichtdeterministischen Zustände jeweils eine eindeutig vorzuziehende Wahl gibt, unabhängig von der vergangenen Zeit, wie zuvor erwähnt.

Auch Beispiel 4 zeigt nichts Neues, die Werte sind wieder gleich.

Das Auffällige bei den Werten von Beispiel 5 ist der geringe Unterschied zwischen der

minimalen und der maximalen Wahrscheinlichkeit. Der Grund dafür liegt im Aufbau des Systems. Die Entscheidung, welches Baseisevent bei einem Ausfall von X zuerst ausfällt, spielt nur dann eine Rolle, wenn beide Ersatzteile von E und F schon ausgefallen sind. Die Wahrscheinlichkeit, dass dies jedoch vor einem Ausfall von X geschieht, ist klein.

Auch in der Fallstudie NDPS [3] ist der Unterschied zwischen minimaler und maximaler Wahrscheinlichkeit zwar vorhanden, jedoch sehr gering. Hier liegt der Grund jedoch in der geringen Ausfallwahrscheinlichkeit des Triggers, wodurch der Nichtdeterminismus an Einfluss verliert. Auch ist wieder nur ein geringer, auf Rundungsfehler basierender, Unterschied bei den Ergebnissen von CORAL und IMCA wahrzunehmen, da die optimalen Entscheidungen zeitunabhängig sind.

Die Ergebnisse von Beispiel 7 sind analog zu Beispiel 1 zu sehen, nur dass auch hier der Unterschied zwischen minimaler und maximaler Wahrscheinlichkeit, aufgrund des sehr geringen Ausfallrisikos des Triggers, sehr klein ist. Tabelle 4.10 zeigt als weiteres Beispiel

Abbildung 4.10.: Erreichbarkeitswahrscheinlichkeit und Analysedauer für das $Erlang(30, 10)$ Beispiel

Tool	Error	Erreicht in 1 s	2 s	3 s	4 s	7 s
MRMC	1e-6	0.1321206	0.2969971	0.4004259	0.5842847	0.9784889
IMCA	1e-6	0.1321205	0.2969971	0.4021003	0.6717769	0.9828448
MRMC	1e-6	0.32 s	1.79 s	5.62 s	13.29 s	94.33 s
IMCA	1e-6	242.06 s	242.75 s	298.16 s	265.03 s	304.31 s

die Ergebnisse der IMC aus Abbildung 4.9 [15]. Hierbei ist ein Unterschied zwischen timed und time-abstract Scheduler-Verfahren zu sehen. Während zu den Zeitpunkten 1 und 2 noch kein Unterschied zu sehen ist, tritt er bei Zeitpunkt 3 gering auf, wird zu Zeitpunkt 4 größer und sinkt dann zu Zeitpunkt 7 wieder. Die vergangene Zeit beeinflusst also die optimale Entscheidung. Ab einem bestimmten Zeitpunkt, hier ungefähr bei Zeitpunkt 3, wird es lukrativer, den sicheren langen Weg über die 30 Zustände zu gehen als den unsicheren kurzen Weg. Je mehr Zeit vergeht, desto wahrscheinlicher ist es, dass der Zielzustand zum Zeitpunkt eines lukrativen Wechsels schon erreicht wurde, wodurch der Einfluss des Wechsels abnimmt. Wie in Kapitel 3.2.2 angekündigt, wurde also gezeigt, dass mit timed Scheduling höhere Werte für die maximale Wahrscheinlichkeit erreicht werden können als mit time-abstract Scheduling. Ein Vorteil von IMCA gegenüber CORAL, unter Nutzung von MRMC, ist also, dass man optimale Ergebnisse erhält. Ein Nachteil wird jedoch offensichtlich, wenn man einen Blick auf die Laufzeiten wirft. Während CORAL bei allen Beispielen für die Analyse konstant um die 3 Sekunden benötigt, schwankt IMCA zwischen 3 Sekunden und 27 Sekunden. Um die Abhängigkeit der Laufzeit IMCAs vom zulässigen Error zu verdeutlichen, sind auch jeweils die Laufzeiten mit dem Error $1e - 4$ im Gegensatz zu $1e - 6$ aufgeführt. Dabei fällt auf,

Abbildung 4.11.: Analysedauer NDT3 mit erhöhten Raten in Sekunden

Tool	Error	Rates *1	*10	*100	*1000
CORAL	1e-6	3.140436695	3.152527513	2.693598574	3.172181205
IMCA	1e-4	0.288527	2.42626	23.4721	295.766

Abbildung 4.12.: Analysedauer NDPS mit erhöhten Raten in Sekunden

Tool	Error	Rates *1	*10	*100	*1000
CORAL	1e-6	3.200065895	3.087775700	3.076511918	3.091565868
IMCA	1e-4	0.341836	1.94588	18.8867	185.712

dass durch eine Verringerung des Errors um den Faktor 10 die Laufzeit in etwa um den selben Faktor abnimmt. Dieselbe Wirkung hat eine Veränderung der maximalen Rate. Dies kann beispielhaft den Tabellen 4.11 und 4.12 entnommen werden. Auch ist zu sehen, dass CORAL durch solche Veränderungen keine Einbuße bezüglich der Laufzeit hat.

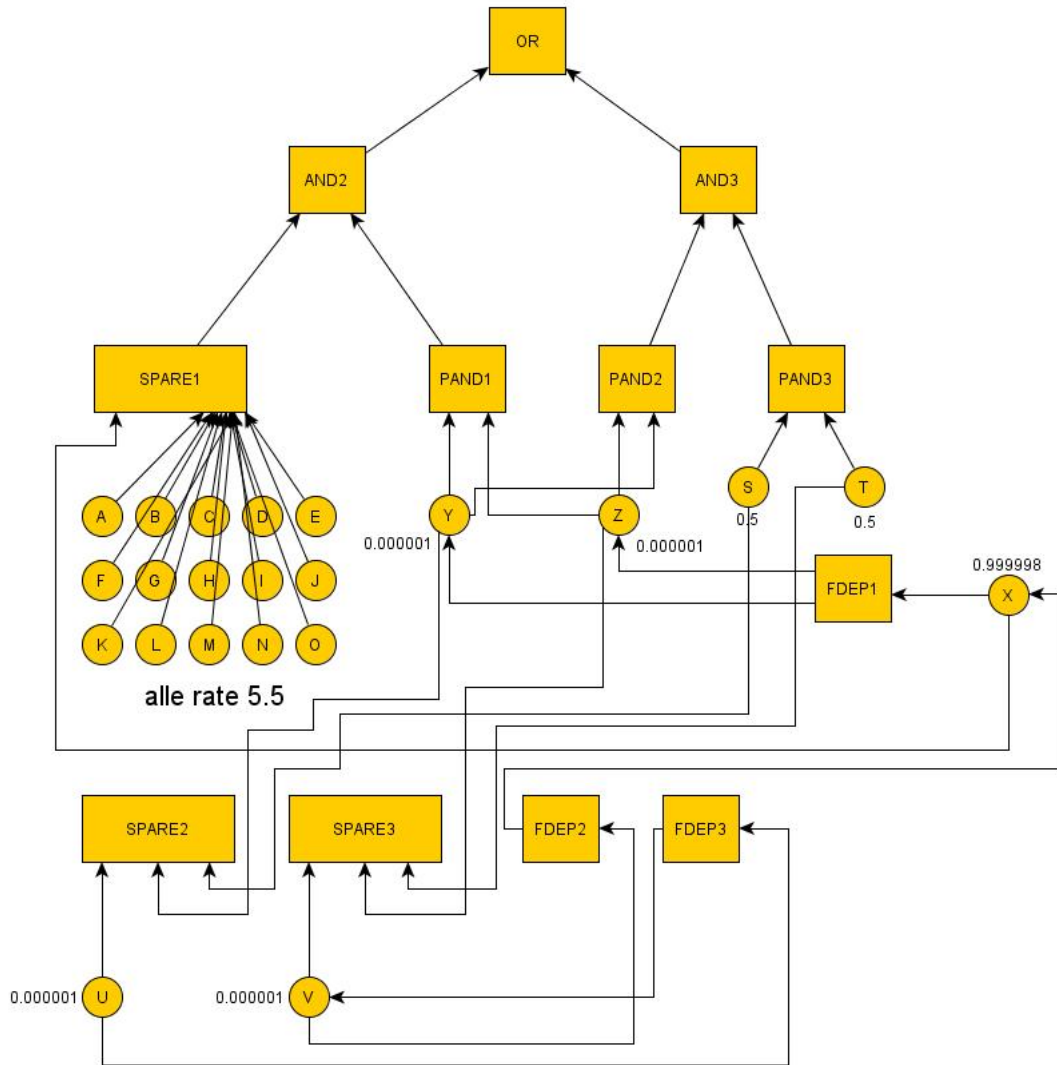
4.2.1. Motivation für den Gebrauch von IMCA

Nun gilt es noch zu zeigen, dass es auch dynamische Fehlerbäume gibt, bei denen der Unterschied zwischen time-abstract und timed Schedulern relevant ist. Ein solches Beispiel ist in Abbildung 4.14 zu sehen. Die Idee dahinter ist die gleiche wie beim *Erlang*(30, 10) Beispiel, also dass es einen sicheren langen Weg gibt, und einen kurzen, bei dem die Erfolgswahrscheinlichkeit bei 50% liegt. Dies ist auch direkt an der entsprechenden IMC zu sehen (Abb. 4.15). Der sichere Weg ist dabei durch das SPARE1 Gate markiert, welches aus 15 Basisevents mit der Ausfallrate 5.5 besteht. Damit SPARE1 also erfüllt ist, müssen alle 15 Basisevents nacheinander ausfallen. Der kurze unsichere Weg ist durch das PAND3 Gate beschrieben, bei dem die zwei Basisevents *S* und *T* die Ausfallrate 0.5 haben. Die Wahrscheinlichkeit, dass *S* vor *T* ausfällt, liegt also bei 50%. Durch den Ausfall von *X* muss jetzt also die nichtdeterministische Entscheidung getroffen werden, ob *Y* oder *Z* zuerst ausfällt. Je nach Wahl wird PAND1 oder PAND2 erfüllt und damit entschieden, ob der lange sichere, oder der kurze unsichere Weg betreten werden soll. Die übrigen Gates und Trigger bewerkstelligen, dass kein Basisevent ausfallen kann, bevor *X*, *V* oder *U* ausgefallen sind. Fällt von diesen Dreien eines aus, so fallen auch sofort die beiden anderen aus.

Abbildung 4.13.: Testergebnisse

Beispiel	Tool	erreicht in 1 Sekunde	erreicht in 2 Sekunden	erreicht in 3 Sekunden	Error	Analyse- Dauer in s
NDT1	CORAL	0.3759696	0.5707830	0.6661130	1e-6	3.2861887
	IMCA	[0.08480026, 0.3759692]	[0.1919152, 0.5707826]	[0.2608310, 0.6661128]	1e-6	3.3149
		[0.08477419, 0.3759886]	[0.1918909, 0.5708057]	[0.2608144, 0.6660940]	1e-4	0.0431831
NDT2	CORAL	0.4482350	0.7814135	0.9227673	1e-6	3.1657471
	IMCA	[0.4482343, 0.4482343]	[0.7814131, 0.7814131]	[0.9227671, 0.9227671]	1e-6	7.42107
		[0.4481940, 0.4481940]	[0.7813925, 0.7813925]	[0.9227590, 0.9227590]	1e-4	0.0947758
NDT3	CORAL	0.2822586	0.5915347	0.7738298	1e-6	3.2211495
	IMCA	[0.1326939, 0.2822584]	[0.3731255, 0.5915346]	[0.5784037, 0.7738297]	1e-6	23.8573
		[0.1326732, 0.2822335]	[0.3731087, 0.5915212]	[0.5783957, 0.7738245]	1e-4	0.27494
NDT4	CORAL	0.5746399	0.7704282	0.8279655	1e-6	3.4389512
	IMCA	[0.1653145, 0.5746396]	[0.3020514, 0.7704280]	[0.3518032, 0.8279654]	1e-6	21.877
		[0.1653074, 0.5746366]	[0.3020515, 0.7704294]	[0.351801, 0.827963]	1e-4	0.253874
NDT5	CORAL	0.0140513	0.0735271	0.1617368	1e-6	3.1009767
	IMCA	[0.01222688, 0.01405121]	[0.06768387, 0.07352693]	[0.1529870, 0.1617366]	1e-6	27.3483
		[0.01221836, 0.01404187]	[0.06766108, 0.07350319]	[0.1529556, 0.1617047]	1e-4	0.338269
NDPS	CORAL	0.0059848	0.0362181	0.0936907	1e-6	3.2499474
	IMCA	[0.005857758, 0.005984725]	[0.03589477, 0.03621780]	[0.09322694, 0.09369012]	1e-6	27.5983
		[0.005851299, 0.005978230]	[0.03587300, 0.03619600]	[0.09318741, 0.09365058]	1e-4	0.303284
NDT6	CORAL	0.1719509	0.3384608	0.4524664	1e-6	3.1034743
	IMCA	[0.1718980, 0.1719507]	[0.3383987, 0.3384607]	[0.4524063, 0.4524664]	1e-6	5.06054
		[0.1718900, 0.1719427]	[0.3383994, 0.3384614]	[0.4524117, 0.4524718]	1e-4	0.0675571

Abbildung 4.14.: Erlang(15,5.5) als DFT



In der Tabelle 4.16 sind die Ergebnisse dieses Beispiels dargestellt. Dabei ist zu sehen, dass ab dem Zeitpunkt 3.0 IMCA eine höhere Wahrscheinlichkeit als CORAL ausgibt. Der maximale Unterschied liegt dabei zu Zeitpunkt 3.5 bei 0.0011728. Die Erklärung dafür, dass eine Differenz besteht, ist dieselbe wie beim *Erlang*(30, 10) Beispiel (zu sehen in 4.9). Ab einem bestimmten Zeitpunkt, wenn die Zeit knapp wird, ist es attraktiver vom sicheren Pfad abzuweichen und sich auf ein Glücksspiel einzulassen. Es fällt jedoch direkt auf, dass die maximale Differenz hier wesentlich kleiner als in dem *Erlang*(30, 10) Beispiel (siehe 4.10) ist. Dies scheint darin begründet, dass CORAL die IMC, bevor er sie an MRMC übergibt, uniformiert, also ihn so verändert, dass alle Zustände die gleiche Gesamtausgangsrate haben. Dadurch arbeitet MRMC mit einem anderen Algorithmus als für nicht-uniformierte CTMDPs [14], durch welchen der Unterschied der Werte scheinbar geringer wird. Deutlich wird dies, wenn man sich zum einen die Werte des Motivationsbeispiels direkt bei MRMC anguckt, ohne dass das Beispiel zuvor uniformiert wurde, und zum anderen einmal das *Erlang*(30, 10) Beispiel uniformiert. Die Ergebnisse in Tabelle Abb. 4.17 bzw. in der letzten Spalte der Tabelle in Abb. 4.16 unterstützen die Annahme. Zur Erklärung, warum das Uniformieren der IMC hier diese Wirkung hat, betrachte man in Abbildung 4.15 Zustand 0, welcher durch die Uniformierung eine Selfloop mit der Rate 83,5 erhält. Der bisherige Vorteil der timed-Scheduler, nach einer bestimmten Zeit die Entscheidung in Zustand 1 zu ändern, wird somit abgeschwächt, da bei time-abstract Schedulingen zwar die vergangene Zeit nicht bekannt ist, wohl aber der bisherige Pfad. Durch die Selfloop kann der Scheduler somit entscheiden, dass ab einer bestimmten Anzahl von Besuchen des Zustands 0 über die Selfloop die Entscheidung geändert wird [1]. Da eine Transition im Schnitt die Dauer $\frac{1}{uni.f.rate}$ hat, kann ein time-abstract Scheduler sogar abschätzen, wie viel Zeit in der Selfloop bereits vergangen ist. Dadurch wird das Ergebnis zwar immer noch nicht optimal, aber der Unterschied wird stark verkleinert. Die Höhe der einheitlichen Gesamtausgangsrate ergibt sich dabei aus der Summe aller Raten der BEs im ursprünglichen DFT und ist damit entsprechend hoch.

Eine Überlegung, die Verbesserung zu umgehen und eine erheblich größere Differenz zu erhalten (kleine Verbesserungen sind durch Variieren der Raten erreichbar), ist es, die Gesamtausgangsrate zu verringern, sodass die Wahrscheinlichkeit, dass die Selfloop durchlaufen wird, sinkt, obwohl einiges an relevanter Zeit vergangen ist. Eine Variante dafür ist alle Raten proportional zueinander zu verringern (Variante 1). Weitere Möglichkeiten sind das Verringern der Raten für den langen (Variante 2) oder kurzen (Variante 3) Weg, bzw. für den langen und den kurzen Weg, aber nicht für den Trigger (Variante 4). Diese Versuche waren jedoch alle nicht von Erfolg gekrönt, wie in Tabelle 4.18 zu sehen ist, wo die Raten jeweils um den Faktor 100 reduziert wurden.

Abbildung 4.15.: Aus Abb. 4.14 entstandene IMC (original und uniformiert)

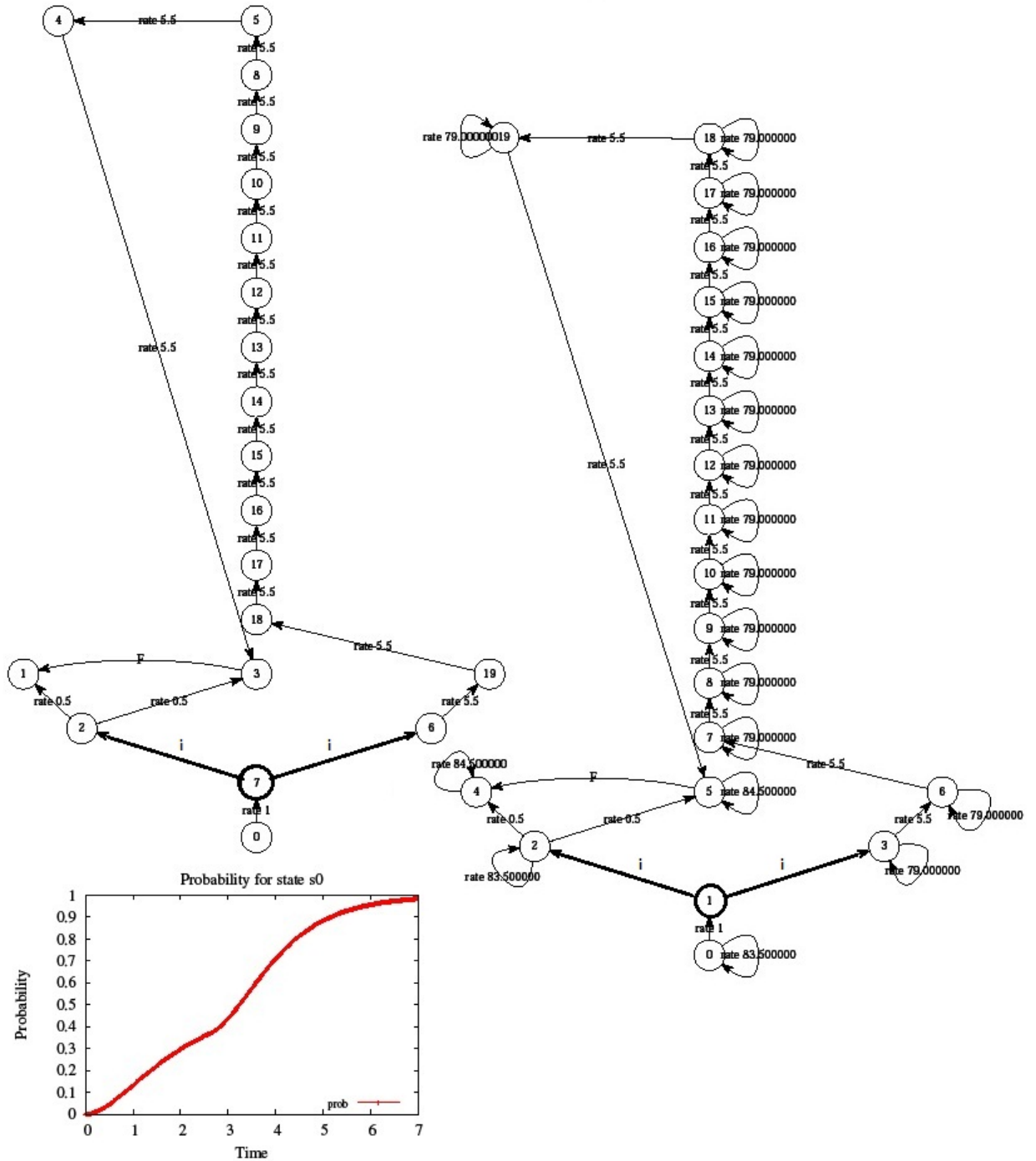


Abbildung 4.16.: Analyseergebnisse für Erlang(15,5.5)-DFT

Erreichbarkeit bis Zeitpunkt	IMCA	CORAL	Differenz	MRMC ohne Uniformierung
0.5	0.04510197	0.0451020	-0.00000003	0.0451020
1.0	0.1321206	0.1321206	0.0000000	0.1321206
1.5	0.2210873	0.2210873	0.0000000	0.2210873
2.0	0.2969971	0.2969971	0.0000000	0.2969971
2.5	0.3563513	0.3563513	0.0000000	0.3563513
3.0	0.4368000	0.4359445	0.0008555	0.4004259
3.5	0.5744817	0.5733089	0.0011728	0.4896407
4.0	0.7094876	0.7083830	0.0011046	0.6580292
4.5	0.8133876	0.8124829	0.0009047	0.7821766
5.0	0.883954	0.8832636	0.0006904	0.8650234
gebrauchte Zeit	73.4	3.6	69.8	4.3

Abbildung 4.17.: Erlang(30,10) mit Uniformierung

Tool	Error	erreicht in 1 Sekunde	erreicht in 2 Sekunden	erreicht in 3 Sekunden	erreicht in 4 Sekunden	erreicht in 7 Sekunden
MRMC	1e-6	0.1321206	0.2969971	0.4020241	0.6713473	0.9827641
IMCA	1e-6	0.1321205	0.2969971	0.4021003	0.6717769	0.9828448

Abbildung 4.18.: Versuche Differenz für Erlang(15,5.5)-DFT zu steigern

Variante	Erreichbarkeit bis Zeitpunkt	IMCA	CORAL	Differenz
1	350	0.574474	0.573309	0.001165
	355	0.588846	0.587674	0.001172
	360	0.603113	0.601943	0.001171
2	265	0.5	0.5	0
	267.5	0.501467	0.501055	0.000412
	270	0.513430	0.513303	0.000127
3	2.2	0.06178938	0.0617123	0.00007708
	2.225	0.06627558	0.0661984	0.00007718
	2.25	0.07096973	0.0708926	0.00007713
4	260	0.46249	0.4624881	0.0000019
	262.5	0.4706376	0.4702553	0.0003823
	265	0.4845634	0.4845051	0.0000583

4.3. Expected Times

Zum Schluss wird noch auf den Erwartungswert für den Ausfallzeitpunkt der DFTs eingegangen.

Definition 3. Für eine IMC $M = (S, Act, \rightarrow, \Rightarrow, s_0)$ beschreibt $EW(s)$ mit $s \in S$ den Erwartungswert für die Zeit, welche von s aus für das Erreichen eines Zustandes in G gebraucht wird. Für Zustände $s \in G$ gilt daher $EW(s) = 0$. Für Zustände $s \in S \setminus G$ mit ausschließlich Markov Transitionen gilt

$$EW(s) = \frac{1}{E(s)} + \sum_{s' \in S} P(s, s') \cdot EW(s').$$

Für Zustände $s \in S \setminus G$ mit mindestens einer internen Transition muss durch den möglicherweise vorhandenen Nichtdeterminismus bei der Berechnung des Erwartungswerts zwischen dem minimalen und dem maximalen Erwartungswert unterschieden werden. Hier gilt

$$EW(s) = \min\{EW(s') \mid \exists s' \in S. s \xrightarrow{i}^* s'\}$$

bzw.

$$EW(s) = \max\{EW(s') \mid \exists s' \in S. s \xrightarrow{i}^* s'\}.$$

Auf Besonderheiten, welche mögliche Kreise aus internen Transitionen betreffen, wird in den Abschnitten 4.3.1 und 4.3.1 eingegangen.

Die Berechnung von $EW(s)$ Zustände mit ausschließlich Markov Transitionen setzt sich dabei aus der Dauer der Verzögerung in s selber, sowie aus der Summe der Erwartungswerte aller möglichen Nachfolgezustände gewichtet nach der Wahrscheinlichkeit, dass genau die Transition zu diesem Zustand gewählt wird.

Im Folgenden wird auf die Berechnung eingegangen, welche von IMCA verwendet wird. Sie basiert auf einer Formel aus [10] und [24]. Dabei wird eine Variante der *value iteration* benutzt. Es werden die zwei Vektoren \vec{v}_i und $\vec{u}_i(s)$ verwendet. In \vec{v}_i werden die aus \vec{u}_{i-1} mittels eines Schrittes berechneten Werte des i -ten value iteration-Schritts gespeichert. \vec{u}_i enthält den maximalen bzw. minimalen Wert aus den über interne Aktionen erreichbaren $\vec{v}_i(s')$. Im Folgenden wird die Berechnung für den maximalen Erwartungswert besprochen und danach kurz auf die Unterschiede bei der Berechnung des minimalen Wertes eingegangen.

4.3.1. Maximaler Erwartungswert

Sei $G \subseteq S$ die Menge der Zielzustände, $Pre(G) \subseteq S$ die Menge aller Zustände von denen ein Pfad nach G existiert, $I \subseteq Pre(G)$ die Menge aller Zustände mit mindestens einer internen Transition und $IS_0 \subseteq I$ die Menge aller Zustände, die Teil eines Kreises aus internen Aktionen sind (siehe Abb. 4.19), bei welchem kein Zustand in G liegt. Dies ist nötig, da bei ihnen der Erwartungswert durch einen Verbleib in dem Kreis ∞ betragen muss. Mit Hilfe von IS_0 kann dies sichergestellt werden. Für $s \in IS_0 \cup (S \setminus Pre(G))$ gilt initial, und damit auch durchgehend, $\vec{v}_0(s) = \infty$, für alle anderen Zustände gilt initial $\vec{v}_0(s) = 0$. Für die weiteren Iterationsschritte $i > 0$ gilt

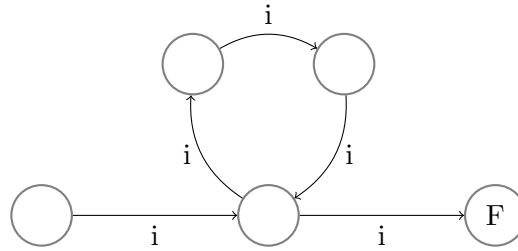
$$\vec{v}_i(s) = \begin{cases} \frac{1}{E(s)} + \sum_{s' \in S} P(s, s') \cdot \vec{u}_{i-1}(s'), & \text{falls } s \in Pre(G) \setminus I \setminus G \\ 0, & \text{falls } s \in G \\ \vec{u}_{i-1}(s), & \text{falls } s \in I \setminus G \end{cases}$$

Des Weiteren gilt für alle $s \in S$

$$\vec{u}_i(s) = \begin{cases} \max\{\vec{v}_i(s') \mid \exists s' \in S. s \xrightarrow{i}^* s'\}, & \text{falls } s \in Pre(G) \setminus IS_0 \setminus G \\ \infty, & \text{falls } s \in IS_0 \\ 0, & \text{falls } s \in G. \end{cases}$$

Dabei beschreibt \xrightarrow{i}^* die reflexive, transitive Hülle von \xrightarrow{i} . Es wird also der maximale, über interne Aktionen erreichbare Wert ausgewählt. Also falls, zum Beispiel, ein Zustand $s' \notin Pre(G)$ oder $s' \in IS_0$ über einen Pfad von internen Aktionen erreichbar ist, so gilt $\vec{u}_i(s) = \infty$. Außerdem gilt offensichtlich für alle $s \notin I$ $\vec{u}_i(s) = v_i(s)$. Sobald ein Fixpunkt, unter Berücksichtigung einer vorher festgelegten Fehlerrate ϵ erreicht ist, also sobald $|\vec{u}_{k-1} - \vec{u}_k| < \epsilon$ gilt, ist der endgültige Erwartungswert zum Erreichen eines Zielzustands $s' \in G$ für jeden Zustand $s \in S$ in $\vec{u}_k(s)$ gefunden.

Abbildung 4.19.: Motivation für IS_0



4.3.2. Minimaler Erwartungswert

Die Berechnung des minimalen Erwartungswerts funktioniert im Prinzip genauso wie die zuvor beschriebene Berechnung des maximalen Erwartungswerts. Vier Punkte gilt es jedoch zu beachten.

Zunächst suchen wir den minimalen Erwartungswert. Das heißt bei der Berechnung der \vec{u}_i wird nicht mehr maximiert, sondern minimiert.

Der zweite Punkt betrifft den Umgang mit Kreisen aus internen Transitionen. Während bei der Maximierung des Erwartungswerts ein solcher Kreis immer betreten und danach nie verlassen wurde, muss bei der Minimierung das Betreten eines solchen Kreises ausgeschlossen werden, da ansonsten keine vernünftige Berechnung möglich wäre. Befindet sich ein Zustand auf einem solchen Kreis und initial gilt $\vec{v}_0(s) = 0 \forall s \in Pre(G)$, dann bleibt der Erwartungswert dieser Zustände durch Verbleiben im Kreis immer bei 0, was jedoch im Allgemeinen falsch ist (siehe Abb. 4.20). Auch das Initialisieren der \vec{v}_0 dieser Zustände auf ∞ verschafft keine Abhilfe, da es möglich wäre, wie in Abb. 4.20, dass die Zustände dann zwar einmal den richtigen minimalen Wert annehmen, dieser danach jedoch im Kreis konstant bleibt, obwohl sich das $\vec{v}_i(s)$, auf welches sich ursprünglich bezogen wurde, in weiteren Iterationen erhöht. Die Lösung liegt darin, den Zutritt zu Kreisen nicht zu gestatten. Dies wird erreicht, indem nicht mehr die \vec{v}_i Werte aller durch interne Transitionen erreichbaren Zustände verglichen werden, sondern nur noch die Werte von Zuständen aus G oder von Zuständen ohne interne Transitionen. Dadurch wird ein Suchen nach solchen Kreisen und damit die Menge IS_0 überflüssig.

Der dritte Punkt erfordert eine Unterscheidung zwischen Zuständen $s \in I$ und $\bar{s} \notin I$. Während bei der Berechnung der \vec{u}_i für die $\bar{s} \notin I$ weiterhin $\vec{u}_i(\bar{s}) = \vec{v}_i(\bar{s})$ gilt, muss den $s \in I$ auf anderem Wege die Möglichkeit genommen werden, den eigenen Wert als Minimum zu wählen, da dieser zu Beginn 0 beträgt und somit immer gewählt werden würde. Dies wird erreicht, indem bei der Berechnung der \vec{u}_i der Übergang \rightarrow^* in \rightarrow^+ geändert wird.

Außerdem brauchen bei der Berechnung der \vec{u}_i nur noch die Nachfolger betrachtet werden, von denen G aus erreicht werden kann, da bei den anderen Zuständen $\vec{v}_i(s') = \infty$ gilt, und sie somit nie gewählt werden würden.

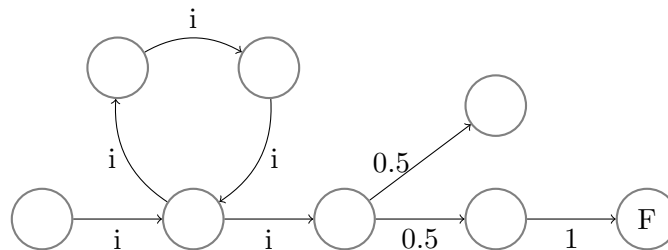
Durch die zuvor erläuterten Punkte kann die Berechnung des minimalen Erwartungswerts angepasst werden. Initial gilt $\vec{v}_0(s) = 0$ für alle $s \in Pre(G)$ und $\vec{v}_0(s) = \infty$ sonst. Weiter gilt für $i > 0$

$$\vec{v}_i(s) = \begin{cases} \frac{1}{E(s)} + \sum_{s' \in S} P(s, s') \cdot \vec{u}_{i-1}(s'), & \text{falls } s \in Pre(G) \setminus I \setminus G \\ 0, & \text{falls } s \in G \\ \vec{u}_{i-1}(s), & \text{falls } s \in I \setminus G \end{cases}$$

Die Berechnung der \vec{u}_i wird angepasst zu

$$\vec{u}_i(s) = \begin{cases} \min\{v_i(s) \mid \exists s' \in G \cup (Pre(G) \setminus I).s \rightarrow^+ s'\}, & \text{falls } s \in I \setminus G \\ 0, & \text{falls } s \in G \\ \vec{v}_i(s), & \text{falls } s \notin I \cup G. \end{cases}$$

Abbildung 4.20.: Motivation für IS_0 bei der Minimierung



4.3.3. Testergebnisse für Erwartungswerte

Es wurden dann noch für die Beispiele die Erwartungswerte für den Ausfall der Systeme berechnet. Dabei ist zu erwarten, dass die Ausfallwahrscheinlichkeit zu dem errechneten Zeitpunkt bei ungefähr 0.5 liegt, was auch bei allen Beispielen ungefähr zutrifft. Bei den Beispielen NDT1, NDT4, NDT5 und dem Motivationsbeispiel zur Benutzung IM-CAs ist der maximale Erwartungswert ∞ . Dies ist der Fall, da die Wahrscheinlichkeit, dass das System nicht ausfällt, also dass nie ein Zielzustand erreicht wird, größer als 0 ist. Die größte Abweichung von 0.5 liegt nach oben mit 0.6762133 bei NDT6 und nach unten mit 0.3928860 bei NDT5 vor. In der 3. Spalte der Tabelle wurde die maximale

Erreichbarkeitswahrscheinlichkeit zum Zeitpunkt der minimalen Erwartungszeit ermittelt. In Spalte 4 dementsprechend die minimale Wahrscheinlichkeit für den maximalen Erwartungswert.

Abbildung 4.21.: Expected Times

Beispiel	Erwartungswerte in Sekunden	benötigte Zeit in Sekunden	max. Wahrsch. bei min. EW	min. Wahrsch. bei max. EW
NDT1	[1.25, ∞)	0.0096471	0.437903	-
NDT2	1,36412	0.0143501	0.5980116	0.5980116
NDT3	[2.15511, 3.20269]	0.0371397	0.6273579	0.6125696
NDT4	[0.823526, ∞)	0.0289492	0.5107208	-
NDT5	[5.69444, ∞)	0.0542269	0.3928860	-
NDPS	[9.17677, 9.18126]	0.0749286	0.6062061	0.6062267
NDT6	[7.26436, 7.26513]	0.0121065	0.6762133	0.6761989
Erlang(15,5.5) DFT	[2,3.72727]	0.0579208	0.296997	0.37667

5. Nutzen der Kreislosigkeit

In diesem Kapitel wird darauf eingegangen, wie die Kreislosigkeit der entstandenen IMCs zur Verbesserung der gebrauchten Analysezeit benutzt werden kann. Dabei wird zum einen kurz betrachtet, welche Schritte im Vergleich zur bisherigen Berechnung nicht mehr nötig sind (Abschnitt 5.1) und zum anderen auf eine mögliche Minimierung des Zustandsraumes der entstehenden IMC durch eine Alternative zur schwachen Bisimulation eingegangen (Abschnitt 5.2).

5.1. Algorithmenanpassungen

In Kapitel 4.3 und 3.2.3 wird auf die Notwendigkeit zur Berechnung möglicher Kreise von internen Transitionen eingegangen und warum es wichtig ist, diese vor der Berechnung zu ermitteln. Diese Notwendigkeit entfällt bei der Analyse von DFTs, da solche Kreise nicht auftreten können. Es liegt also nahe, das Suchen nach diesen aus dem Algorithmus zu entfernen. Um die Wirkung dessen zu untersuchen, wurde die Dauer zur Berechnung des maximalen Erwartungswerts für eine Kette mit 2001 Zuständen und gleichmäßig wachsenden Raten von 0,1 bis 200 mit und ohne Kreissuche ermittelt. Dabei ergaben sich nach jeweils zehn Testläufen Durchschnittswerte von 651,2 Sekunden für die Berechnung mit Kreissuche und 659,5 für selbige ohne Kreissuche. Es ist also kein Vorteil beim Entfernen der Kreissuche zu erkennen, im Gegenteil ist sogar der Durchschnittswert des Algorithmus mit Kreissuche kleiner, was jedoch eher an einer natürlichen Schwankung der Werte liegt. Der Grund für den nicht vorhandenen, merkbaren Zeitgewinn ist, dass das Ermitteln solcher Kreise durch eine einfache Tiefensuche geschieht, welche kaum Zeit kostet.

5.2. Minimierung des Zustandsraumes

Wie bereits erwähnt tritt auch bei der Analyse von DFTs das *state-explosion problem* auf. Um den Effekt abzuschwächen, minimiert CORAL das Ergebnis jeder parallelen Komposition zweier IMCs, wobei es auf schwache Bisimulation zurückgreift (siehe Kapitel 3). Eine Alternative wird in [17], [16] und [19] vorgestellt. Dabei handelt es sich

um eine Methode der Reduzierung von Darstellungen für azyklische Phasentypverteilungen. Wie bereits in Kapitel 2 beschrieben, wird die Ausfallwahrscheinlichkeit der BEs als Exponential- oder Phasentypverteilung angegeben. Dabei ist anzumerken, dass eine Exponentialverteilung im Prinzip eine Phasentypverteilung mit genau einer Phase (sowohl im aktiven, als auch im schlafenden Zustand) ist. Die verwendbaren Phasentypverteilungen sind dabei azyklisch, es handelt sich also um azyklische Phasentypverteilungen (APH-Verteilungen).

APH-Verteilungen können als CTMCs mit genau einem absorbierenden Zustand dargestellt werden, bei denen von allen anderen Zuständen aus ein Weg zu diesem existiert. Die Transitionsfunktion einer CTMC kann dabei durch eine Matrix Q ausgedrückt werden. Dabei stellt für den Zustandsraum $S = \{1, \dots, n+1\}$, mit $n+1$ als absorbierender Zustand, $Q(i, j)$, $i \neq j$, die Wahrscheinlichkeit dar, in einem unendlich kleinen Zeitschritt von Zustand i nach Zustand j zu kommen. Anders ausgedrückt ist $Q(i, j)$ die Beschriftung der Kante von Zustand i zu Zustand j in der entsprechenden CTMC. Des Weiteren gilt $Q(i, i) = -\sum_{j \in S, j \neq i} Q(i, j)$. Für APH-Verteilungen gilt außerdem, dass eine triangulierte Version der Matrix Q existieren muss. Sei A nun eine $n \times n$ Submatrix von Q bei welcher die Spalte und Zeile, welche den absorbierenden Zustand betreffen, entfernt wurde. Zudem sei $\vec{\alpha}$ ein Zeilenvektor, welcher den initialen Wahrscheinlichkeiten der Zustände 1 bis n entspricht und \vec{e} ein Spaltenvektor passender Dimension, dessen Einträge alle 1 sind. Eine Zufallsvariable Z ist dann nach einer azyklischen Phasentypverteilung mit der Darstellung (α, A) verteilt, wenn die Verteilungsfunktion durch

$$F(t) = Pr(Z \leq t) = \begin{cases} 1 - \vec{\alpha} e^{At} \vec{e}, & \text{wenn } t \in \mathbb{R}_{\geq 0}, \\ 0, & \text{sonst.} \end{cases}$$

gegeben und A triangulierbar ist. Die Ausfallwahrscheinlichkeit der BEs kann durch eine APH-Verteilung dargestellt werden, während dies für DFT-Gates im allgemeinen nicht gültig ist, da zum Beispiel beim PAND die Möglichkeit besteht, dass das Gate durch Ausfälle der BEs in einer bestimmten Reihenfolge nicht mehr ausfallen kann und somit zwei unterschiedliche absorbierende Zustände existieren. Glücklicherweise trifft das jedoch nur auf die dynamischen Gates zu. Im Folgenden wird gezeigt, dass die Ausfallwahrscheinlichkeit eines FT Top Elements mit APH-Verteilungen für seine BEs, ebenfalls durch eine APH-Verteilung dargestellt werden kann.

Dafür nehmen wir zunächst an, dass es m BEs mit entsprechenden CTMCs X_1, \dots, X_m zur Darstellung der APH-Verteilung gibt. Dann können wir durch die Bildung des Kreuzprodukts dieser CTMCs eine CTMC X bilden, welche die Position der einzelnen CTMCs X_1, \dots, X_m angibt. Des Weiteren können wir eine Funktion $\vec{b} : S \rightarrow \{0, 1\}^m$ bilden, welche für jeden Zustand in X angibt, welche der zugrundeliegenden X_1, \dots, X_m einen absorbierenden Zustand erreicht hat, und damit welches BE ausgefallen ist. Das Ausfallverhalten des FTs kann, durch Abwesenheit dynamischer Gates anders als für DFTs, als boolesche Funktion $f : \{0, 1\}^m \rightarrow \{0, 1\}$ angegeben werden. Dementsprechend gibt $f(\vec{b}(s))$ an, ob der FT in Zustand s ausgefallen ist. Man kann zeigen, dass alle Zustände,

für die $f(\vec{b}(s)) = 1$ (wobei mit 1 der Ausfall dargestellt wird) gilt, ohne weiteres zu einem Zustand zusammengefasst werden können und auch die Kreislosigkeit erhalten bleibt. Damit ist gezeigt, dass auch der FT durch eine APH-Verteilung dargestellt werden kann.

Weiterhin beschreibe $Bi(\lambda_1, \dots, \lambda_n)$ eine Matrix der Form

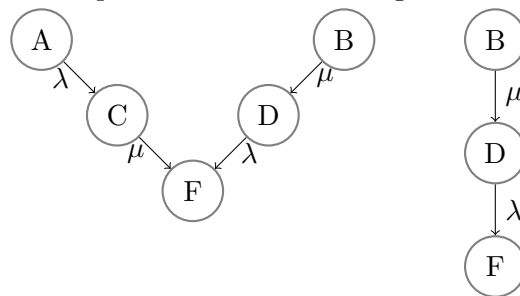
$$\begin{pmatrix} -\lambda_1 & \lambda_1 & 0 & \dots & 0 \\ 0 & -\lambda_2 & \lambda_2 & \dots & 0 \\ 0 & 0 & -\lambda_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -\lambda_n \end{pmatrix},$$

welche *ordered bidiagonal representation* (OBR) genannt wird.

Für jede APH Darstellung (α, A) mit den Eigenwerten $-\lambda_1, \dots, -\lambda_n$ (oBdA: $\lambda_n \geq \dots \geq \lambda_1 > 0$) von A existiert eine eindeutige Darstellung $(\beta, Bi(\lambda_1, \dots, \lambda_n))$, sodass beide Darstellungen dieselbe azyklische Phasentypverteilung repräsentieren. Zur effizienten Umwandlung von allgemeinen APH Darstellungen in OBRs wird der in [11] beschriebene *spectral polynomial algorithm* (SPA) verwendet, welcher in $O(n^3)$ (n =Größe der APH Darstellung) arbeitet.

Die Minimierung der APH Darstellungen läuft dann so ab, dass die Darstellung zunächst in eine OBR gewandelt wird, bei welchem dann überflüssige Zustände entfernt werden können. Das genaue Vorgehen ist in [16] und [20] nachzugucken. Die Idee dahinter ist in Abbildung 5.1 zu sehen (aus [18]). Während durch schwache Bisimulation hier keine Ersparnis möglich wäre, werden die Zustände A und B hier zusammengefasst, da sie lineare Kombinationen voneinander sind.

Abbildung 5.1.: Beispiel für die Minimierung von APH-Verteilungen



In CORAL kann diese Art der Minimierung eingesetzt werden, indem in den DFTs nach Subtrees ohne dynamische Gates, also FTs, gesucht wird [8], welche, wie zuvor

beschrieben, durch APH-Verteilungen darstellbar sind. Bei diesen wird die Minimierung per schwacher Bisimulation dann durch den zuvor beschriebenen Algorithmus ersetzt. Dass dies enorme Auswirkungen auf den Zustandsraum haben kann, zeigen Ergebnisse aus [16], welche in Tabelle 5.2 wiedergegeben sind. Dabei wurde zunächst ein DFT bestehend aus einem 2/3 VOTING-Gate ohne Änderung und mit der neuen Minimierungsmöglichkeit untersucht. Im ersten Fall hatte die entstehende CTMC 21 Zustände und 41 Transitionen, im zweiten Fall 14 Zustände und 22 Transitionen. Hier erscheint der Unterschied noch nicht besonders groß. Setzt man jedoch einmal mehrere dieser VOTING-Gates in ein größeres System ein, und vergleicht es mit den Ergebnissen nach Ersetzen der Gates durch jeweils ein BE mit derselben APH-Verteilung und nach Minimieren dieser durch obiges Verfahren, so erhält man die Werte aus Tabelle 5.2.

Abbildung 5.2.: Resultate aus [16]

Minimierung	# Zustände	# Transitionen	Zeit (s)	Ausfallwars.
Bisimulation	1777955	21895068	14047.99	0.209
APH-Min	507067	5010000	367.71	0.209

6. Fazit

In dieser Arbeit wurde die Analyse bezüglich der Zuverlässigkeit von DFTs mittels CORAL genauer untersucht. Ein besonderes Augenmerk galt dabei den Fällen, in denen Nichtdeterminismus auftritt. Deren Analyse wird aktuell mittels MRMC durchgeführt, welches die geforderte Berechnung der Erreichbarkeitswahrscheinlichkeit in CTMDPs basierend auf time-abstract Schedulern ermittelt. Zunächst wurde der Nichtdeterminismus anhand mehrerer DFTs näher untersucht und einige Aspekte dessen betrachtet.

Den Kern dieser Arbeit bildete jedoch der Vergleich MRMCs mit IMCA, einem Tool, bei welchem die Berechnung der Erreichbarkeitswahrscheinlichkeit in IMCs auf timed Schedulern basiert. Erwartet wurde, dass durch diesen Unterschied für bestimmte Beispiele ein deutlicher Unterschied in den ermittelten Werten liegt. Dies traf auch zu und wurde anhand eines Beispiels aus [15] erfolgreich gezeigt. Aufgrund dieser Ergebnisse war die drastische Reduzierung des Unterschieds bei der Analyse eines entsprechenden DFTs mit CORAL unter der Nutzung MRMCs bzw. IMCAs überraschend. Es wurde festgestellt, dass dies durch die in CORAL durchgeführte Uniformierung der IMCs vor der Übergabe an MRMC begründet werden kann.

Da weiterhin, wie erwartet, ein enormer Unterschied der Laufzeit zugunsten der Nutzung MRMCs festgestellt werden konnte, spricht mehr für eine weitere Benutzung MRMCs. Einen Vorteil, den die Nutzung IMCAs allerdings hat, ist die beliebig genaue Approximation des optimalen Ergebnisses, was mit dem bisherigen Vorgehen CORALs unmöglich bleibt. Sobald es also wichtig ist, sehr genau zu arbeiten, ist IMCA im Vorteil. Interessiert jedoch zum Beispiel nur die Antwort auf die Frage „Ist die Wahrscheinlichkeit eines Ausfalls innerhalb eines Zeitraums größer als 50%“, so sollte, zumindest zunächst, eine Analyse mit MRMC durchlaufen werden. Liefert diese dann schon einen Wert über 50%, ist eine Berechnung mit IMCA nicht mehr nötig. Andernfalls kommt es wieder darauf an, wie genau das gewünschte Ergebnis sein soll.

Zuletzt wurde noch auf einen Algorithmus eingegangen, wodurch die IMCs stärker als mittels schwacher Bisimulation minimiert werden können. Die Umsetzung des Algorithmus aus [19] in CORAL kann als Aspekt zukünftiger Untersuchungen dienen.

A. CD mit Quellcode

siehe CD

Literaturverzeichnis

- [1] C. Baier, H. Hermanns, J.-P. Katoen, B. R. Haverkort *Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes* Theoretical Computer Science 345 (1), pp. 2-26, 2005
- [2] C. Baier, H. Hermanns, J.-P. Katoen, B. R. Haverkort *Model-checking algorithms for continuous Markov chains* IEEE TSE 29, pp. 524-541, 2003
- [3] H. Boudali, P. Crouzen, M. Stoelinga *A Rigorous, Compositional, and Extensible Framework for Dynamic Fault Tree Analysis* IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 2, pp. 128-143, 2010
- [4] T. Chen, T. Han, J.-P. Katoen, A. Mereacre *Computing maximum reachability probabilities in Markovian timed automata* Tech. Rep. AIB-2010-06, 2010
- [5] D. Coppit, K. J. Sullivan, J. V. Dugan *Formal semantics of models for computational engineering: A case study on dynamic fault trees* Proceeding of the International Symposium on Software Reliability Engineering, pp. 270-282. IEEE, 2000
- [6] O. Coudert, J. Madre *Fault Tree Analysis: 10²⁰ Prime Implicants and Beyond* Annual Proceedings on Reliability and Maintainability Symposium, 1993 and IEEE Computer Society, pp. 240-245, 1993
- [7] J. B. Dugan, S. J. Bavuso, M. A. Boyd *Dynamic fault-tree models for fault-tolerant computer systems* IEEE Transactions on Reliability, 41(3): pp. 363-377, 1992
- [8] Y. Dutuit, A. Rauzy *A linear-time algorithm to find modules of fault trees* IEEE Transactions on Reliability, vol. 45(3), pp. 422-425, 1996
- [9] H. Garavel, F. Lang, R. Mateescu, W. Serwe *CADP 2006: A Toolbox for*

- the Construction and Analysis of Distributed Processes* Proc. 19th Intl. Conf. Computer Aided Verification (CAV 2007), pp. 158-163, Volume 4590 of LNCS. Edited by W. Damm, H. Hermanns, Berlin: Springer-Verlag, 2007
- [10] D. Guck, J.-P. Katoen, M. R. Neuhäuser *IMCA: An Interactive Markov Chain Analyzer* unveröffentlichte Notiz, 2011
- [11] Q. M. He, H. Zhang *Spectral polynomial algorithms for computing bi-diagonal representations for phase type distributions and matrix exponential distributions* Stochastic Models, vol. 2, no. 2, pp. 289–317, 2006
- [12] H. Hermanns *Interactive Markov Chains: The Quest for Quantified Quality* volume 2428 of LNCS, Heidelberg: Springer-Verlag Berlin, 2002
- [13] H. Hermanns, J.-P. Katoen *The How and Why of Interactive Markov Chains* Formal Methods for Components and Objects (FMCO), pp. 311–337. Volume 6286 of LNCS, Springer-Verlag, 2010
- [14] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, D. N. Jansen *The ins and outs of the probabilistic model checker MRMC* Quantitative Evaluation of Systems (QEST), pp. 167–176. IEEE CS Press, 2009
- [15] M. R. Neuhäuser *Model Checking Nondeterministic and Randomly Timed Systems* PhD Thesis, RWTH Aachen University / University of Twente, 2010
- [16] M. R. Pulungan *Reduction of Acyclic Phase-Type Representations* <http://scidok.sulb.uni-saarland.de/volltexte/2009/2431/>, PhD thesis, Universität des Saarlandes, Saarbrücken, 2009
- [17] M. R. Pulungan, P. Crouzen *Acyclic phase-type distributions in fault trees* Proceedings of the International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS), 2009
- [18] Folien zu [17], zu finden unter <http://webspn.hit.bme.hu/pmccs2009/slides/pmccs2009.pdf>
- [19] M. R. Pulungan, H. Hermanns *A method for reducing acyclic phase-type representations* AVACS Technical Report No. 32, SFB/TR 14 AVACS, Juli 2007

- [20] M. R. Pulungan, H. Hermanns *Acyclic minimality by construction - almost* Proceedings of QEST, pp.63-72, 2009
- [21] K. A. Reay, J. D. Andrews *A Fault Tree Analysis Strategy Using Binary Decision Diagrams* Reliability Engineering & System Safety, Vol. 78, No. 1, pp. 45-56, 2002
- [22] W. E. Vesely, F. F. Goldberg, N. H. Roberts, D. F. Haasl *Fault Tree Handbook* NUREG-0492. Technical report, NASA, 1981
- [23] L. Xing, J. B. Dugan, B. A. Morrissette *Efficient Reliability Analysis of Systems with Functional Dependence Loops* Maintenance and Reliability, pp. 65-69, 3(43), 2009
- [24] L. Zhang, M. R. Neuhäuser *Model Checking Interactive Markov Chains* Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pp. 53-68. Volume 6015 of Lecture Notes in Computer Science. Springer, 2010
- [25] <http://www.mrmc-tool.org/>
- [26] <http://www.inrialpes.fr/vasy/cadp/>
- [27] <http://www.inrialpes.fr/vasy/cadp/man/bcg.html>
- [28] http://www.inrialpes.fr/vasy/cadp/man/bcg_read.html#sect19
- [29] http://www.inrialpes.fr/vasy/cadp/man/bcg_transient.html
- [30] <http://moves.rwth-aachen.de/imca/>