Implementation von Kriterien zur Erkennung und Einstufung von Feature–Interactions

von
Dirk Barthel

9. Dezember 1996

Zusammenfassung

In dieser Arbeit wird die Entwicklung eines Werkzeugs dargestellt, das zur Unterstützung der Analyse von Feature-Interaktionen in Intelligenten Netzwerken dient.

Das Werkzeug führt dazu auf einer Estelle-Spezifikation eines Intelligenten Netzwerkes eine spezielle Form einer Erreichbarkeitsanalyse durch. Das Ziel dieser Analyse ist die Erkennung aller Situationen im Erreichbarkeitsgraphen mit potentiellen Feature-Interaktionen anhand bestimmter vorgegebener Kriterien.

Es werden zunächst die theoretischen Grundlagen der eingesetzten Kriterien und des benutzten Analyseverfahrens dargestellt. Besonderes Augenmerk liegt dabei auf den eingesetzten Methoden zur Vermeidung einer Zustandsexplosion bei der Erreichbarkeitsanalyse. Es zeigt sich in diesem Zusammenhang, daß die starke Nebenläufigkeit in verteilten Systemen ein Haupthindernis bei der Analyse ist. Danach wird der interne Aufbau und der praktische Einsatz des in C++ implementierten Werkzeugs erläutert.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die vorliegende Arbeit ist Bestandteil eines Projekts zur Entwicklung von Methoden und Werkzeugen zur Erkennung und Auflösung von **Feature–Interaktionen** in **Intelligenten Netzwerken** (**IN**[1]).

Die Feature–Interaktionen können entstehen, wenn zu einem bereits existierenden System neue Features hinzugefügt werden, so daß alte und neue Features miteinander interagieren und sich möglicherweise gegenseitig stören können.

Durch den Einsatz einer formalen Beschreibungstechnik (**FDT**[2]) ist es möglich, das Problem der Feature–Interaktionen in Intelligenten Netzwerken auf einer abstrakten und formalen Ebene zu behandeln.

Auf dieser Ebenen kann das Problem der Feature–Interaktionen durch folgende Fragen charakterisiert werden:

- *Wie können Feature–Interaktionen erkannt werden ?*
- *Wie können die erkannten Feature–Interaktionen entsprechend ihrer „Gefährlichkeit“ klassifiziert werden ?*
- *Wie können Feature–Interaktionen aufgelöst werden ?*

Zur Beantwortung dieser Fragen ist zunächst eine formale Definition der Begriffe *Feature* und *Feature–Interaktion* notwendig. Diese formalen Definitionen und *Kriterien zur Erkennung und Einstufung* von Feature–Interaktionen sind in [Bre95] und [Bre96] zu finden.

[1] **I**ntelligent **N**etwork
[2] **F**ormal **D**escription **T**echnique

Eine Methode zur Auflösung von Feature–Interaktionen mittels *Präzedenzen* ist für die FDT Estelle in [BrGo94a] und [BrGo94b] entwickelt worden. Dabei wird durch Einführung von Präzedenzen zwischen Features eine Vorrangbeziehung für Estelle–Transitionen definiert, so daß Indeterminismus zwischen Transitionen verschiedener Features verhindert werden kann.

1.1 Aufgabenstellung

Ziel dieser Arbeit ist, die in [Bre95] entwickelten Kriterien zu implementieren, so daß die Analyse eines Intelligenten Netzwerkes auf Feature–Interaktionen unterstützt werden kann. Als Spezifikationssprache für die IN wird die FDT Estelle benutzt.

Die Kriterien erfordern eine **Erreichbarkeitsanalyse** auf den Hauptzuständen der Estelle–Module und eine **Datenabhängigkeitsanalyse** auf bestimmten anderen Elementen (zum Beispiel Variablen) des in Estelle spezifizierten Systems. Diese Analysen sollen anhand eines **Erreichbarkeitsgraphen** durchgeführt werden.

Die Knoten des gerichteten Graphen repräsentieren die jeweiligen Hauptzustände der einzelnen Estelle–Module und die benannten, gerichteten Kanten die Estelle–Transitionen, die beim Übergang von einem Knoten zum nächsten benutzt werden.

Die Unterstützung der Analyse besteht darin, *Berechnungsfolgen* im Erreichbarkeitsgraphen zu finden, welche die Kriterien für eine Feature–Interaktion erfüllen. Diese Situationen sind dann mit Hilfe der Kriterien aus [Bre95] entsprechend ihrer „Gefährlichkeit" einzustufen und schließlich als Ergebnis der Analyse auszugeben.

Für die Einstufung der Feature–Interaktionen ist eine Datenabhängigkeitsanalyse erforderlich. Dabei ist zu analysieren, welche Elemente (wie zum Beispiel Variablen, Nachrichten, Nachrichtenparameter, etc.) eines Features durch ein anderes Feature beeinflußt werden können. Dazu ist es nicht notwendig, Werte für die einzelnen Elemente zu berechnen, sondern es genügt die *Tatsache der Beeinflussung* durch ein anderes Feature.

Weiterhin soll mit Hilfe des Werkzeugs überprüft werden, ob durch die Auflösung von Feature–Interaktionen durch Einführung von Präzedenzen neue Feature–Interaktionen mit anderen Features enstanden sind.

Durch die Hinzunahme von Informationen über Indeterminismen zwischen Transitionen können die Ergebnisse einer Analyse verbessert werden. Diese

Informationen liefert ein bereits existierendes Werkzeug **CONFINE**[3] zur Analyse von Feature–Interaktionen in Estelle–Spezifikationen. CONFINE analysiert eine Estelle–Spezifikation modullokal auf das *mögliche* Vorhandensein von Indeterminismen und Überlappungen zwischen den Transitionen[4]. Zudem kann CONFINE als *sicher* nicht ausführbar erkannte Transitionen aus der Spezifikation entfernen.

Ein Unterschied zwischen CONFINE und dem in dieser Arbeit beschriebenen Werkzeug ist, daß keine Erreichbarkeitsanalyse mittels CONFINE durchgeführt wird. Dadurch können Feature–Interaktionen, die nicht auf Indeterminismus zwischen Transitionen basieren, nicht erkannt werden[5].

Ein weiterer Unterschied zu CONFINE besteht darin, daß CONFINE die Analyse auf Indeterminismus zwischen Transitionen *verschiedener* Estelle–Module nicht unterstützen kann, während dies durch das hier vorgestellte Werkzeug mittels einer Suche im globalen Erreichbarkeitsgraphen möglich wird.

CONFINE liefert dennoch wertvolle Hinweise zur Beschleunigung des Analyseverfahrens und zur Verbesserung der Ergebnisses einer Analyse mit dem hier beschriebenen Werkzeug. Diese Hinweise sind jedoch nicht prinzipiell notwendig für die Analyse.

1.2 Gliederung

In Kapitel 2, „*Grundlagen*“, werden die von J. Bredereke in [Bre95] entwickelten theoretischen Grundlagen für die Formalisierung einer Feature–Interaktion und für die Erkennung bzw. Einstufung von Feature–Interaktionen vorgestellt.

Die weiteren Abschnitte des Kapitels behandeln Randbedingungen an das Werkzeug, welche möglichen neuen Feature–Interaktionen durch das Einführen von Präzedenzen entstehen können und schließlich die Ergebnisse einer Analyse mittels CONFINE.

In Kapitel 3 „*Konzepte und Algorithmen der Analyse*“, werden die wesentlichen Konzepte und Algorithmen der Implementation vorgestellt, die der Analyse zugrunde liegen.

[3] reCOgnition of Feature Interactions in intelligent Networks with Estelle

[4] Eine Überlappung ist eine Situation, in der Indeterminismus zwischen zwei Transitionen durch unterschiedliche Prioritäten verhindert wird.

[5] siehe Basiskriterium 1 in Abschnitt 2.3.1

Zunächst wird der Aufbau des Erreichbarkeitsgraphen und des sogenannten **Transitionsgraphen** vorgestellt. Letzterer repräsentiert eine andere Sichtweise auf den Erreichbarkeitsgraphen und wird zur Steigerung der Laufzeiteffizienz bei der Analyse eingesetzt. Weiterhin wird das eigentliche Analyseverfahren beschrieben und eine Abschätzung für die Laufzeitkomplexität des Verfahrens gegeben. Die Bedeutung der Nebenläufigkeit in Bezug auf den Aufbau des Transitionsgraphen und auf die Analyse hat, wird in einem weiteren Abschnitt des Kapitels behandelt. Außerdem werden Anforderungen an einen Estelle–Spezifikationsstil zur Erleichterung der Analyse entwickelt und eine zu Estelle ähnliche, textuelle Form der Spezifikation vorgestellt. Diese Form enthält nur die für den Aufbau des Transitionsgraphen und für die Durchführung der Analyse unbedingt notwendigen Informationen.

In Kapitel 4 „*Details der Implementation*“ wird der praktische Einsatz, die interne Struktur und mögliche Erweiterungen des Werkzeuges beschrieben.

Schließlich folgt noch eine kurze Zusammenfassung der Ergebnisse der Arbeit in Kapitel **Kapitel 5**, „*Zusammenfassung und Ausblick*“.

Kapitel 2

Grundlagen

Ziel dieser Arbeit ist die Unterstützung der Analyse einer Estelle–Spezifikation auf Feature–Interaktionen.

Die theoretische Grundlage für das in dieser Arbeit vorgestellte Analyseverfahren bildet die von J. Bredereke in [Bre95] entwickelte **Formalisierung einer Feature–Interaktion** für Intelligente Netzwerke und die aus der Formalisierung abgeleiteten Feature–Interaktions–Kriterien (**FI–Kriterien**).

Während die in [Bre95] entwickelten FI–Kriterien für endliche, erweiterte Automaten (**EFSA**[1]) definiert sind, wird in dieser Arbeit ihre Umsetzung und Anpassung für die FDT Estelle beschrieben.

Dabei wird vorausgesetzt, daß Intelligente Netzwerke in einen Kerndienst (**Basissystem**) und darauf aufbauende Erweiterungen bzw. neue Leistungsmerkmale (**Features**) gegliedert werden kann. ([ITU93]).

Zum besseren Verständnis werden in Abschnitt 2.1 einige später benötigte Grundbegriffe kurz erläutert. In Abschnitt 2.2 wird dann die in [Bre95] entwickelte Formalisierung einer Feature–Interaktion behandelt und in Abschnitt 2.3 die aus der Formalisierung abgeleiteten FI–Kriterien vorgestellt. Die weiteren Abschnitte beschäftigen sich mit *Randbedingungen* für den Aufbau des Erreichbarkeitsgraphen und die Durchführung der Analyse ([Bre95]), *Präzedenzen* zur Auflösung von Feature–Interaktionen ([BrGo94a], [BrGo94b]), und den Ergebnissen einer Analyse einer Estelle–Spezifikation durch das Werkzeug *CONFINE* ([Th95], [ThBr95]), soweit diese für das in Kapitel 3 vorgestellte Analyseverfahren hilfreich sind.

[1] **E**xtended **F**inite **S**tate **A**utomaton

2.1 Grundbegriffe

In diesem Abschnitt werden einige für diese Arbeit wichtige Begriffe kurz erläutert und ihr entsprechendes Gegenstück in Estelle angegeben. Eine umfassende Darstellung der Begriffe ist in [Bre95] zu finden.

- **Lokaler Zustand**
  Unter dem Begriff des lokalen Zustandes eines in Estelle spezifizierten Systems wird eine Belegung für alle Elemente eines erweiterten endlichen Automatens bzw. für ein Estelle–Modul mit Werten verstanden. Zum Beispiel ist „(Idle, 42, 'NoPanic')“ eine Belegung

  - für den Hauptzustand (Idle),
  - für eine Variable vom Typ Ganzahl (42) und
  - für eine Variablen vom Typ Zeichenkette ('NoPanic').

  Diese Belegung repräsentiert einen lokalen Zustand eines Estelle–Moduls mit drei Elementen. *Elemente* eines Estelle–Moduls können der Hauptzustand, Variablen, Modulparameter, Nachrichten Nachrichtenparameter etc. sein. Der Haupzustand ist dabei ein ausgezeichnetes Element in Estelle bzw. in der Automatentheorie und hat einen endlichen Wertebereich.

- **Globaler Zustand**
  Der globale Zustand beschreibt den Gesamtzustand des Systems. Er setzt sich aus den lokalen Zuständen des Systems zusammen. In Estelle wird durch den globalen Zustand entsprechend der Zustand der Spezifikation beschrieben. Er gibt damit eine Belegung für alle Elemente der Estelle–Module an.

- **Abstrakter globaler Zustand**
  Der abstrakte globale Zustand ist eine Reduzierung des globalen Zustandes auf die Hauptzustände der einzelnen lokalen Zustände. Die sonstigen Elemente werden im abstrakten globalen Zustand nicht mehr repräsentiert.

- **Initialer globaler Zustand**
  Dies ist ein ausgezeichneter globaler Zustand, der Start–Zustand des Systems. I.a. können nach Ausführung der „Initialize“–Transitionen einer Estelle–Spezifikation unendlich viele initiale globale Zustände existieren. Für den Aufbau des Erreichbarkeitsgraphen und die

Durchführung der Analyse ist es aber erforderlich, daß diese Menge von initialen globalen Zuständen endlich ist. [2]

- **Zustandsraum**
  Der Zustandsraum ist die Menge der globalen Zustände, die ein spezifiziertes System annehmen kann. Letztlich beschreibt der Zustandsraum alle Kombinationen von Belegungen für die Elemente der lokalen Zustände der Teilsysteme.

- **Erweiterter Zustandsraum**
  Damit werden die Elemente der lokalen Zustände des Zustandsraumes ohne die Hauptzustände bezeichnet.

- **Transition**
  Eine Transition beschreibt die Änderungen der Werte der Elemente des Zustandsraumes beim Übergang von einem globalen Zustand zum nächsten. Der Übergang wird auch als **Schalten** bzw. **Feuern** einer Transition bezeichnet. In Estelle kann eine Transition erst dann schalten, wenn gewisse Bedingungen (**Schaltbedingungen**) erfüllt sind.

  Jede Transition hat zur Unterscheidung innerhalb ihres Features (s.u.) einen eindeutigen Namen.

- **Feature**
  Ein Feature ist eine endliche Menge von Transitionen und Erweiterungen des Zustandsraumes um neue Elemente, die eine bestimmte Funktionalität in einem Intelligenten Netzwerk realisieren.

  Jedes Feature hat zur Identifizierung einen eindeutigen Namen. Damit die Transitionen ihren entsprechenden Feature zugeordnet werden können, setzt sich der Namen einer Transition aus einem Namensteil für die Zuordnung zu einem Feature und einem Namensteil zur Unterscheidung der Transitionen innerhalb eines Features zusammen.

  Man beachte, daß das Basissystem auch ein Feature darstellt.

- **Globaler Automat**
  Der globale Automat ist der durch die gesamte Spezifikation beschriebene endliche erweiterte Automat. Formal ist es ein Tupel, bestehend aus der Menge der globalen Zustände, der Menge der Transitionen der Spezifikation und dem initialen globalen Zustand.

[2] In der vorliegenden Version des Werkzeuges wird davon ausgegangen, daß *genau* ein initialer abstrakter globaler Zustand vorliegt.

- **Abstrakter globaler Automat**
  Bei diesem Automaten wird die Menge der globalen Zustände des globalen Automaten durch die Menge der abstrakten globalen Zustände ersetzt.

- **Berechnungsfolge**
  Eine Berechnungsfolge ist eine (i.a. unendliche) Folge von globalen Zuständen. Sie startet immer im initialen globalen Zustand. Ein nächster globaler Zustand in der Berechnungsfolge wird durch das Schalten einer Transition der Spezifikation erreicht.

- **Berechnungsfolgen eines Features**
  Dies sind alle Berechnungsfolgen, in denen beim Übergang von einem globalen Zustand zum nächsten mindestens eine Transition des gegebenen Features benutzt wird.

  Man beachte, daß dieselben globalen Zustände mehreren Berechnungsfolgen verschiedener Features angehören können.

- **Hinzufügen eines Features**
  Das Hinzufügen eines Features bedeutet, daß in der Regel eine Erweiterung des Zustandsraumes einer Spezifikation erfolgt, und daß neue, dieses Feature realisierende Transitionen in die Spezifikation aufgenommen werden. Dadurch werden neue Berechnungsfolgen im globalen Automaten möglich, d.h. das Basissystem wird um das neue Feature erweitert.

Zum besseren Verständnis der Arbeit werden folgende Konventionen für die Namensgebung von Transitionen und Features benutzt:

- Mit $f, g, h, \ldots, l$ werden Transitionen der zum Basissytem hinzugefügten Features $F, G, H, \ldots, L$ bezeichnet.

- b bezeichnet eine Transition des Basissystems B.

- s und t bezeichnen beliebige Transitionen der Spezifikation. Dies können Transitionen eines hinzugefügten Features oder Transitionen des Basissystems sein.

- Ein Index (z.B. t_i) dient zur weiteren Unterscheidung der Transitionen der gesamten Spezifikation bzw. eines Features (z.B. f_i). Damit stammen zum Beispiel die Transitionen f_1 und f_2 aus dem selben Feature F.

2.2 Zentrale Aussage der Formalisierung von Feature–Interaktionen

Die zentrale Aussage der Formalisierung einer Feature–Interaktion in [Bre95] ist:

> *Eine Feature–Interaktion liegt dann vor, wenn das* ***Verhalten*** *eines Features durch das Hinzufügen eines anderen Features verändert wird, unabhängig von der Implementation von Indeterminismus für eine spezielle FDT.*

Das Verhalten eines Features wird in der Formalisierung aus [Bre95] durch die Menge der *Berechnungsfolgen* beschrieben, in denen eine Transition des Features vorkommt. Damit gilt für das Vorhandensein einer Feature–Interaktion, daß die Menge der Berechnungsfolgen für ein Feature durch das Hinzufügen eines anderen Features in einer beliebigen Art und Weise verändert bzw. beeinflußt werden muß.

Eine direkte Änderung der Menge der Berechnungsfolgen für ein Feature F ergibt sich, wenn durch das Hinzufügen eines Features G neue Berechnungsfolgen entstehen, in denen dann Transitionen aus beiden Features F und G vorhanden sind.

Eine indirekte Änderung (genauer: Eine Beeinflussung der Berechnungsfolgen) eines Features kann durch *Indeterminismus* entstehen. Zum Beispiel nehme man an, daß ein Feature F hinzugefügt worden ist, und daß eine Transition f des Features F mit einer anderen Transition[3] t ein indeterministisches Verhalten zeigt. Weiterhin nehme man an, daß für eine beliebige Berechnungsfolge eines bereits früher hinzugefügten Features G die Transition t verantwortlich für einen Übergang von einem globalen Zustand zum nächsten ist. Dann führt die indeterministische Auswahlmöglichkeit zwischen den Transitionen f und t dazu, daß die Berechnungsfolge für Feature G mit der Transition g nicht mehr oder nur manchmal ausgeführt wird. Dies ist sicherlich eine Änderung des Verhaltens von Feature G. Diese Beeinflussung des Features G ist unabhängig von der Implementation von Indeterminismus, weil bereits die Tatsache, daß eine indeterministische Auswahlmöglichkeit existiert, für die Änderung des Verhaltens genügt.

Für das Basissystem B muß in Bezug auf die Änderung des Verhaltens durch das Hinzufügen eines Features F eine Ausnahme gemacht werden, weil

[3] Dies kann eine Transition des Basissystems oder eines beliebigen Features sein.

dieses Hinzufügen die Berechnungsfolgen des Basissystems ändert und damit eine Feature–Interaktion wäre. Gerade das Ändern bzw. das Erweitern des Verhaltens des Basisisytems soll aber durch das Hinzufügen eines Features erreicht werden. Aus diesem Grund wird die Änderung des Verhaltens des Basissystems durch ein Feature nicht als Feature–Interaktion eingestuft.

Im nächsten Abschnitt werden die aus der Formalisierung einer Feature–Interaktion in [Bre95] abgeleiteten FI–Kriterien vorgestellt. Das in dieser Arbeit beschriebene Werkzeug sucht gemäß dieser Kriterien nach Feature–Interaktionen und stuft sie entsprechend nach ihrer „Gefährlichkeit“ bzw. „Qualität“ ein.

2.3 Kriterien für eine Feature–Interaktion

Die FI–Kriterien lassen sich in zwei Gruppen einteilen. Die eine Gruppe enthält die FI–Kriterien, die das Vorliegen einer Feature–Interaktion anhand von Berechnungsfolgen erkennen lassen. Die andere Gruppe enthält diejenigen FI–Kriterien, die eine Einstufung der Feature–Interaktion entsprechend ihrer „Gefährlichkeit“ bzw. „Qualität“ erlauben. Diese beiden Gruppen werden in den Unterabschnitten 2.3.1 und 2.3.2 behandelt.

Für die Anwendung der FI–Kriterien zur Einstufung von Feature–Interaktionen bzgl. ihrer Qualität bedarf es zusätzlich der Betrachtung der Auswirkungen einer Transition auf die Elemente des Zustandsraumes. Diese Auswirkungen werden in der Regel die Arbeitsweise einer anderen Transition beeinflussen. Stammen die sich beeinflussenden Transitionen aus verschiedenen Features, dann kann eine „gefährliche“ Feature–Interaktion vorliegen. Diese Auswirkungen werden unter dem Begriff der **Datenabhängigkeit** zwischen Transitionen zusammengefaßt und in Unterabschnitt 2.3.3 näher erläutert.

Für eine ausführliche und formale Definition bzgl. einer Feature–Interaktion und der einzelnen FI–Kriterien, sei auf [Bre95] verwiesen.

2.3.1 FI–Kriterien zur Erkennung einer Feature–Interaktion

Die in diesem Abschnitt vorgestellten FI–Kriterien dienen zur Erkennung einer Feature–Interaktion. Diese werden im weiteren auch als **Basiskriterien** bezeichnet.

Man beachte, daß diese Basiskriterien grundlegende Situationen für eine Feature–Interaktion beschreiben. Alle weiteren Situationen für Feature–Interaktionen sollten durch eine Kombination dieser Basiskriterien abgedeckt sein. Eine Feature–Interaktion liegt auf jeden Fall dann vor, wenn eines der folgenden Basiskriterien zutrifft:

1. Basiskriterium 1 besagt, daß eine Feature–Interaktion dann vorliegt, wenn durch das Hinzufügen eines Features G neue Berechnungsfolgen entstehen, in denen jeweils mindestens eine Transition eines bereits zuvor enthaltenen Features F und des neuen Features G an den Berechnungsfolgen beteiligt sind. In diesem Fall gehören die neuen Berechnungsfolgen zur Menge der Folgen von Feature F und zur Menge der Folgen von Feature G. Abbildung 2.1 zeigt die entstehende Berechnungsfolge.

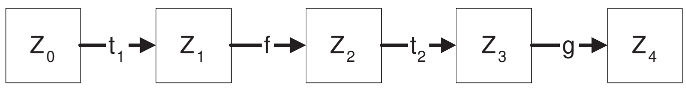

Abbildung 2.1: Zwei Transitionen verschiedener Features in einer Berechnungsfolge

2. Eine weitere Situation für eine Feature–Interaktion ist in Abbildung 2.2 dargestellt. Die Abbildung zeigt das Basiskriterium 2. Dieses Kriterium findet dann Anwendung, wenn zwischen Transitionen f und g aus verschiedenen Features F und G ein indeterministisches Verhalten durch das Hinzufügen eines der beiden Features entsteht.

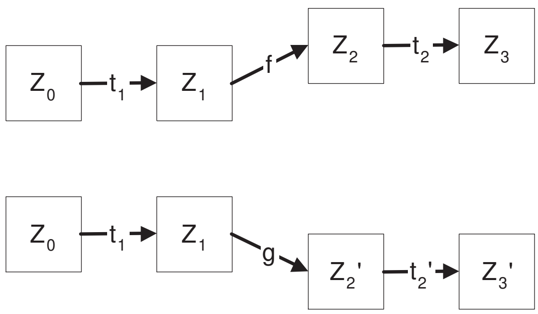

Abbildung 2.2: Indeterminismus zwischen Transitionen verschiedener Features.

3. Ein drittes Basiskriterium liegt in der in Abbildung 2.3 gezeigten Situation vor. Dieses Kriterium wird dann angewendet, wenn eine indirekte Beeinflussung der Berechnungsfolgen für die Transitionen f aus dem Feature F und der Transition g aus einem anderen Feature G besteht. Die dargestellte Situation zwischen den verschiedenen Features F und G ist unabhängig davon, aus welchem Feature die Transition t stammt[4]. Es muß aber gelten, daß zwischen f und t ein Indeterminismus vorliegt. Dieser Indeterminismus führt dazu, daß eine Transition g des Features G bei Ausführung der Berechnungsfolge von Feature F nicht zum Zuge kommt, so daß eine Feature–Interaktion zwischen F und G entsteht.

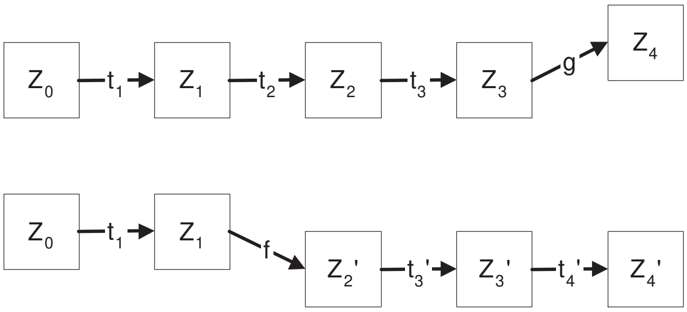

Abbildung 2.3: Indeterminismus zwischen einer Transition eines hinzugefügten Features und einer Transition des Basissystems.

2.3.2 FI–Kriterien zur Einstufung einer Feature–Interaktion

Die weiteren in [Bre95] vorgeschlagenen FI–Kriterien dienen dazu, die durch die Basiskriterien erkannten Feature–Interaktionen entsprechend ihrer „Gefährlichkeit“ bzw. „Qualität“ einzuteilen. Dazu werden folgende FI–Kriterien benötigt:

1. Das in Abbildung 2.4 gezeigte **Cross–Reset–Kriterium** ermöglicht es, bestimmte Situationen, die durch das Basiskriterium 3 oder das Basiskriterium 1 entdeckt werden, als unproblematisch einzustufen.
   Die Idee bei diesem Kriterium ist es, daß die betrachteten reaktiven Systeme nach Beendigung ihrer Arbeit wieder in einen definierten Zustand, den sogenannten **Null–Zustand**, zurückkehren, um auf eine

[4]Die Transition t könnte zum Beispiel eine Transition des Basissystems B sein.

erneute Anfrage reagieren zu können. Dabei verliert jeder der beteiligten Automaten des Systems seine bis dahin angesammelten Informationen vollständig, so daß ein erneuter Start nicht von einem vorherigen Durchlauf abhängig wird.
Obwohl sich die Berechnungsfolgen des Features F und des Features G nach Basiskriterium 1 überschneiden bzw. beeinflussen, ist die in der Abbildung 2.4 dargestellte Situation unkritisch, weil Transition g des Features G erst nach Erreichen des Null–Zustandes zum Zuge kommt.

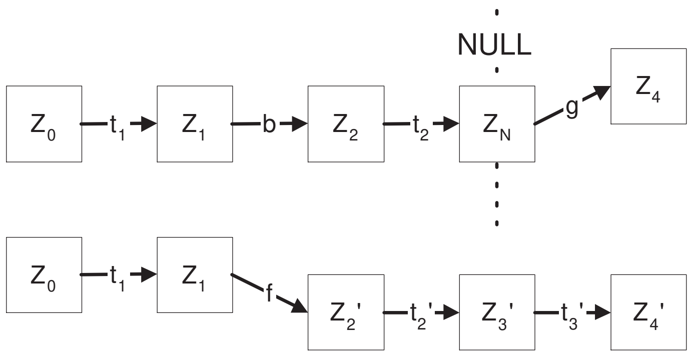

Abbildung 2.4: Cross–Reset Situation

2. Ein weiteres Kriterium um harmlose Feature–Interaktionen zu identifizieren, ist das **Independent–FI–Kriterium**. Es bezieht sich auf die in Abbildung 2.5 dargestellte Situation.
   Das Ziel dieses Kriteriums ist es, solche Situationen mit Indeterminismus zwischen Transitionen als harmlos zu erkennen, in denen diese Transitionen unabhängig und in beliebiger Reihenfolge schalten können. Dies erfordert, daß keine der Transitionen Auswirkungen auf die Elemente des Zustandsraumes der jeweiligen anderen Transition hat, d.h. die Transitionen arbeiten auf Elementen aus disjunkten Teilmengen des Zustandsraumes. Unter dieser Vorraussetzung kann das Schalten einer der Transitionen nicht das Schalten der anderen Transition beeinflussen oder sogar verhindern.
   Wichtig ist dieses Kriterium im Zusammenhang mit verteilten Systemen, weil durch die Nebenläufigkeit der Subsysteme sehr viele Situationen mit Indeterminismus zwischen Transitionen entstehen können, die dieses Kriterium erfüllen. Diese können dann mit dem Independent–FI–Kriterium als weniger „gefährlich“ eingestuft werden.

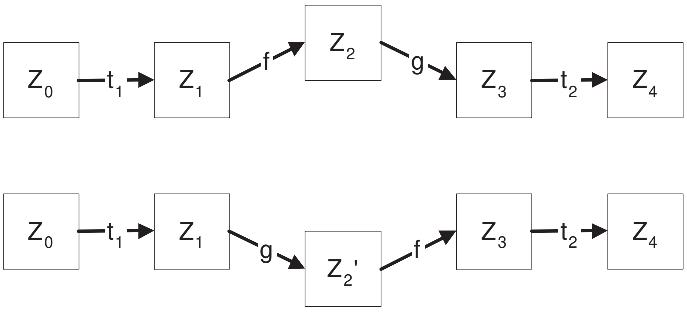

Abbildung 2.5: Independent Situation

3. Das **Minor–FI–Kriterium** erlaubt es, eine Feature–Interaktion als harmlos einzustufen, wenn
   (a) eine Transition f des Features F keine Auswirkungen auf die Elemente des erweiterten Zustandsraumes hat, auf denen eine Transition g eines anderen Features G arbeitet, und, wenn
   (b) die Transition f in den gleichen Hauptzustand zurückführt, aus dem sie gestartet ist oder eine in ihren Auswirkungen auf den Hauptzustand äquivalente Basistransition b zu f existiert.

   Die Rückkehr in den gleichen Hauptzustand bzw. die Existenz einer Transition des Basissystems mit äquivalenten Auswirkungen auf den Hauptzustand werden auch unter dem Begriff **Major–State–Preserving** zusammengefaßt (siehe [Bre95]). Abbildung 3 zeigt eine typische Situation für das Minor–FI–Kriterium.

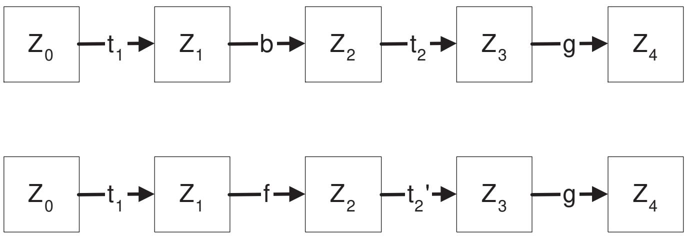

Abbildung 2.6: Minor–FI Situation

2.3.3 Datenabhängigkeit zwischen Transitionen

Zwischen zwei Transitionen besteht eine Datenabhängigkeit,

- wenn Elemente des Zustandsraumes durch eine der beiden Transitionen verändert werden, so daß die jeweilige andere Transition von den Änderungen beeinflußt wird, oder,
- wenn beide Transitionen gemeinsam Elemente des Zustandsraumes ändern, so daß eine beliebige dritte Transition dadurch beeinflußt wird.

Hinter der Datenabhängigkeit zwischen Transitionen steht damit die Frage nach der Beeinflussung des Verhaltens eines Features durch Änderungen von Elementen des Zustandsraumes durch ein anderes Feature. Dabei steht nicht im Vordergrund, *wie* der Zustandsraum geändert wird, sondern die Tatsache der Beeinflussung an sich. Auf diese Weise ist es möglich, von den globalen Zuständen zu abstrahieren. Man gelangt dann zu den abstrakten globalen Zuständen, ohne Eigenschaften zur Erkennung oder Einstufung von Feature–Interaktionen zu verlieren.

Der wesentliche Vorteil der Abstraktion ist, daß eine Zustandsexplosion[5] vermieden werden kann.

Dies erleichtert die Analyse, da beim Übergang von einem abstrakten globalen Zustand zum nächsten keine Wertverfolgung für die einzelnen Elemente des erweiterten Zustandsraumes erforderlich wird. So müssen zum Beispiel für eine konkrete Estelle–Spezifikation die neuen Werte einer Variablen, die durch das Schalten einer Estelle–Transition verändert würden, nicht berechnet werden. Statt dessen muß lediglich festgehalten werden, daß eine Transition ein Element verändert hat, und welche anderen Elemente, auf denen die Transition arbeitet, davon abhängig sind. Dann können Änderungen, die eine Transition durchführt, in den Berechnungsfolgen nachvollzogen werden, ohne die konkreten Belegungen der Elemente mit Werten berechnen zu müssen.

Als Grundlage für die Datenabhängigkeitsanalyse dient das in [Bre95] eingeführte **Output–Dependency–Pattern**. Mit diesem Pattern wird eine Beziehung hergestellt zwischen den durch eine Transition geänderten Elementen des Zustandsraumes (beschrieben durch das **Output–Pattern**)

[5]Enthält zum Beispiel der Zustandsraum ein Element, dessen Wertebereich unbeschränkt ist, dann gibt es unendlich viele globale Zustände. Dies bezeichnet man als Zustandsexplosion.

und den Komponenten des Zustandsraumes, von denen diese Änderungen abhängig sind (beschrieben durch das **Input–Pattern**).

Als Beispiel für die einzelnen Pattern diene eine Estelle–Transition mit der Anweisungsfolge „$x := x + y; z := x$“ als einziger Anweisungen im Transitionsblock. Weiterhin habe diese Estelle–Transition zur Vereinfachung des Beispiels keine Transitionsklauseln[6]. Dann bestimmt die Menge $\{x, z\}$ der linken Seiten der Zuweisungen das *Output–Pattern*, die Menge $\{x, y\}$ der rechten Seiten der Zuweisungen das *Input–Pattern* und die Menge { x$\leftarrow\{x, y\}$, z$\leftarrow\{x, y\}$ } das *Output–Dependency–Pattern* für diese Estelle–Transition. $\leftarrow$ hat dabei die Semantik, daß die linke Seite von den Elementen der rechten Seite abhängig ist.

Man beachte, daß die Zuweisung einer Konstanten an eine Variable, jede der bis dahin bestandenen Abhängigkeiten der Variablen von anderen Elementen des Zustandsraumes unterdrückt. Die Variable ist also nach der Zuweisung einer Konstanten von keinen anderen Elementen mehr abhängig. Dies gilt auch für den Hauptzustand, wenn man die `TO`–Klausel in Estelle wie eine Zuweisung einer Konstanten auf eine ausgezeichnete Variable betrachtet.

Es lassen sich zwei Arten von Datenabhängigkeiten zwischen Transitionen unterscheiden:

1. Eine *direkte* und
2. Eine *indirekte* Datenabhängigkeit

Eine *direkte* Datenabhängigkeit besteht zwischen zwei Transitionen, wenn das Output–Pattern der einen Transition sich mit dem Input–Pattern der anderen Transition überschneidet, d.h. wenn die Schnittmenge der beiden Pattern nicht leer ist.

Dagegen zeichnet sich eine *indirekte* Datenabhängigkeit zwischen zwei Transitionen t_1 und t_2 dadurch aus, daß die Elemente des Zustandsraumes, auf denen eine dritte Transition t_x arbeitet, durch eine der Transitionen (zum Beispiel t_1) geändert wird und sich diese Änderungen indirekt über das Output–Pattern der Transition t_x auf die andere Transition t_2 auswirken. Dies ist nur dann kritisch, wenn die Transitionen in der Reihenfolge t_1, t_x, t_2 beim Übergang von einem globalen Zustand zum nächsten in einer Berechnungsfolge auftreten.

[6] Ansonsten müssen die in den Transitionsklauseln referenzierten Elemente des Zustandsraumes geeignet in das *Input–Pattern* aufgenommen werden.

Eine indirekte Datenabhängigkeit besteht auch, wenn beide Transitionen t_1 und t_2 über ihre Output–Pattern eine Auswirkung auf die Elemente der Transition t_x haben, d.h. ein Teil des Input–Patterns der Transition t_x ist von der Transition t_1 und ein anderer Teil ist von der Transition t_2 abhängig. Abbildung 2.7 zeigt die beiden Möglichkeiten für eine indirekte Datenabhängigkeit.

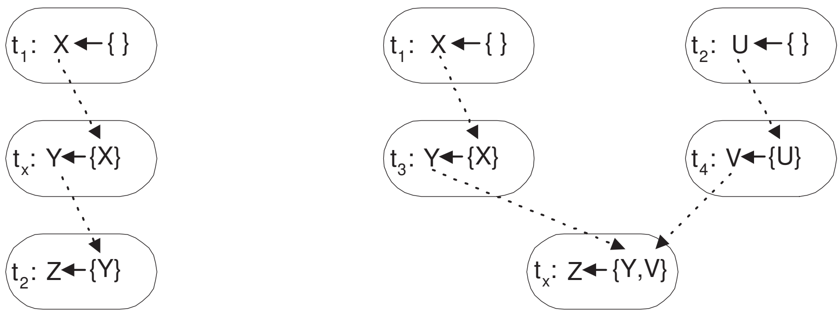

Abbildung 2.7: Indirekte Datenabhängigkeiten zwischen zwei Transitionen

Zusammenfassend bedeutet *Datenabhängigkeit* zwischen Transitionen verschiedener Features, daß diese Transitionen Elemente des Zustandsraumes ändern, die nicht ausschließlich ihnen zugeordnet sind, und daß diese Änderungen eine Auswirkung auf Transitionen der anderen Features haben.

2.4 Randbedingungen

Die Randbedingungen der Arbeit ergeben sich zum größten Teil direkt aus den FI–Kriterien aus [Bre95] und aus prinzipiellen Überlegungen zur Eindämmung des Problems einer Zustandsexplosion.

Eine der Randbedingung zur Vereinfachung der Analyse ist der Verzicht auf dynamisches Kreieren und Terminieren von Komponenten der Spezifikation bzw. Elementen des Zustandsraumes. Dadurch kann die Analyse auf statische Strukturen beschränkt werden. Außerdem erlaubt die Einschränkung auf statische Strukturen, jedes Element des Zustandsraumes durch den in der Spezifikation benutzten qualifizierten Namen[7] eindeutig zu identifizieren.

[7] z.B. Variablennamen, Transitionsnamen, etc.

Eine der wichtigsten Randbedingungen ist, daß keine Wertverfolgung für die erweiterten Zustände durchgeführt werden muß, wie im Abschnitt 2.3.3 zum Thema *Datenabhängigkeit* bereits erläutert wurde. Für das Vorhandensein einer Datenabhängigkeit zwischen Transitionen von verschiedenen Features ist die allgemeinere Frage relevant „*wird ein Feature durch ein anderes Feature über das Ändern von Elementen des Zustandsraumes in seinem Verhalten gestört*“ und nicht die speziellere Frage „***wie*** *wird das Verhalten des einen Features durch das andere Feature beeinflußt*“.

Dadurch ist es möglich, von den Belegungen der Elemente des erweiterten Zustandsraumes mit Werten zu abstrahieren. Dies bedeutet, daß die Output–Dependency–Pattern ausreichen zur Beschreibung einer Datenabhängigkeit zwischen Transitionen verschiedener Features. Diese Abstraktion erlaubt eine erhebliche Einschränkung des zu betrachtenden Zustandsraumes. Tatsächlich geht man durch diese Abstraktion von einem im allgemeinen unendlichen Zustandsraum zu einem endlichen, meistens aber immer noch sehr großen Zustandsraum über, da nur noch die Belegungen der Hauptzustände eines lokalen Zustandes mit Werten betrachtet werden müssen.

Für die Berechnungsfolgen bedeutet dies, daß anstatt der Folgen von globalen Zuständen nun Berechnungsfolgen bestehend aus abstrakten globalen Zuständen betrachtet werden können. Die Berechnungsfolgen werden also durch die Hauptzustände der einzelnen Estelle–Module eines verteilten Systems bestimmt[8].

Dieser Übergang bedeutet aber nicht, daß keine unendlichen Berechnungen mehr durchgeführt werden können. Es kann zum Beispiel das System in einer Schleife unendlich oft immer wieder in denselben abstrakten globalen Zustand zurückkehren.

Die Abstraktion auf die Hauptzustände ist zulässig, weil alle Feature–Interaktionen nach dem Abstraktionsschritt immer noch erkennbar sind.

Eine weitere durch [Bre95] gegebene Randbedingung ist, daß alle Transitionen einen eindeutigen Namen erhalten, bestehend aus einem Anteil zur Bestimmung des Features zu dem eine Transition gehört und einem Anteil zur Identifizierung der Transition innerhalb der Menge der Transitionen eines Features. Erst dadurch ist es möglich Transitionen verschiedener Features zu unterscheiden und die FI–Kriterien anzuwenden.

[8] Es sei angemerkt, daß das Werkzeug nicht nur die Hauptzustände, sondern auch einige andere Elemente des Zustandsraumes zusätzlich beachtet. Dies wird in Abschnitt 3.1.1 des Kapitels 3 näher erläutert.

2.5 Präzedenzen zur Auflösung von erkannten Feature–Interaktionen

Eine Auflösung von Feature–Interaktionen, die aufgrund von Indeterminismus zwischen Transitionen verschiedener Features entstehen, kann durch Einführung einer Vorrangbeziehung (**Präzedenzen**) zwischen den verschiedenen Feature erreicht werden (siehe [BrGo94a], [BrGo94b]). Die Präzedenzen werden in Form einer **Präzedenzmatrix** angegeben.

Sie definieren eine *Partialordnung* „$\succ$“ auf den Features, so daß bei unerwünschtem Indeterminismus zwischen Transitionen verschiedener Feature eine Vorrangbeziehung zwischen diesen Transitionen entsteht. Besteht zum Beispiel eine Präzedenz $F \succ G$ zwischen einem Feature F und einem Feature G, dann erhalten alle Transitionen f des Features F Vorrang vor den Transitionen g des Features G. Wenn dann bei Ausführung einer Spezifikation in einem globalen Zustand als nächstes die Transitionen f und g schalten können, dann wird die Transition f[9] und nicht g ausgewählt.

Eine Umsetzung der Präzedenzen für die FDT Estelle geschieht dadurch, daß *Prioritäten* für die Transitionen entsprechend der Präzedenzen zwischen den Features vergeben werden. Dazu wird eine Priorität logisch in zwei Teile aufgespalten:

- Einen Teil, der die Präzedenz des Features widerspiegelt und
- einen Teil, der die Vorrangbeziehung der Transitionen innerhalb eines Features regelt.

Dies gewährleistet für ein Feature F mit einer höheren Präzedenz gegenüber einem anderen Feature G, daß alle Transitionen des Features F eine höhere Priorität als die Transitionen des Features G erhalten.

Die Einführung einer Präzedenz zwischen zwei Features F und G ist allerdings kritisch, da eine Vorrangbeziehung zwischen *allen* Transitionen des Features F und des Features G entsteht. Dadurch kann es passieren, daß eine Transition h eines dritten Features H, wie in Abbildung 2.8 dargestellt, nicht ausgeführt wird und damit das Verhalten des Features H verändert wird, d.h. es besteht eine Feature–Interaktion zwischen dem Feature F und dem Feature H. Gilt aber $F \succ H$, d.h. das Feature F hat bereits Vorrang

[9] Sofern es nicht noch eine schaltbare Transition i aus einem anderen Feature I mit $I \succ F$ gibt.

vor dem Feature H, dann ist diese Situation unkritisch und die Auflösung des ursprünglichen Indeterminismus durch $F \succ G$ kann in dieser Weise durchgeführt werden. Andernfalls kann eine Auflösung mit $F \succ H$ versucht werden (falls nicht schon eine Präzedenz $H \succ F$ existiert).

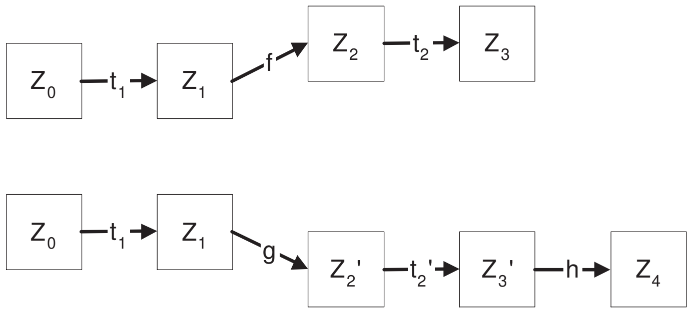

Abbildung 2.8: Durch eine Präzedenz $F \succ G$ verursachte Feature–Interaktion zwischen F und H

Man beachte, daß die Situation in Abbildung 2.8 äquivalent zu einer Situation ist, in der für das Feature G das Basissystem B eingesetzt wird. Das Problem mit einer Präzedenz zwischen Feature F, dem Feature H und dem Basissystem B stellt sich dann in der gleichen Weise und kann entsprechend gelöst werden.

2.6 Beschleunigung der Analyse durch CONFINE

CONFINE ([Th95], [ThBr95]) ist ein für die FDT Estelle geschriebenes Werkzeug zur Erkennung von *Indeterminismen* und *Überlappungen* zwischen Transitionen einer Estelle–Spezifikation.

Zwei Transitionen *überlappen* sich, wenn der zu Grunde liegende Indeterminismus zwischen den Transitionen durch unterschiedliche Prioritäten verhindert wird.

Das Werkzeug CONFINE untersucht dazu die Klauseln der Estelle–Transitionen jeweils eines Modulrumpfes auf ihre gleichzeitige Erfüllbarkeit. Wenn CONFINE nicht die gleichzeitige Erfüllbarkeit der Klauseln der Transitionen ausschließen kann, dann meldet CONFINE das Vorhandensein eines *möglichen* Indeterminismus bzw. einer *möglichen* Überlappung zwischen den Transitionen.

Man beachte, daß von CONFINE *alle* Paare von Transitionen mit möglichen Indeterminismen bzw. Überlappungen gemeldet werden, wodurch im Umkehrschluß gefolgert werden kann, daß für alle anderen Paare kein Indeterminismus bzw. keine Überlappung vorliegt. Diese Tatsache ist wichtig im Zusammenhang mit der Abstraktion von globalen Zustände auf abstrakte globale Zustände, denn in diesem Fall wird als einzige Bedingung für das Schalten von Transitionen der Hauptzustand des entsprechenden Estelle–Moduls herangezogen. Dies hat zur Konsequenz, daß in der Regel mehr Transitionen eines Estelle–Moduls in einem abstrakten globalen Zustand schaltbar werden können als in dem entsprechenden globalen Zustand. Für die Analyse mittels der FI–Kriterien aus Abschnitt 2.2 bedeutet dies, daß durch die Abstraktion mehr Situationen entstehen, in denen ein Indeterminismus zwischen Transitionen vorliegen könnte.

Die von CONFINE gelieferten Ergebnisse über die Eigenschaften der Spezifikation ermöglichen es nun, die durch die Abstraktion zusätzlich entstandenen Situationen mit Indeterminismus zwischen Transitionen als solche zu erkennen und zu behandeln.

Weiterhin folgt, daß die Anzahl der gemeldeten potentiellen Feature–Interaktionen verkleinert werden kann, weil ohne die Information, daß eine determinisitische Auswahl zwischen zwei Transitionen vorliegt, die entsprechenden Situationen mittels dem Basiskriterium 2 als Feature–Interaktionen erkannt würde.

Ein weiteres wichtiges Ergebnis von CONFINE ist, daß CONFINE solche Transitionen aus der Spezifikation[10] entfernen kann, die von CONFINE als *sicher* nicht ausführbar („**dead code**“) erkannt worden sind. Dieses Ergebnis ist wieder im Zusammenhang mit dem Übergang von globalen Zustände auf abstrakte globale Zustände wichtig, da durch die Abstraktion keine Entscheidungen über sicher nicht ausführbare Transitionen getroffen werden können, so daß Berechnungsfolgen mit diesen als sicher nicht ausführbar erkannten Transitionen im Erreichbarkeitsgraphen zu berücksichtigen wären. Durch CONFINE ist es nun möglich, auf solche unmöglichen Berechnungsfolgen zu verzichten.

Darüber hinaus liefert CONFINE weitere Ergebnisse über Eigenschaften der Spezifikation, die für die vorliegende Arbeit aber keine Rolle spielen.

Für weitere Informationen über das Werkzeug CONFINE und das benutzte Analyseverfahren sei auf [Th95] verwiesen.

[10]bzw. aus dem PET–Objektformat, siehe auch Technische Umgebung des hier vorgestellten Werkzeuges in Abschnitt 4.1.1

Kapitel 3

Konzepte und Algorithmen der Analyse

In diesem Kapitel werden die wesentlichen Konzepte und Algorithmen, auf denen die Implementation des Werkzeuges beruht, vorgestellt.

Ziel der Arbeit ist die Implementation der in Abschnitt 2.2 vorgestellten FI–Kriterien. Dazu baut das Werkzeug einen **Erreichbarkeitsgraphen** bzw. einen dazu äquivalenten **Transitionsgraphen** auf und führt auf diesem Transitionsgraphen eine Suche nach den Situationen für Feature–Interaktionen mittels der FI–Kriterien durch. Der Erreichbarkeitsgraph bzw. der Transitionsgraph dient dabei als Mittel zur Darstellung aller möglichen Berechnungsfolgen einer Estelle–Spezifikation.

Die erkannten Feature–Interaktionen werden dann mit den FI–Kriterien entsprechend ihrer „Gefährlichkeit“ eingestuft. Zur Einstufung der Feature–Interaktionen ist zusätzlich eine Datenabhängigkeitsanalyse zwischen den Transitionen der verschiedenen Features mittels der Output–Dependency–Pattern notwendig, so daß das Minor–FI–Kriterium bzw. das Independent–FI–Kriterium angewendet werden kann.

Weiterhin untersucht das Werkzeug anhand des Erreichbarkeitsgraphen die Auswirkungen der Einführung von Präzedenzen zwischen Features, so daß entschieden werden kann, ob eine Auflösung erfolgreich durchgeführt worden ist und ob dadurch neue Feature–Interaktionen enstanden sind.

Schließlich werden als Ergebnis der Analyse alle Feature–Interaktionen zwischen den Estelle–Transitionen ausgegeben.

Als Hilfsmittel für die Analyse dient das in Abschnitt 2.5 vorgestellte Werkzeug CONFINE.

In Abschnitt 3.1 dieses Kapitels wird der zum *Erreichbarkeitsgraphen* äquivalente *Transitionsgraph* für die Darstellung der Berechnungsfolgen vorgestellt und eine Erweiterung der abstrakten globalen Zustände um einige Elemente des Zustandsraumes besprochen.

Abschnitt 3.2 behandelt dann das im Werkzeug angewendete Verfahren zur Erkennung einer Feature–Interaktion und zu ihrer Einstufung entsprechend ihrer „Gefährlichkeit“. Unter anderem werden in diesem Abschnitt die **Tiefensuche** als der grundlegende Algorithmus zur Suche nach Feature–Interaktionen im Transitionsgraphen und eine Abschätzung für die Laufzeitkomplexität des Analyseverfahrens angegeben.

Die weiteren Abschnitte dieses Kapitels beschäftigen sich zunächst mit den Auswirkungen der Nebenläufigkeit in einem verteilten System auf den Aufbau des Transitionsgraphen und das Analyseverfahren (Abschnitt 3.3), dann mit den notwendigen und für die Analyse günstigen Anforderungen an einen Estelle–Spezifikationstil (Abschnitt 3.4) und schließlich mit der sogenannten **Zwischenform**, einer Estelle ähnlichen, textuellen Form der Ausgangsspezifikation (Abschnitt 3.5).

3.1 Der Transitionsgraph

Grundlage für die im Abschnitt 3.2 vorgestellte Suche nach Feature–Interaktionen mittels der FI–Kriterien aus Abschnitt 2.3 ist der *gerichtete Transitionsgraph*. Dieser entsteht durch eine Änderung der Sichtweise auf den Erreichbarkeitsgraphen.

Die *Knoten* des Erreichbarkeitsgraphen stellen die erreichbaren abstrakten globalen Zustände des Systems dar. Die benannten gerichteten *Kanten* des Erreichbarkeitsgraphen repräsentieren die beim Übergang von einem abstrakten globalen Zustand zum nächsten geschalteten Estelle–Transitionen.

Erreichbar bedeutet in diesem Zusammenhang, daß eine Estelle–Transition existiert, die für den Übergang von einem abstrakten globalen Zustand zum nächsten verantwortlich ist und daß vom initialen abstrakten globalen Zustand nur erreichbare Zustände als Folgezustände im Graphen möglich sind.

Die Verschiebung der Sichtweise vom Erreichbarkeitsgraphen zum Transitionsgraphen besteht darin, daß die Knoten des Transitionsgraphen die Kanten des Erreichbarkeitsgraphen und die Kanten des Transitionsgraphen die Knoten des Erreichbarkeitsgraphen repräsentieren.

Dies bedeutet, daß im Transitionsgraphen die geschalteten Transitionen zu **Knoten** und die abstrakten globalen Zustände zu **gerichteten Kanten** werden. Dabei ist für jede Kante des Erreichbarkeitsgraphen ein eigener Knoten im Transitionsgraphen erforderlich.

In Abbildung 3.1 ist ein Beispiel für die Umwandlung eines Erreichbarkeitsgraphen in einen Transitionsgraphen gezeigt.

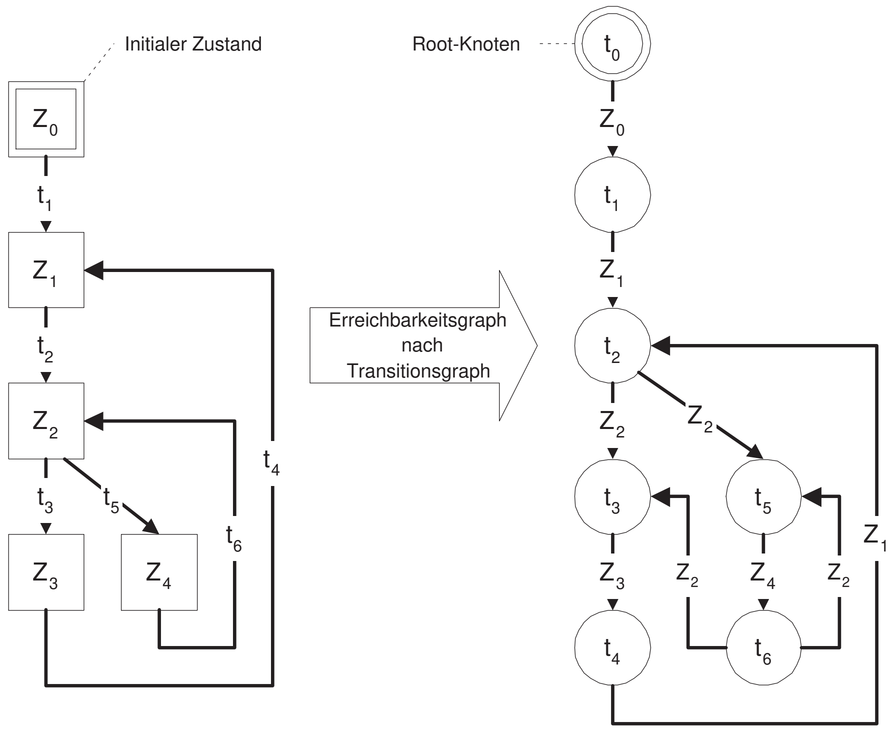

Abbildung 3.1: Umwandlung des Erreichbarkeitsgraphen in den Transitionsgraphen

Ein Grund für den Übergang vom Erreichbarkeitsgraphen zum Transitionsgraphen ist die Beobachtung, daß für die Anwendung der FI–Kriterien die Fragen im Vordergrund stehen „*Welche Transitionen haben mit welchen anderen Transitionen ein indeterministisches Verhalten?*“ und „*Welche Transitionen können in den Berechnungsfolgen nach einer Transition folgen?*„.

Die Information „*Aus welchem und in welchen abstrakten globalen Zustand wird das System durch das Schalten einer Transition überführt?*“ ist lediglich für die Bestimmung der schaltbaren Transitionen beim Aufbau des Graphen wichtig, nicht jedoch für die Analyse mittels der FI–Kriterien.

Ein weiterer Vorteil des Transitionsgraphen ist die Tatsache, daß die Folge der geschalteten Transitionen bereits alle für die Analyse notwendigen Informationen enthalten, so daß eine Berechnungsfolge über die Folgen der geschalteten Transitionen, die beim Übergang von einem abstrakten globalen Zustand zum nächsten benutzt werden, definiert werden kann.

Diese Definition für die Berechnungsfolgen ist zulässig, da durch den Übergang von den globalen Zuständen zu den abstrakten globalen Zuständen die Auswirkungen des Schaltens einer Estelle–Transition auf die abstrakten globalen Zustände deterministisch sind. Der Grund dafür ist, daß beim Feuern einer Transition ein deterministischer Zustandsübergang stattfindet. Eine indirekte Änderung durch einen indeterministischen Anteil (zum Beispiel in den `Provided`–Klauseln einer Transition) wird durch die Abstraktion auf die Hauptzustände nicht bei der Auswahl der in einem abstrakten globalen Zustand als nächste zu schaltenden Estelle–Transitionen herangezogen.

Zusammen mit der Beobachtung über die Anwendung der FI–Kriterien und der Tatsache, daß die geschalteten Transitionen alle notwendigen Informationen für die Analyse enthalten, ist es nach dem Aufbau des Transitionsgraphen möglich, auf die konkreten Attribute[1] der abstrakten globalen Zustände zu verzichten.

Ein weiterer Vorteil des Transitionsgraphen ist, daß der zur Analyse benutzte Tiefensuch–Algorithmus ([Sed95]) auf *Knoten* arbeitet. Prinzipiell könnte die Analyse auch auf den *Kanten* des Erreichbarkeitsgraphen durchgeführt werden. Dann würde aber letztlich während der Benutzung des Tiefensuch–Algorithmus eine implizite Umwandlung des Erreichbarkeitsgraphen in den Transitionsgraphen durchgeführt werden. Da im Transitionsgraphen diese Umwandlung explizit bereits vorgenommen worden ist, ist eine Suche auf dem Transitionsgraphen laufzeiteffizienter.

Aus diesen Gründen bietet der Transitionsgraph zur Erkennung von Feature–Interaktionen mittels der FI–Kriterien eine bessere und natürlichere Darstellung als der Erreichbarkeitsgraph.

[1]Dies sind die konkreten Belegungen für die Hauptzustände der einzelnen Estelle–Module. Im Werkzeug wird ein abstrakter globaler Zustand nur noch durch die in dem Zustand schaltbar gewordenen Estelle–Transitionen repräsentiert.

Die wesentliche Eigenschaft des Erreichbarkeitsgraphen, daß *alle* Berechnungsfolgen durch den Graphen erfaßt werden (so daß mittels der FI–Kriterien alle Feature–Interaktionen entdeckt werden können) bleibt erhalten, da die Änderung der Sichtweise letztlich eine bijektive Abbildung zwischen Erreichbarkeitsgraphen und Transitionsgraphen erlaubt.

3.1.1 Erweiterung der abstrakten globalen Zustände

In der aktuellen Version des Werkzeuges werden die abstrakten globalen Zustände ([Bre95]) um einige Komponenten des Zustandsraumes erweitert.

Diese Erweiterungen umfassen die *Nachrichten*, die zwischen verschiedenen Estelle–Modulen ausgetauscht werden, und einige *Variablen* der Spezifikation. Diese Variablen werden im folgenden als **Zustandsvariablen** bezeichnet. Damit werden neben den Hauptzuständen der einzelnen Estelle–Module zusätzlich Nachrichten und Zustandsvariablen beim Aufbau des Graphen berücksichtigt.

Die *Zustandsvariablen* müssen dabei in ihrem Verhalten und ihrer Anwendung den Hauptzuständen der Estelle–Module entsprechen, so daß eine umfangreiche Wertverfolgung für die Zustandsvariablen vermieden werden kann. Deshalb dürfen diesen Variablen nur Konstanten zugewiesen werden und diese Zuweisung darf innerhalb des Transitionsblockes nicht von einem anderen Element des erweiterten Zustandsraumes abhängig sein. Dadurch wird sichergestellt, daß keine Werte eines anderen Elementes des erweiterten Zustandsraumes verfolgt werden müssen, und daß ein eindeutiger nächster Zustand beim Schalten einer Transition erreicht werden kann.
Weiterhin dürfen in der aktuellen Version des Werkzeuges die Zustandsvariablen nur auf Gleichheit mit einer Konstanten abgeprüft werden. Diese Einschränkung könnte in einer Erweiterung des Werkzeuges fallengelassen werden.

Diese Semantik für die Zustandsvariablen erlaubt es, eine feinere Abbildung der Berechnungsfolgen in den Erreichbarkeitsgraphen bzw. Transitionsgraphen durchzuführen, als dies nur mit den Hauptzuständen möglich wäre. Als Beispiel für die feinere Abbildung diene eine Estelle–Transition, die beim Schalten nicht den Hauptzustand ihres Moduls, dafür aber eine der Zustandsvariablen ihres Moduls ändert. Dann ist der nächste erreichbare Zustand der gleiche abstrakte globale Zustand, aus dem die Transition wegführt, falls man nur die Hauptzustände betrachtet. Dagegen führt die

Transition unter Berücksichtigung der Zustandsvariablen in einen neuen abstrakten globalen Zustand.

Ein weiterer Vorteil ist, daß diese Variablen bei der Datenabhängigkeitsanalyse nicht berücksichtigt werden müssen, weil sie die Berechnungsfolgen beim Aufbau des Graphen direkt manipulieren und weil durch die ausschließliche Zuweisung von Konstanten[2] eine Datenabhängigkeit unterbrochen wird.

Ein Nachteil bei der Verwendung von Zustandsvariablen ist, daß sie die Menge der im Erreichbarkeitsgraphen bzw. Transitionsgraphen darzustellenden Zustände vergrößert. Diese Menge entspricht jedoch besser den bei einer Ausführung erreichten globalen Zuständen des spezifizierten Systemes. Trotzdem sollte man mit der Verwendung dieser Zustandsvariablen sehr sparsam umgehen, da ansonsten das Problem der Zustandsexplosion verschärft werden kann.

Die zweite Erweiterung der abstrakten globalen Zustände um die *Nachrichten*, die in einem verteilten System zwischen den einzelnen Estelle–Modulen ausgetauscht werden können, dient ebenfalls dem Zweck, die Menge der im Erreichbarkeitsgraphen bzw. Transitionsgraphen dargestellten abstrakten globalen Zuständen besser der Menge der globalen Zuständen anzupassen, die bei einer Ausführung des spezifizierten Systemes auftreten können.

Die Idee ist, daß die meisten der Nachrichten u.a. zur Synchronisation zwischen den einzelnen Estelle–Modulen dienen. Dies gilt im besonderen für die zu analysierenden Protokollautomaten Intelligenter Netzwerke.

Ohne Berücksichtigung der Nachrichten ist die einzige Bedingung für das „Schalten" einer Estelle–Transitionen eines Moduls, daß der durch das Modul spezifizierte Automat sich im entsprechenden Hauptzustand befindet. Dies bedeutet für eine Eingabetransition, daß sie auch dann im Graphen einen Zustandsübergang initiieren kann, wenn die Nachricht, auf die sie wartet, nicht anliegt. Dadurch ist es möglich, daß jedes Modul der Spezifikation unabhängig von jedem anderen Modul seine Eingabetransitionen „feuern" kann, d.h. die Nebenläufigkeit zwischen den Modulen wird bei der Analyse wesentlich gesteigert. Für den Transitionsgraphen hat dies zur Folge, daß je nach spezifiziertem System in den Graphen extrem viele Berechnungsfolgen aufgenommen werden, die bei Ausführung des spezifizierten Systemes nicht möglich sind.

Die Aufnahme von Nachrichten in den abstrakten globalen Zustand erfolgt durch die Aufnahme der Warteschlangen für die einzelnen Interaktionspunk-

[2]Dies entspricht einer Initialisierung der Variablen

te in einer Estelle–Spezifikation. Damit gilt für den Übergang von einem abstrakten globalen Zustand zum nächsten mittels einer Eingabetransition,

- daß der Automat sich im korrekten Hauptzustand befunden hat,
- daß die Zustandsvariablen den entsprechenden Wert angenommen haben und
- daß an dem Interaktionspunkt, an dem die Eingabetransition auf Nachrichten wartet, die entsprechende Nachricht angelegen hat.

Dadurch wird die Synchronisation zwischen den Modulen stärker berücksichtigt und es werden weniger (im Sinne der Estelle–Semantik) unmögliche Berechnungsfolgen mit in den Graphen aufgenommen.

Allerdings muß beachtet werden, daß in Estelle keine Begrenzung für die Anzahl der Interaktionen in einer Warteschlange vorgesehen sind, so daß unendlich viele Nachrichten in eine Warteschlange eingereiht werden können. Dadurch können unendlich viele abstrakte globale Zustände entstehen. Um dieses Problem zu vermeiden, muß die Länge der Warteschlangen begrenzt werden. Das Werkzeug bietet die Möglichkeit, für jede Warteschlange eine individuelle Maximallänge anzugeben, so daß eine möglichst gute Anpassung an das spezifizierte System möglich sein sollte.

Prinzipiell ist die Entscheidung für oder gegen eine Aufnahme der Nachrichten in die abstrakten globalen Zustände sehr von dem jeweiligen spezifizierten System abhängig. Findet zum Beispiel eine starke Synchronisation der einzelnen lokalen Automaten durch den Nachrichtenaustausch zwischen den Automaten statt[3], dann ist die Erweiterung des abstrakten globalen Zustandes mit Nachrichten wahrscheinlich sinnvoll. Sind die Automaten dagegen nur lose verbunden[4], dann ist es wesentlich besser die Nachrichten nicht mit aufzunehmen, da sie die Menge der im Erreichbarkeitsgraphen bzw. Transitionsgraphen darzustellenden abstrakten globalen Zustände vergrößert.

Das Werkzeug bietet über eine Option „**–msg**“[5] die Möglichkeit, den Transitionsgraphen mit oder ohne Beachtung der Nachrichten aufzubauen, so daß dem Nutzer des Werkzeuges die Entscheidung zur Laufzeit freisteht.

[3] d.h. das Voranschreiten der Automaten wird wesentlich durch die Nachrichten bestimmt

[4] d.h. die Automaten synchronisieren sich nur schwach.

[5] siehe Abschnitt 4.1.2 über die Optionen beim Aufruf des Werkzeuges

Im folgenden wird für die vorliegende Arbeit die Konvention getroffen, daß der „abstrakte globale Zustand“ die Erweiterung um Nachrichten und Zustandsvariablen enthält.

3.1.2 Aufbau des Transitionsgraphen

Zum besseren Verständnis des Aufbaus und der Analyse des Graphen sind die im folgenden definierten Begriffe von Bedeutung. Sie werden anhand des Transitionsgraphen in Abbildung 3.2 näher erläutert. $t_{i,j}$ bezeichnet dabei die i-te Transition der Spezifikation, j gibt an, welcher Knoten das Schalten der i-ten Transition kreiert, und Z_i den durch die i-te Transition erreichten abstrakten globalen Zustand. Zwei Knoten $t_{a,i}$ und $t_{a,j}$ sind dabei genau dann identisch (es gilt also $i = j$), wenn die durch das Feuern erreichten abstrakten globalen Zustände identisch sind. Derartige Knoten werden als **Instanzen** ihrer Transition bezeichnet.

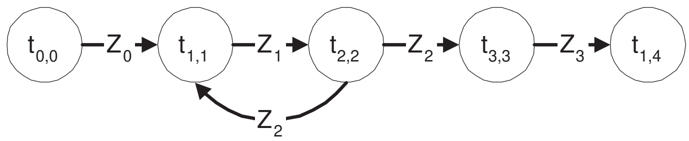

Abbildung 3.2: Beispiel für einen Transitionsgraphen

- **Eltern**–Knoten
  Ein Eltern–Knoten zeichnet sich dadurch aus, daß er Nachfolgeknoten im Graphen besitzt, d.h. die Transition führt in einen abstrakten globalen Zustand, in dem weitere Transitionen schaltbar sind. In Abbildung 3.2 sind dies die Knoten $t_{0,0}, t_{1,1}, t_{2,2}, t_{3,3}$. Der Knoten $t_{1,4}$ ist kein Eltern–Knoten, da er keine Nachfolger besitzt.

- **Kind**–Knoten
  Ein Kind–Knoten ist ein Nachfolgeknoten eines Eltern–Knotens. In der Abbildung 3.2 sind dies die Knoten $t_{1,1}, t_{2,2}, t_{3,3}, t_{1,4}$.

- **Geschwister**–Knoten
  Zwei Knoten sind Geschwister, wenn sie den selben Eltern–Knoten besitzen, d.h. Geschwister sind im gleichen abstrakten globalen Zustand schaltbar. Die Knoten $t_{1,1}$ und $t_{3,3}$ aus Abbildung 3.2 sind Geschwister, da sie den Knoten $t_{2,2}$ als gemeinsamen Eltern–Knoten haben.

- **Root**–Knoten
  Der Root–Knoten ist der Anfangsknoten des Graphen. Er symbolisiert die Initialisierungsphase des Systems, d.h. im Root–Knoten werden die Initialisierungstransitionen des Systems zusammengefaßt. Der Knoten $t_{0,0}$ ist der Root–Knoten in Abbildung 3.2.

- **Pfad**
  Ein Pfad im Graphen ist eine Folge von Knoten und gerichteten Kanten, die auf dem Weg von einem Knoten zu einem Anderen benutzt werden. Der Pfad entspricht einer Teilfolge einer Berechnungsfolge. Pfade sind zum Beispiel in der Abbildung 3.2 die Folgen von Knoten $t_{1,1}, t_{2,2}, t_{1,1}$

- **Einfacher Pfad**
  Ein einfacher Pfad ist ein Pfad im Transitionsgraphen, in dem kein Knoten mehrfach vorkommt. Einfache Pfade sind in Abbildung 3.2 z.B. $t_{2,2}, t_{1,1}$ und $t_{1,1}, t_{2,2}, t_{3,3}, t_{1,4}$. Man beachte, daß auf einem einfachen Pfad eine Transition mehrfach vorkommen kann, wie zum Beispiel die Transition t_1. Die Knoten $t_{1,1}$ und $t_{1,4}$ repräsentieren verschiedene **Instanzen** einer Transition im Transitionsgraphen bzw. die bei einer Ausführung des Systems **geschalteten** Transitionen.

Der prinzipielle Aufbau des Graphen erfolgt dadurch, daß zuerst der initiale, abstrakte globale Zustand des Systems bestimmt wird. Dieser läßt sich anhand der Initialisierungstransitionen der einzelnen Estelle–Module bestimmen, indem die Hauptzustände und Zustandsvariablen auf die in den Initialisierungstransitionen angegebenen Werte gesetzt werden. Außerdem werden die den Interaktionpunkten zugeordneten Warteschlangen initialisiert.

Die Initialisierungstransitionen aller Modulinstanzen werden dann zu einem ersten Knoten, dem *Root–Knoten* zusammengefaßt, d.h. der Root–Knoten repräsentiert die Initialisierungstransitionen. Dieser Knoten enthält einen Verweis auf den initialen Zustand. Danach werden die im initialen Zustand schaltbaren Estelle–Transitionen bestimmt, für jede dieser Transitionen ein neuer Knoten mit dem jeweiligen Namen der Transition erzeugt und Verweise auf diese neuen Knoten in einer Liste für die Nachfolgeknoten des Root–Knotens zusammengefaßt. Damit ist die Initialisierungsphase des Transitionsgraphen abgeschlossen.

Aufbauend auf den schaltbaren Transitionen des Root–Knotens werden dann die nachfolgenden Knoten bzw. schaltbaren Transitionen und die durch das

Feuern der Transitionen erreichten abstrakten globalen Zustände des Transitionsgraphen bestimmt. Dies erfolgt nicht rekursiv und in drei Phasen:

Auswahl: Aus den Verweisen auf die nächsten zu schaltenden Transitionen des aktuellen Knotens wird sequentielle nach und nach eine noch nicht bearbeitete Transition ausgewählt und als Arbeitsknoten betrachtet.
Sind alle zu schaltenden Transitionen des aktuellen Knotens bereits bearbeitet, dann wird die Auswahl mit einem anderen Knoten im Graphen fortgesetzt. Dieser wird dann zum aktuellen Knoten erklärt.
Wurden alle im Graphen entstandenen Knoten bereits einmal zum aktuellen Knoten erklärt und bearbeitet, dann ist der Aufbau des Graphen vollendet und es kann mit der Analyse begonnen werden.

Feuern: Die ausgewählte Transition wird „gefeuert“. Dazu wird ein neuer abstrakter globaler Zustand durch Kopieren des bereits erreichten Zustandes kreiert. Danach werden die Auswirkungen der Transition auf die einzelnen Komponenten des neuen Zustandes bestimmt, d.h. es werden der Hauptzustand und die Zustandsvariablen des betroffenen Moduls aktualisiert und, wenn die Transition Ausgaben macht, die Nachrichten in die entsprechenden Warteschlangen eingereiht.
Dann findet eine Überprüfung des neuen Zustandes statt, d.h. es wird entweder der neue abstrakte Zustand dem Arbeitsknoten zugeordnet und mit der nächsten Phase, der Bestimmung der in dem neuen Zustand schaltbaren Transitionen weiter gemacht, oder es wird der Aufbau unterbrochen (s.u.). In diesem Fall wird der Funktion zur Behandlung der Unterbrechnung der aktuelle Knoten und der Arbeitsknoten übergeben. Nach Rückkehr aus der Funktion wird je nach Art der Unterbrechung der Arbeitsknoten mit einer Markierung versehen[6]. Der Aufbau des Graphen wird anschließend mit der Auswahlphase fortgesetzt.

Bestimmen: Die in dem neuen Zustand schaltbaren Transitionen werden bestimmt, für jede dieser Transitionen ein neuer Knoten mit dem Namen der jeweiligen Transition erzeugt und Verweise auf diese neuen Knoten in den Arbeitsknoten eingetragen. Außerdem

[6] siehe Kapitel 4

wird zur Beschleunigung der in Abschnitt 3.2 beschriebenen Analyse im Arbeitsknoten eine Markierung gesetzt, falls einer der neuen Knoten eine geschaltete Transition eines Features darstellt.

Dann beginnt der Zyklus von neuem mit der Auswahlphase.

Für die Unterbrechung des Zyklus muß eine der folgenden Bedingungen zutreffen:

- Der nach dem Feuern einer Transition erreichte neue abstrakte globale Zustand entspricht einem Null–Zustand[7]. Alle nachfolgenden Knoten sind dann für die Analyse ohne Bedeutung und brauchen deshalb im Graphen nicht zu erscheinen, so daß der Aufbau an einer anderen Stelle fortgesetzt werden kann.

- Das Schalten der Transition führt zu einem im Graphen bereits durch das Feuern einer andere Transition erreichten abstrakten globalen Zustand, d.h. es existiert im Graphen bereits ein Knoten, dessen abstrakter globaler Zustand identisch zum neuen abstrakten globalen Zustand ist.

  In diesem Fall wird die Aktualisierung des Arbeitsknotens in der Form durchgeführt, daß der Arbeitsknoten einen Verweis auf den bereits existierenden Zustand erhält, der neue Zustand gelöscht wird und die Liste der schaltbaren Transitionen des anderen Knotens durch einen Verweis dem Arbeitsknoten zugeordnet wird.

  Gilt zusätzlich, daß der Arbeitsknoten und der Knoten mit dem bereits exisitierenden Zustand gleich sind, d.h. sie repräsentieren zusätzlich noch die gleiche Transition, dann wird in der Liste der Nachfolger des aktuellen Knotens nach dem Verweis auf den Arbeitsknoten gesucht, dieser auf den Knoten mit dem bereits exisitierenden Zustand gesetzt und der Arbeitsknoten gelöscht.

  Zur Beschleunigung der Analyse wird weiterhin festgestellt, ob der aktuelle Knoten und der Arbeitsknoten gleich sind. In diesem Fall erfüllt die Transition, die der Knoten repräsentiert, die Bedingung des Major–State–Preservings des Minor–FI–Kriteriums aus Abschnitt 2.3.1. Dann wird der aktuelle Knoten entsprechend makiert.

[7] siehe Cross–Reset–Kriterium aus Abschnitt 2.3.2

- Die einem Interaktionspunkt zugeordnete Warteschlange kann keine Nachrichten mehr aufnehmen. Dadurch ist es nicht möglich, einen neuen Zustand zu generieren.

  Diese Bedingung ist wichtig, da sie garantiert, daß die Menge der im Transitionsgraphen vorhandenen Knoten endlich ist[8].

- Eine spontane Transition bzw. eine Transition mit `Delay`–Klauseln wurde hintereinander n mal ausgewählt, d.h. es gibt einen einfachen Pfad, so daß n mal dieselbe Transition nacheinander für den Übergang von einem Zustand zum Nächsten verantwortlich ist. n ist dabei eine durch den Benutzer vorgegebene Konstante[9].

  Durch diese Bedingung ist eine Beschneidung des Graphen möglich, wenn zum Beispiel das mehrmalige Feuern derselben spontanen Transition in Folge nur die Warteschlangen füllt, aber ansonsten sich die entstehenden Teilgraphen nach einmaligen, zweimaligen usw. Feuern der Transition nicht unterscheiden.

- Die vorgegebene maximale Pfadlänge eines einfachen Pfades wurde erreicht. Dies beschränkt die Tiefe des Transitionsgraphen.

- Die vorgegebene maximale Anzahl von Knoten im Graphen wurde erreicht. Der Graph gilt dann als aufgebaut. Es wird aber in diesem Fall nicht sofort zur Analyse übergegangen, sondern es werden die noch nicht ausgewählten aber bereits kreierten Knoten markiert und erst dann zur Analysephase übergegangen.

Es sei angemerkt, daß beim Aufbau jedem Knoten eine eindeutige Kennung (**Knoten–ID**) zur Unterscheidung zugeordnet wird. Dadurch lassen sich die verschiedenen Instanzen von einer Transition aus der Spezifikation im Transitionsgraphen unterscheiden.

3.1.3 Alternatives Verfahren zum Aufbau des Transitionsgraphen

Ein alternatives Verfahren für den Aufbau und die Analyse wäre, daß jede Berechnungsfolge aufgebaut, mittels der FI–Kriterien überprüft und wieder gelöscht wird. D.h. ausgehend vom initialen abstrakten globalen Zustand

[8] siehe Abschnitt 3.1.1 über die Erweiterungen des abstrakten globalen Zustandes.

[9] siehe Abschnitt 3.5 über die Zwischenform.

bzw. vom Root–Knoten werden die nachfolgenden Knoten bzw. Transitionen bestimmt, und rekursiv von jedem von diesen aus, diese Bestimmung für alle Nachfolgeknoten durchgeführt. Die Rekursion bricht jeweils ab, wenn ein Null–Zustand erreicht wurde.

Der Nachteil dieses Algorithmus ist, daß Knoten mehrmals entstehen, überprüft und wieder gelöscht werden. Dies geschieht zum Beispiel, wenn von einem Knoten bis zu einem seiner Nachfolgeknoten sich die entsprechenden Teilfolgen der zugehörigen Berechnungsfolgen unterscheiden, die weiteren Teilfolgen der Berechnungsfolgen dann aber identisch sind.

Tatsächlich handelt es sich bei diesem Algorithmus um ein Backtracking–Verfahren. Aufgrund des exponentiellen Laufzeitverhaltens bzgl. der Anzahl von Knoten und der Anzahl von Verzweigungen ([Sed95]) ist dieser Algorithmus für die Analyse ungeeignet.[10].

Damit Knoten nur einmal erzeugt und möglichst wenige male mit dem in Abschnitt 3.2 beschriebene Suchverfahren mit seinem wesentlich besserem Laufzeitverhalten[11] untersucht werden müssen, wird der Transitionsgraph vollständig im Speicher aufgebaut. Dadurch ist es im Gegensatz zum Backtracking–Verfahren möglich, bereits im Graphen existierende, durch das Schalten einer Transition erreichte, abstrakte globale Zustände wiederzuerkennen.

3.2 Analyseverfahren

In diesem Abschnitt wird das im Werkzeug benutzte Verfahren zur Erkennung und Einstufung einer Feature–Interaktion mittels der FI–Kriterien beschrieben.

Zum besseren Verständnis wird im folgenden ein Knoten, der eine Instanz einer beliebigen Estelle–Transition t repräsentiert, ebenfalls mit t bezeichnet. Weiterhin werden die Eigenschaften zwischen Transitionen wie Indeterminismus etc. entsprechend auf die Knoten des Graphen angewendet, da diese Instanzen von Transitionen darstellen.

Voraussetzung für die Analyse ist, daß der Transitionsgraph im Speicher aufgebaut ist.

[10] In einer ersten Version einer Testspezifikation für ein Telefonsystem ([Mi96]) ist ein Graph mit mehr als 100000 Knoten vom Werkzeug aufgebaut worden.

[11] Eine einfache Tiefensuche kann in linearer Zeit bzgl. der Knoten und Kanten durchgeführt werden ([Sed95]).

Das Verfahren besteht aus drei Phasen:

- **Vorbereitungsphase**:
  In der Vorbereitungsphase werden für die spätere Analyse einige interne Objekte zum Speichern der Ergebnisse bzw. von Zwischenergebnissen erzeugt und initialisiert.

  Abschnitt 3.2.3 behandelt diese Vorbereitungsphase.

- **Paar–Analysephase**:
  Die erste Phase der Analyse durchläuft einmal den Transitionsgraphen und sucht nach Indeterminismen zwischen jeweils zwei verschiedenen Transitionen. Dabei werden diese Paare mittels der FI–Kriterien auf verschiedene Listen verteilt, so daß die Paare bzw. die Knoten in den Paaren für die Weiterverarbeitung in der nächsten Analysephase nach Feature–Interaktionen gruppiert bereit stehen.

  Zum Beispiel muß für die Anwendung des Minor–FI–Kriteriums auf ein Paar (f, b) von Knoten, deren Transitionen die Bedingung des Major–State–Preservings erfüllen, anschließend festgestellt werden, mit welchen anderen von dem Knoten b aus erreichbaren Instanzen von Transitionen möglicherweise eine Feature–Interaktion vorliegen könnte[12].

- **Pfad–Analysephase**:
  Während der Pfad–Analyse wird hauptsächlich für die Knoten bzw. Paare der Listen aus der vorherigen Analysephase bestimmt, welche Knoten von einem Knoten aus erreichbar sind, und dann mittels dieser Informationen entschieden, welche Feature–Interaktion tatsächlich vorliegt.

Die Aufteilung der Analyse in zwei Phasen ist nötig, da für eine Instanz f einer Transition aus einem Feature F (d.h. für einen Knoten im Graphen) Berechnungsfolgen im Transitionsgraphen existieren können, so daß mehrere FI–Kriterien angewendet werden müssen.

Dies ist z.B. der Fall, wenn eine Instanz für eine Transition b im Graphen zwei Geschwister–Knoten als Instanzen der Transitionen f und g hat, so daß f und b als auch g und b das Basiskriterium 3 erfüllen, dann müßte für den Knoten b zweimal eine Tiefensuche gestartet werden. Durch Trennung in die zwei Phasen kann dies verhindert werden.

[12] siehe Abbildung 3 aus Abschnitt 2.3.2.

Während der Analysephasen werden die Ergebnisse einer Analyse durch CONFINE (siehe Abschnitt 2.6 bzw. [Th95]) und die Präzedenzen (siehe Abschnitt 2.5 bzw. [BrGo94a], [BrGo94b]) für Entscheidungen bzgl. Indeterminismus bzw. fehlerhafte Auflösung einer Feature–Interaktion mittels Präzedenzen herangezogen.

Die Ergebnisse und Präzedenzen werden vor dem Start eingelesen und in Form von Matrizen im Speicher gehalten.

Die Ergebnisse von CONFINE werden als Matrix über die Estelle–Transitionen benötigt, so daß ein Flag in der durch Zeile und Spalte definierten Zelle der Matrix einen *möglichen* Indeterminismus zwischen den entsprechenden Transitionen anzeigt.

Die Präzedenzmatrix ist entsprechend eine Matrix über die Features, so daß eine Markierung in einer Zelle angibt, daß das Feature in der Zeile eine höhere Präzedenz als das Feature in der Spalte aufweist.

Damit die Vorbereitungsphase und die Analysephasen schnell durchgeführt werden können, wird vor der eigentlichen Analyse vom Werkzeug sichergestellt, daß

- jede Transition eine eindeutige Kennung, **Transitions–ID** erhält,
- jede Variable, jeder Nachrichtenparameter und jede Nachricht der Spezifikation eine eindeutige Kennung **Variablen–ID** erhalten,
- die Output–Dependency–Pattern[13] in Form von Matrizen über die Elemente des erweiterten Zustandsraumes, getrennt nach den Transitionen, vorliegen[14]. Die Zeilen und Spalten der Matrizen repräsentieren dabei die Variablen–ID's. Eine Markierung in einer Zelle dieser Matrix besagt, daß die Variable in der Zeile von der Variablen in der Spalte abhängig ist.

Bevor die Beschreibung der wichtigen Teile der einzelnen Phasen erfolgt, wird in Abschnitt 3.2.2 der Basisalgorithmus — *die Tiefensuche* — zum Durchsuchen des Graphens vorgestellt.

Die Beschreibung der Algorithmen erfolgt umgangssprachlich und ist auf das Wesentliche beschränkt. Zu den Details der Implementation sei auf den C++–Quelltext verwiesen.

[13] siehe Abschnitt 2.3.3

[14] Es werden nur die Elemente des erweiterten Zustandsraumes betrachtet, da die Änderungen des Hauptzustandes bzw. der Zustandsvariablen mit der Zuweisung einer Konstanten gleichzusetzen ist (siehe Abschnitt 2.3.3).

Zum besseren Verständnis des Analyseverfahrens wird im folgenden Abschnitt 3.2.1 ein Ausschnitt aus einem möglichen Transitionsgraph mit einigen Situationen für Feature–Interaktionen vorgestellt.

3.2.1 Ein Beispiel für Feature–Interaktionen im Transitionsgraphen

Die in Abbildung 3.3 auf Seite 42 gezeigten Situationen für potentielle Feature–Interaktionen in einem Transitionsgraphen sind durch Kombination der einzelnen FI–Kriterien aus Abschnitt 2.3 entstanden.

Diese Situationen beschreiben damit Berechnungsfolgen im Transitionsgraphen und sind Beispiele für Feature–Interaktionen. Die Situationen müssen vom Werkzeug erkannt und entsprechend ihrer „Gefährlichkeit“ eingestuft werden.

Die Knoten in der Abbildung 3.3 sind mit einem Namen für die jeweilige Transition gekennzeichnet. Dieser Name setzt sich aus dem Namen des Features und einem Index für die entsprechende Estelle–Transition aus der Spezifikation (Transitions–ID) zusammen.
b_i bezeichnet eine Transition aus dem Basissystem, alle anderen Knotennamen sind Transitionen aus zum Basissystem hinzugefügten Features.

Die Kanten sind mit den Namen der jeweiligen abstrakten globalen Zustände versehen, in denen die Transition zu den jeweiligen Knoten (auf den die Kante verweist) schaltbar ist.

Bei den dargestellten Feature–Interaktionen wird davon ausgegangen, daß an den entsprechenden Stellen die Bedingung des Indeterminismus zwischen zwei Transitionen erfüllt ist.

Die in Abbildung 3.3 auf Seite 42 exemplarisch gezeigten Situationen und einige Erläuterungen zu den potentiellen Feature–Interaktionen sind:

S1: In dieser ersten Situation erfüllen die Knoten f_8 und b_1 die Bedingung des Major–State–Preserving, so daß alle von f_8 aus erreichbaren Knoten möglicherweise das Minor–FI–Kriterium erfüllen.

Erfüllt ein erreichbarer Knoten j_{12} das Minor–FI–Kriterium nicht, d.h. es besteht eine Datenabhängigkeit zwischen f_8 und j_{12}, dann handelt es sich um eine Feature–Interaktion nach Basiskriterium 1 aus Abschnitt 2.3.1.

Man beachte, daß durch die Berücksichtigung von Nachrichten bzw. von Zustandsvariablen beim Aufbau des Graphen die Bedingung des Major–State–Preservings gegenüber der Definition in [Bre95] erweitert werden muß, da diese Definition außer dem Hauptzustand zunächst keine weiteren Komponenten des Zustandsraumes berücksichtigt.

Die Idee für die Bedingung des Major–State–Preservings ist, daß zwei Transitionen im gleichen abstrakten globalen Zustand schaltbar werden , und daß das Schalten die beiden Transitionen in einen (für beide Transitionen gleichen) abstrakten globalen Zustand führt.

Werden nun Nachrichten und Zustandsvariablen berücksichtigt, dann sind zwei abstrakte globale Zustände genau dann gleich, wenn die Hauptzustände, die Zustandsvariablen und die Warteschlangen der Interaktionspunkte gleich sind.

Die Bedingung des Major–State–Preservings wird im Transitionsgraphen zu der Bedingung, daß ein Knoten des Basissystems und eine Knoten eines hinzugefügten Features einen gemeinsamen Eltern–Knoten besitzen und daß ihre Kinder–Knoten genau die gleichen sind.

Man beachte, daß die Bedingung des Major–State–Preservings auch erfüllt ist, wenn ein Knoten direkt zu sich selbst zurückführt, d.h. er ändert keine Komponente des abstrakten globalen Zustandes. Durch die Makierung beim Aufbau des Transitionsgraphen[15] kann die Bedingung ohne großen Aufwand geprüft werden.

S2: Die zweite Situation beschreibt den Fall, daß die beiden Knoten f_8 und g_9 die Bedingung des Independent–FI–Kriteriums nicht erfüllen, da die Reihenfolge für das Schalten der beiden entsprechenden Transitionen nicht beliebig sein kann. Dies wird durch den Übergang von f_8 zu b_2 verursacht.

Der Übergang von Knoten f_8 zu b_2 kann erfolgen, wenn zum Beispiel f_8 und g_9 jeweils zwei verschiedene Nachrichten Msg_1 und Msg_2 an ein anderes Estelle–Modul versenden. Kann aus diesem anderen Modul die Transition b_2 nur dann schalten, wenn die Nachricht Msg_1 vor der Nachricht Msg_2 eintrifft, dann ergibt sich die dargestellte Situation im Transitionsgraphen.

Ohne den Übergang von Knoten f_8 zu b_2 ist es möglich, daß die beiden Knoten f_8 und g_9 die Bedingung des Independent–FI–Kriteriums

[15] siehe Abschnitt 3.1.2

erfüllen. Dazu müssen die entsprechenden Transitionen auf disjunkten Teilmengen über die Elemente des Zustandsraumes arbeiten.

Dieser Fall bedeutet nicht, daß die Knoten b_1 und g_9 zwangsläufig das Independent–FI–Kriterium erfüllen müssen, da zum Beispiel zwischen den beiden Knoten eine direkte Datenabhängigkeit über Elemente des erweiterten Zustandsraumes bestehen kann, so daß die Bedingung des Independent–FI–Kriteriums nicht erfüllt wird.

S3: In dieser Situation erfüllen die beiden Knoten h_{10} und i_{11} das Independent–FI–Kriterium, weil die Reihenfolge, wie dargestellt, beliebig ist.

Allerdings kann durch die Schleife eine indirekte Datenabhängigkeit zwischen h_{10} und i_{11} auf dem Pfad $(h_{10}, i_{11}, b_4, b_2, i_{11})$ bestehen, so daß h_{10} und i_{11} möglicherweise das Minor–FI–Kriterium erfüllen können.

Anschaulich kann man dies dadurch erläutern, daß man die Schleife expandiert, d.h. die Folgen von Knoten in einer Schleife beliebig oft aneinander hängt.

S4: Die Situation (S4) stellt eine „gefährliche“ Feature–Interaktion nach Basiskriterium 1 dar, weil die Berechnungsfolgen mit den Knoten k_{14} und l_{15} von dem Knoten j_{12} abhängig sind.

S5: Diese Situation ist eine „gefährliche“ Feature–Interaktion nach Basiskriterium 2, da ein Indeterminismus zwischen den Knoten k_{14} und l_{15} besteht.

S6: Situation (S6) erfüllt das Basiskriterium 3, d.h. durch das Hinzufügen des Features G werden die Berechnungsfolgen für die Features J und L verändert.

Für die Analyse bedeutet dies, daß von dem Knoten b_7 ausgehend alle erreichbaren Knoten gefunden werden müssen. Diejenigen Knoten bzw. Transitionen, die aus einem hinzugefügten Feature stammen, bestimmen dann, mit welchem Feature das Feature G interagiert.

Man beachte, daß für den Knoten g_9 ebenfalls die Menge der von ihm aus erreichbaren Knoten bestimmt werden muß, da möglicherweise eine Feature–Interaktion nach Basiskriterium 1 existieren kann. Als Beispiel sei die Interaktion zwischen Feature G und Feature J genannt, da g_9 und j_{12} auf einem Pfad liegen.

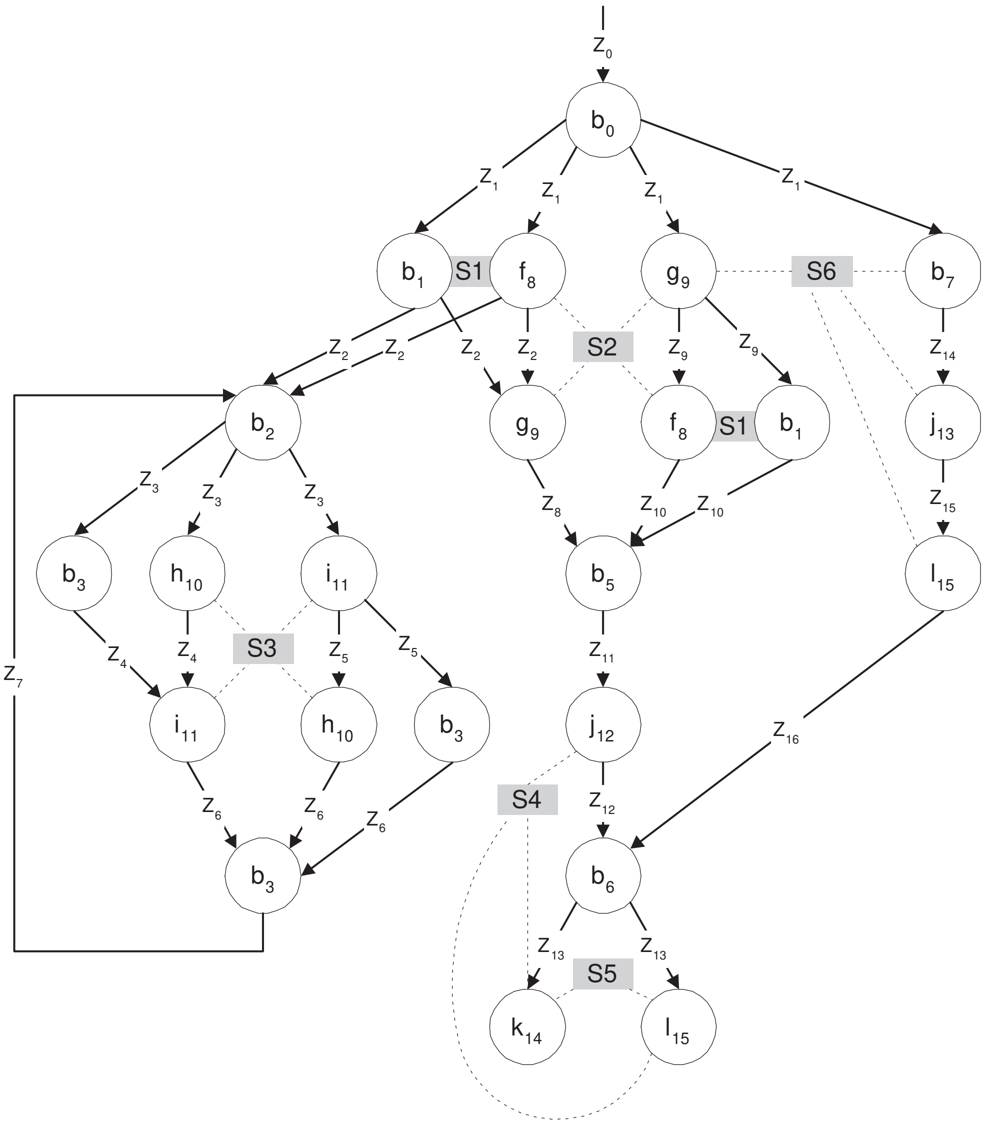

Abbildung 3.3: Beispiel für zu analysierende Situationen im Transitionsgraphen.

Weitere Situationen entstehen, wenn Präzedenzen zwischen Features eingeführt werden. In diesem Fall müssen die Auswirkungen von Präzedenzen betrachtet werden[16].

Da Präzedenzen über die Vergabe von Prioritäten für Estelle–Transitionen realisiert sind, können die Präzedenzen eine Vorrangbeziehung nur zwischen Transitionen *eines* Estelle–Moduls definieren. Dies hat bei der Betrachtung eines in Estelle spezifizierten Systems mit mehreren Estelle–Modulen zur Folge, daß eine Auflösung von Feature–Interaktionen durch Präzedenzen nicht zwischen Transitionen der verschiedenen Estelle–Module stattfinden kann.

Die Auflösung eines Indeterminismus mittels Präzedenzen ist deshalb nicht ohne weiteres auf ein System mit mehreren Estelle–Modulen (insbesondere nicht auf ein verteiltes System) zu übertragen.

Als Beispiel nehme man an, daß für die Features B, G und J in Abbildung 3.3 die Präzedenzen $G \succ B$ und $G \succ J$ definiert sind. Weiterhin sei der Knoten j_{13} bzw. die entsprechende Transition aus einem anderen Estelle–Modul als die Transitionen für die Knoten g_9 und b_7. Dann besteht eine Feature–Interaktion zwischen G und J aufgrund von Basiskriterium 3, obwohl die Präzedenz $G \succ J$ besteht. Damit hat die Auflösung des Indeterminismus zwischen den Knoten g_9 und b_7 mittels einer Präzedenz keinen Einfluß auf den Knoten j_{13}, d.h. die Feature–Interaktion zwischen G und J bleibt bestehen.

Für diesen Fall meldet das Werkzeug eine Feature–Interaktion zwischen g_9 und j_{13} nach Basiskriterium 3 *im Zusammenhang* mit Präzedenzen. Für den anderen Fall, daß die Transitionen aus dem selben Estelle–Modul stammen und die Auflösung wegen der Präzedenz $J \succ G$ nicht erfolgreich ist, wird eine Feature–Interaktion *aufgrund* von Präzedenzen zwischen g_9 und j_{13} gemeldet. Weiterhin wird vom Werkzeug ausgegeben, daß die Auflösung mittels Präzedenzen zwischen g_9 und b_7 *nicht erfolgreich* war.

Zusammenfassend ist zu sagen, daß für jede Transition, die aus einem hinzugefügten Feature stammt, das Basiskriterium 1 angewendet werden muß. Die erkannten Feature–Interaktionen können dann möglicherweise mit dem Minor–FI–Kriterium bzgl. ihrer „Gefährlichkeit“ eingestuft werden. Dies ergibt sich aus den Situationen bzw. aus den FI–Kriterien und wird während der Analysephase berücksichtigt.

[16]siehe Abschnitt 2.5 über Präzedenzen.

3.2.2 Tiefensuche

Die Tiefensuche ist ein einfacher Algorithmus zur rekursiven Analyse eines Graphen ([Sed95]).

Im Werkzeug wird eine nicht–rekursive Variante der Tiefensuche angewendet. Diese benutzt einen **Stack**, zwei `While`–Schleifen (eine **äußere** `While`–Schleife und eine **innere** `While`–Schleife) und ein Feld `aVisitedField` zur Markierung der bereits bearbeiteten Knoten. Die Indizes des Feldes sind äquivalent zur eindeutigen Kennzeichnung der Knoten mittels der **Knoten-ID**.

Die wesentlichen Funktionen, die im Algorithmus benutzt werden, sind:

`push()` Damit wird auf dem Stack ein Element, d.h. ein Zeiger auf einen Knoten des Transitionsgraphen, abgelegt.

`pop()` Mit `pop()` wird das oberste Element vom Stack geholt.

`visit()` In dieser Funktion findet die eigentliche Verarbeitung der Elemente statt. Die Funktion wird mit dem zu verarbeitenden Element aufgerufen. Nach Beendigung der Funktion wird der Knoten durch einen Eintrag in `aVisitedField` als bearbeitet markiert.

`visitRoot()` Diese Funktion ist für eine Verarbeitung des ersten Knotens gedacht, für den eine Tiefensuche stattfinden soll.

Wenn zum Beispiel alle Nachfolger eines Knotens bestimmt werden sollen, dann müssen entsprechend die Kinder des Knotens auf den Stack gelegt werden.

Man beachte: Wenn ein Knoten einen Verweis auf sich selbst hat, dann ist er ein Kind zu sich selbst und wird entsprechend auf den Stack gelegt. Andernfalls wird der Knoten nicht auf den Stack gelegt und auch nicht als bearbeitet markiert, da er Teil einer Schleife über mehrere Knoten sein könnte.

Diese Funktion legt mindestens einen Knoten auf den Stack ab.

Die genannten Funktionen ermöglichen eine hohe Flexibilität und Wiederverwendbarkeit des Algorithmus. Tatsächlich enthält das Werkzeug eine

C++–Klasse mit dem eigentlichen Tiefensuch–Algorithmus und den „Visit"–Funktionen als virtuelle Member–Funktionen, so daß durch Vererbung und Überladen der „Visit"–Funktionen der Graph nach verschiedenen Gesichtspunkten abgesucht werden kann, ohne den eigentlichen Tiefensuch–Algorithmus ändern zu müssen.

Die notwendigen Argumente für die Tiefensuche können als Member–Elemente in der jeweiligen abgeleiteten C++–Klasse abgelegt werden, so daß die überladenen „Visit"–Funktionen darauf zugreifen können.

Gestartet wird die Tiefensuche, indem ein Knoten (der sogenannte **Start–Knoten**) der `deepSearch()`–Funktion der C++–Klasse übergeben wird. Von diesem Start–Knoten ausgehend werden dann alle erreichbaren Knoten im Graphen nacheinander besucht.

Der Algorithmus arbeitet folgendermaßen:

1. Übergebe den Start–Knoten an die `visitRoot()`–Funktion, so daß diese mindestens einen ersten Knoten mit der Funktion `push()` auf den Stack ablegen kann.

2. <u>Äußere–While–Schleife</u>:
   Solange sich ein Knoten auf dem Stack befindet:

   2.1. Hole das oberste Element mit `pop()` vom Stack, erkläre es zum aktuellen Knoten.

   2.2. Überprüfe: Ist der aktuelle Knoten schon besucht worden, d.h. ist eine Markierung im Feld `aVisitedField` unter der Knoten–ID für den aktuellen Knoten gesetzt ?

   JA: Gehe zum Beginn der äußeren Schleife, (2.).

   2.3. Übergebe den aktuellen Knoten zur Verarbeitung der `visit()`–Funktion.

   2.4. Setze eine Markierung unter der Knoten–ID des aktuellen Knotens in `aVisitedField`.

   2.5. Setze einen Zeiger auf den ersten Nachfolgeknoten des aktuellen Knotens.
   Falls kein Nachfolgeknoten existiert, wird der Zeiger automatisch auf `NULL` gesetzt.

   2.6. <u>Innere–While–Schleife</u>:
   Solange der Zeiger ungleich `NULL` ist:

2.6.1. Lege den Nachfolgeknoten mit `push()` auf den Stack ab.

2.6.2. Setze den Zeiger auf den nächsten Nachfolgeknoten aus der Liste der als nächstes schaltbaren Knoten des aktuellen Knotens.
Falls das Ende der Liste erreicht ist, wird der Zeiger automatisch auf `NULL` gesetzt.

2.6.3. Gehe zum Beginn der inneren Schleife, (2.6.).

2.7. Gehe zum Beginn der äußeren Schleife, (2.).

3. Setze alle Einträge in `aVisitedField` zurück, so daß eine neue Tiefensuche beginnen kann.

Die Bearbeitung findet in der Reihenfolge Eltern–Knoten → Kinder–Knoten statt, d.h. in Form einer Bearbeitung „in der Tiefe“[17].

Benutzt man eine Queue statt eines Stacks, wird eine Bearbeitung in der Breite durchgeführt, d.h. die Eltern und deren Geschwister–Knoten werden vor den Kinder–Knoten bearbeitet[18].

Die Tiefensuche garantiert, daß alle Knoten des Graphen genau einmal in der `visit()`–Funktion verarbeitet werden. Insbesondere bedeutet dies, daß alle Knoten in einer Schleife genau einmal verarbeitet werden. Dies liefert uns *einfache Pfade*[19].

Der wesentliche Vorteil des Tiefensuch–Algorithmus ist, daß die Laufzeitkomplexität des Algorithmus proportional zur Summe aus Anzahl der Knoten N und Anzahl der Kanten E im Graphen ist, d.h. für die Laufzeitkomplexität gilt $O(N+E)$ ([Sed95]). Dies folgt aus der Tatsache, daß die Knoten und Kanten nur einmal besucht werden.

3.2.3 Vorbereitungsphase

Die Vorbereitungsphase dient dazu, einige interne Datenstrukturen aufzubauen, so daß die für die Analysephasen benötigten Informationen effizient zugreifbar werden. Außerdem müssen einige Ergebnisse der ersten Analysephase für eine Weiterverarbeitung in der zweiten Phase zwischengespeichert werden. Dabei wird an einigen Stellen ein erhöhter Speicherbedarf zugunsten eines effizienteren Zugriffs in Kauf genommen.

[17] Diese Vorgehensweise entspricht der „Pre–Order Traversierung“ eines Baumes.
[18] Dies ist analog zur „Level–Order Traversierung“ eines Baumes.
[19] D.h. Knoten können nicht mehrmals auf einem Pfad vorkommen, siehe Abschnitt 3.1.2

Die benötigten Objekte sind:

- Die Matrix `IndependentMatrix`, deren Zeilen und Spalten die Transitionen der Estelle–Spezifikation repräsentieren. Eine Zeile der Matrix gibt an, mit welchen anderen Transitionen (Spalten) die Bedingung des Independent–FI–Kriteriums erfüllt wird.

  Dazu werden für alle möglichen Paare von Transitionen die Schnittmenge zwischen dem Output–Pattern der einen und dem Input–Pattern der anderen Transition und die Schnittmenge zwischen den beiden Output–Pattern der Transitionen des Paares gebildet.

  Falls diese beiden Schnittmengen leer sind, wird in den Zeilen und Spalten der Matrix `IndependentMatrix` eine Markierung angebracht, d.h. die beiden Transitionen erfüllen die Bedingung des Independent–FI–Kriteriums.

  Zur Bildung der Schnittmengen müssen

  - die Hauptzustände,
  - die Zustandsvariablen,
  - die Interaktionspunkte, an denen eine Nachricht empfangen bzw. an denen Nachrichten in anderen Modulen empfangen werden können, und
  - die Output–Dependency–Pattern für die Elemente des erweiterten Zustandsraumes

  geprüft werden.

  Die Output–Dependency–Pattern der jeweiligen Transitionen liegen als $V \times V$ Bit–Matrizen über den Variablen, Nachrichten und Nachrichtenparametern der Spezifikation vor, so daß über einfache Matrizenoperationen die Schnittmengen gebildet werden können. V gibt dabei die Gesamtanzahl der Variablen, Nachrichten und Nachrichtenparameter an.

  Die anderen Elemente des Zustandsraumes müssen explizit auf eine Überschneidung geprüft werden. Zum Beispiel: Stammen die beiden Transitionen aus dem selben Modul und ändert eine davon den Hauptzustand oder eine Zustandsvariable ?

  Man beachte, daß die Bedingung für das Independent–FI–Kriterium zwischen zwei Transitionen s und t keine symmetrische Relation darstellt. Wenn zum Beispiel die Schnittmenge zwischen dem Output–Pattern von s und dem Input–Pattern von t leer ist, dann folgt nicht, daß die Schnittmenge des Input–Patterns von s mit dem Output–Pattern von t auch leer ist. Deshalb müssen für die Matrix `IndependentMatrix` entsprechend beide Schnittmengen gebildet werden.

Der Fall, daß eine Transition mit sich selbst die Bedingung des Kriteriums erfüllt, braucht nicht berücksichtigt zu werden, weil aufgrund des Verfahrens beim Aufbau des Transitionsgraphen eine Instanz einer Transition niemals eine weitere Instanz der selben Transition als Geschwister–Knoten haben kann. Dies ergibt sich aus der Interleaving–Semantik von Estelle.

Insgesamt müssen für T Transitionen in der Spezifikation $T^2 - T$ Paare gebildet werden, so daß als Abschätzung eine Laufzeitkomplexität für die Bildung der Matrix von $O((T^2 - T) * (m(V, V) + H))$ entsteht.

Faktor $m(V, V)$ gibt die Komplexität für den Vergleich der Zellen zwischen den Output–Dependency–Pattern Matrizen an, d.h. er bestimmt die Anzahl der Operationen auf den Matrizen. Faktor H beschreibt die Komplexität für die Bestimmung von eventuellen Überschneidungen zwischen Hauptzuständen, Zustandsvariablen etc.

Aus der o.g. Komplexitätsabschätzung läßt sich folgern, daß die Transitionen und die Elemente des erweiterten Zustandsraumes die Komplexität quadratisch beeinflussen, während die Überprüfung der Hauptzustände, Zustandsvariablen und Interaktionspunkte linear eingeht.

- Ein Feld `FeatureMaskField` als Maske zum schnellen Ausmaskieren von Transitionen des Basissystems.

  Die Feldelemente sind jeweils einer Transition der Spezifikation zugeordnet, so daß mittels der Transitions–ID auf ein Feldelement zugegriffen werden kann. Weiterhin wird in jedem Element des Feldes eine Markierung gesetzt, wenn die zugehörige Transition aus einem hinzugefügten Feature stammt.

  Die Laufzeitkomplexität für die Initialisierung dieses Feldes ist $O(T)$, wenn T die Anzahl der Transitionen darstellt. Dies ist im Vergleich zur Bildung der Matrix `IndependentMatrix` vernachlässigbar.

- Eine Sammel–Liste `CollectList`, in der Zeiger auf alle Knoten im Graphen gesammelt werden, für welche eine Tiefensuche gestartet werden muß.
  Dabei wird darauf geachtet, daß die Knoten genau einmal in dieser Liste vorhanden sind.

- Eine Liste `Base1List` mit Zeigern auf Knoten. Für diese Knoten genügt eine einfache Tiefensuche und die Überprüfung, welche Features erreichbar sind.

  Diese Liste repräsentiert die Knoten bzw. Transitionen, die nach dem Basiskriterium 1 aus Abschnitt 2.3.1 erkannt werden und nicht als weniger „gefährlich" eingestuft werden können.

- Eine Liste `Base3List` mit Paaren von Zeigern auf Knoten. Für diese Paare muß das Basiskriterium 3 aus Abschnitt 2.3.1 angewendet werden.

- Eine Liste `MajorList`, die Zeiger auf alle diejenigen Knoten enthält, für welche eine Datenabhängigkeitsanalyse durchgeführt werden muß, um das Minor–FI–Kriterium benutzen zu können. Alle Knoten in der Liste erfüllen die Bedingung des Major–State–Preserving.

- Eine Liste `PrecedenceList`, in der Paare von Zeigern auf Knoten abgelegt werden, zwischen denen eine Präzedenz besteht. Die Auswirkungen der Präzedenzen[20] müssen für diese Paare von Knoten ermittelt werden.

- Schließlich wird eine Resultats–Matrix `ResultMatrix` benötigt, in der das Ergebnis der Analyse festgehalten wird.

  Die Zeilen und Spalten repräsentieren die Transitionen der Estelle–Spezifikation.

  In die Matrix wird eingetragen, zwischen welchen Transitionen eine Feature–Interaktion besteht und welches FI–Kriterium dies entdeckt bzw. als weniger "gefährlich" eingestuft hat. Jede Zelle der Matrix enthält dazu ein Feld. Ist ein Element dieses Feldes markiert, dann hat dies die Bedeutung, daß Situationen im Transitionsgraphen existieren, in denen die beiden Transitionen die entsprechende Feature–Interaktion aufweisen.

Die Laufzeitkomplexität der Vorbereitungsphase wird im wesentlichen durch die Komplexität für die Bestimmung der Matrix `IndependentMatrix` bestimmt. Die Laufzeitkomplexität für die anderen Objekte ist im Vergleich zu der Komplexität für die Bestimmung der Matrix `IndependentMatrix` vernachlässigbar, da die Objekte nur fest initialisiert werden.

[20] siehe Abschnitt 2.5 bzw. Abschnitt 3.2.1

3.2.4 Erste Analysephase: Paar–Analyse

Die erste Phase der Analyse führt genau eine Tiefensuche auf dem Transitionsgraphen durch und sucht nach Situationen, in denen ein Indeterminismus zwischen zwei Transitionen vorliegt.

Dazu werden zwischen den Kindern eines Knotens alle möglichen Paare gebildet und auf diese Paare die FI–Kriterien angewendet. Die Reihenfolge innerhalb der Paare spielt dabei keine Rolle.

Daraus ergibt sich, daß für einen Knoten mit durchschnittlich k Kindern insgesamt $\binom{k}{2} = \frac{(k-1)*k}{2}$ Paare von Transitionen gebildet werden müssen. Wenn der Transitionsgraph N Knoten hat, dann sind für den gesamten Graphen $N * \frac{(k-1)*k}{2}$ Paare zu bilden.

Je nachdem welche Situation für ein Paar vorliegt, werden die Knoten aus dem Paar oder das Paar selbst auf die Listen aus dem vorherigen Abschnitt aufgeteilt und damit einer Weiterverarbeitung in der zweiten Analysephase zugeführt.

Die Behandlung der Paare von Transitionen wird in der `visit()`–Funktion des Tiefensuch–Algorithmuses erledigt.

Bevor der Kern dieser Funktion näher beschrieben wird, einige allgemeine Bemerkungen zum Algorithmus:

- Die Tiefensuche wird mit dem Root–Knoten[21] des Transitionsgraphen gestartet. Dieser wird in der `visitRoot()`–Funktion auf den Stack gelegt.

- Jeder noch nicht besuchte Knoten des Graphen wird der `visit()`–Funktion übergeben. Die Anwendung der Kriterien erfolgt dann auf die Kinder des übergebenen Knotens.

- Hat ein Knoten genau ein Kind und ist dieses Kind eine Instanz einer Transition aus einem Feature[22], dann wird ein Zeiger auf das Kind in die Listen `Base1List` und in `CollectList` eingefügt.

- Hat der übergebene Knoten kein Kind, da zum Beispiel während der Aufbauphase des Graphen der abstrakte globale Null–Zustand erreicht worden ist, dann wird mit einem anderen Knoten im Graphen weitergearbeitet.

[21]siehe Abschnitt 3.1.2
[22]damit ist hier und im folgenden nicht das Basissystem–Feature gemeint

- Die möglichen Kombinationen von Knoten in einem Paar sind (f, b) und (f, g). f und g repräsentieren Transitionen aus verschiedenen Features F und G und b eine Transition des Basissystems B.

  Die Kombinationen (f_i, f_j) bzw. (g_i, g_j) und (b_i, b_j) für $i \neq j$ brauchen nicht untersucht zu werden, da zwischen Transitionen eines Features per Definition keine Feature–Interaktion besteht.

  Weiterhin gilt, daß die Reihenfolge in einem Paar keine Rolle spielt, also das Paar (f, g) anlog zum Paar (g, f) behandelt wird.

- Das Independent–FI–Kriteriums wird auch auf Paare (f, b) angewendet, wenn die Bedingung des Major–State–Preservings zwischen den beiden Transitionen nicht zutrifft.

- Die Entscheidung, wann ein Knoten entweder in die Liste `Base1List` oder in die Liste `MajorList` eingefügt werden kann, ist erst möglich, wenn alle Paare unter den Kindern des übergebenen Knotens behandelt worden sind. Erst dann ist sicher zu entscheiden, ob die Bedingung des Major–State–Preservings erfüllt wird und welches der beiden Kriterien Anwendung finden muß.

  Deshalb werden die in Frage kommenden Knoten in einer temporären Liste `TempList` abgelegt und nach Bearbeitung aller Paare auf die Liste `Base1List` bzw. auf die Liste `MajorList` verteilt.

Im Folgenden wird die Arbeit des Kerns der `visit()`–Funktion vorgestellt. Zum Zeitpunkt des Aufrufs dieser Funktion ist die Paarbildung mit zwei Kinder-Knoten bereits erfolgt. Außerdem sind die Kinder–Knoten, die einen Verweis auf sich selbst besitzen, und die Kinder–Knoten, die als einziges Kind eines Eltern–Knotens existieren, bereits in die Listen `MajorList` bzw. `Base1List` und in `CollectList` eingefügt worden.

Die einzelnen Abschnitte des Kerns sind zum besseren Verständnis entsprechend der möglichen Kombinationen der Transitionen in einem Paar aufgeteilt. Die Reihenfolge, in der die Transitionen in einem Paar auftauchen können, spielt keine Rolle.

(f, b): Besteht zwischen f und b ein Indeterminismus ?

  JA: 1. Ist eine Präzedenz $F \succ B$ bzw. $B \succ F$ definiert ?

    JA: Füge das Paar in die Liste `PrecedenceList` und den Zeiger auf b in die Liste `CollectList` ein und gehe zu (4)

2. Sind f und b Major–State–Preserving, d.h. haben sie die gleichen Kinder–Knoten ?

   JA: Setze im Knoten für die Transition f das Flag `S_MAJOR`, nimm den Knoten f in die Liste `MajorList` auf und gehe zu (4).

3. Erfüllen die beiden Transitionen die Bedingungen des Independent–FI–Kriteriums ?

   JA: Markiere die beiden Transitionen in der Matrix `ResultMatrix` als „Independent“.

   NEIN: Füge das Paar in die Liste `Base3List` und den Zeiger auf b in die Liste `CollectList` ein.

4. Füge den Zeiger auf den Knoten für die Transition f in die Listen `TempList` und `CollectList` ein.

NEIN: Es liegt keine Interaktion vor.

Gehe zur Bearbeitung des nächsten Paares über.

(f, g): Besteht zwischen f und g ein Indeterminismus ?

JA:

1. Ist eine Präzedenz $F \succ G$ bzw. $G \succ F$ definiert ?

   JA: Füge das Paar in die Liste `PrecedenceList` ein und gehe zu (3).

2. Erfüllen die beiden Transitionen die Bedingungen des Independent–FI–Kriteriums ?

   JA: Markiere die beiden Transitionen in der Matrix `ResultMatrix` als „Independent“.

   NEIN: Markiere in der Resultats–Matrix die beiden Transitionen als Feature–Interaktion, basierend auf dem Basiskriterium 2 aus Abschnitt 2.3.1. Außerdem füge das Paar in die Liste `Base3List` ein.

   Gehe zu (3)

3. Füge die Zeiger auf die Knoten für die Transitionen f und g in die Listen `TempList` und `CollectList` ein.

NEIN: Es liegt keine Interaktion vor.

Gehe zur Bearbeitung des nächsten Paares über.

Der wesentliche Unterschied zwischen den beiden Paaren (f, b) und (f, g) liegt darin, daß für (f, b) zusätzlich die Bedingung Major–State–Preserving des Minor–FI–Kriteriums überprüft werden muß.

Bevor die `visit()`–Funktion verlassen wird, müssen die in der temporären Liste `TempList` angesammelten Knoten auf eine der beiden Listen `MajorList` und `Base1List` verteilt werden.

Ein Knoten wird der Liste `MajorList` hinzugefügt, wenn das Flag `S_MAJOR` im Knoten gesetzt ist. Andernfalls wird der Knoten in die Liste `Base1List` eingefügt.

Zum genauen Ablauf dieses Teils der Analyse sei auf den Quelltext verwiesen.

Nach Beendigung der Tiefensuche gilt:

- Die Liste `CollectList` enthält *alle* Knoten des Transitionsgraphen, von denen aus eine Tiefensuche gestartet werden muß.
- Auf alle Knoten in der Liste `Base1List` muß das Basiskriterium 1 angewendet werden, d.h. diese Knoten können mit ihren Nachfolgerknoten eine „gefährliche“ Feature–Interaktion bilden.
- Die Knoten für das Minor–FI–Kriterium sind in der Liste `MajorList`.
- Die Paare von Knoten, für welche das Basiskriterium 3 angewendet werden muß, sind in der Liste `Base3List`.
- Die Liste `PrecedenceList` enthält alle Paare von Knoten bzw. Transitionen zwischen denen eine Präzedenz definiert ist.
- Alle Paare von Knoten bzw. Transitionen, zwischen denen ein Indeterminismus derart besteht, daß das Basiskriterium 2 Anwendung findet, oder die das Independent–FI–Kriterium erfüllen, sind in der Matrix `ResultMatrix` markiert.
- Das Cross–Reset–Kriterium wird bereits beim Aufbau des Graphen behandelt.
- Die Knoten können in mehreren der Listen enthalten sein. Zum Beispiel wenn f mit b das Basiskriterium 3 erfüllt, dann ist f in der Liste `Base3List` und in der Liste `Base1List` enthalten, da f mit einer anderen Transition g eine Feature–Interaktion nach Basiskriterium 1 haben kann.

Die Laufzeitkomplexität dieser Phase der Analyse wird im wesentlichen durch die Laufzeitkomplexität der Tiefensuche (d.h. die Zeit bis alle Knoten des Graphen besucht worden sind) und der Komplexität für die Bildung und Untersuchung der Paare über die Kinder eines Knotens in der `visit()`–Funktion bestimmt.

Ist N die Anzahl von Knoten, E die Anzahl der Kanten im Graphen und k die durchschnittliche Anzahl von Kindern für einen Knoten des Graphen, dann ist die Komplexität der Tiefensuche $O(N + E)$ und die Komplexität für die Paare $O(N * \frac{(k-1)*k}{2})$.

Zusammen ergibt sich für diese Analysephase als Abschätzung eine Lauzeitkomplexität von $O((N + E) + \frac{1}{2} * N * k * (k - 1))$.

3.2.5 Zweite Analysephase: Pfad–Analyse

In der zweiten Phase werden die in der ersten Phase in den verschiedenen Listen gesammelten Knoten entsprechend der jeweiligen FI–Kriterien weiterverarbeitet. Im folgenden werden die Ideen, wie diese Weiterverarbeitung prinzipiell stattfindet, kurz vorgestellt.

Für fast alle Knoten aus den Listen ist es notwendig, eine Tiefensuche durchzuführen, so daß die von einem Knoten aus erreichbaren Nachfolgeknoten bekannt sind.

Damit für jeden Knoten aus der Liste genau eine Tiefensuche durchgeführt wird, sind während der ersten Analysephase die Knoten genau einmal in die Liste `CollectList` eingefügt worden.

Die Ergebnisse der Tiefensuchen werden in einer Matrix (`ReachableMatrix`) abgelegt, so daß über die Knoten–ID eines Knotens (Zeilen–Index) auf die Menge der erreichbaren Transitionen zugegriffen werden kann. Als Spalten–Index dient die Transitions–ID.

Diese Matrix enthält damit für jeden Knoten der Liste `CollectList` welche Estelle–Transitionen[23] im Graphen von diesem Knoten aus erreichbar sind.

Die Laufzeitkomplexität für die Erstellung der Matrix wird durch die Anzahl C von Knoten in der Liste `CollectList` und der Komplexität für eine Tiefensuche im Transitionsgraphen $O(N + E)$ bestimmt, so daß sich eine Laufzeitkomplexität von $O(C * (N + E))$ für die Erstellung der Matrix ergibt.

[23] nicht die Knoten

Man beachte: Eine Spalte der Matrix `ReachableMatrix` gibt diejenigen Knoten an, von denen ein Pfad zu der durch die Spalte bestimmten Estelle–Transition führt.

Dies sind natürlich nicht alle Knoten, von denen aus eine Instanz einer Transition erreichbar ist. Es sind aber mindestens alle Knoten, die Instanzen der Transitionen aus den hinzugefügten Features bilden. Dies wird durch die erste Analysephase gesichert.

Mit Hilfe des Spaltenverktors der Matrix `ReachableMatrix` kann eine Überprüfung der indirekten Datenabhängigkeit zwischen zwei Transitionen aus einem hinzugefügten Feature erfolgen, wenn die beiden Transitionen eine dritte Transition beeinflussen. Siehe dazu auch Abschnitt 2.3.3 und [Bre95].

Ist die Matrix `ReachableMatrix` erstellt, dann kann die Liste `CollectList` gelöscht werden, da sie für die weitere Verarbeitung nicht mehr benötigt wird.

Die weiteren Verarbeitungsschritte dieser Phase werden im folgenden entsprechend der Listen aufgeführt.

Base1List

Für die Knoten dieser Liste liegt eine Feature–Interaktion gemäß Basiskriterium 1 vor.

Damit das FI–Kriterium für einen Knoten effizient entschieden werden kann, wird mittels der Knoten–ID die entsprechende Zeile aus der Matrix `ReachableMatrix` mit der Maske `FeatureMaskField`[24] verknüpft. Bei der Verknüpfung der Zeile mit der Maske handelt es sich um eine „UND“–Operation, d.h. wenn ein Feldelement aus der Zeile und das entsprechende Feldelement aus der Maske jeweils eine Markierung besitzen, dann erhält das Ergebnis ebenfalls eine Markierung.

Das Ergebnis der „UND“–Operation wird dann in die Matrix `ResultMatrix` übertragen, d.h. in der Zeile für die Transition werden für alle markierten Feldelemente des Ergebnisses eine Markierung für eine Feature–Interaktion basierend auf dem Basiskriterium 1 eingefügt.

Dieser Arbeitsschritt wird für jeden der Knoten aus der Liste ausgeführt.

Die Liste kann danach gelöscht werden.

[24]Die Maske gibt an, welche Transitionen aus einem Feature stammen. Für die Transitionen des Basissystems existierten in der Maske keine Markierungen.

Die Komplexität für die Abarbeitung der Liste beträgt $O(R * m(1,T))$, wobei R die Anzahl der Knoten in der Liste, T die Anzahl der Transitionen der Spezifikation und $m(1,T)$ die Anzahl der notwendigen Operationen auf einer Zeile der $N \times T$ Matrix `ReachableMatrix` für einen Knoten ist.

Base3List

Für die Paare (f,t) mit $t \in \{b,g\}$ von Knoten bzw. Transitionen dieser Liste liegt möglicherweise eine Situation entsprechend des Basiskriteriums 3 vor, d.h. es muß festgestellt werden, welche Transitionen aus den hinzugefügten Features auf dem Pfad von der Transition t aus erreicht werden können.

Exemplarisch wird die Vorgehensweise für das Paar (f,b) beschrieben. Für das Paar (f,g) erfolgt die Bestimmung analog.

Für die Feststellung, welche Transitionen von dem Knoten b aus erreicht werden können, wird aus der Matrix `ReachableMatrix` für den Knoten b die entsprechende Zeile geholt. Diese Zeile wird dann mit dem Feld `FeatureMaskField` über eine „UND"–Operation verknüpft, so daß das Ergebnis der Verknüpfung angibt, welche Transitionen aus hinzugefügten Features stammen. Zwischen diesen Transitionen und der Transition f besteht dann eine Feature–Interaktion gemäß Basiskriterium 3.

In der Zeile für Transition f der Matrix `ResultMatrix` werden diese Transitionen als Feature–Interaktionen nach Basiskriterium 3 markiert.

Dieser Vorgang wird für jedes Paar aus der Liste ausgeführt.

Die Liste kann danach gelöscht werden.

Wenn F die Anzahl der Paare in der Liste, T die Anzahl von Transitionen in der Spezifikation und $m(1,T)$ die Anzahl der notwendigen Operationen für eine Zeile der $N \times T$ Matrix `ReachableMatrix` für ein Paar ist, dann ergibt sich eine Abschätzung der Laufzeitkomplexität für diesen Teil der Analyse zu $O(F * m(1,T))$.

PrecedenceList

Für ein Paar (u,v)[25] dieser Liste muß die Auswirkung der Einführung von Präzedenzen untersucht werden. Im folgenden habe o.B.d.A. das Feature U eine höhere Präzedenz gegenüber dem Feature V.

[25] Transition $u, v \in \{b, f, g\}$ und $u \neq v$

Man beachte, daß dabei untersucht werden muß, ob die von Knoten v aus erreichbaren Transitionen aus dem selben oder einem anderen Estelle–Module wie die Transition des Knotens u stammen[26].

Für die Untersuchung wird ausgehend von Knoten v die erreichbaren Transitionen bestimmt. Dies erfolgt mit der Matrix `ReachableMatrix`. Als Index in die Zeile wird die Knoten–ID der Transition v aus dem Paar benutzt.

Für jede Markierung in dieser Zeile wird mittels der Präzedenzmatrix für die entsprechende Transition bzw. das Feature, aus dem sie stammt überprüft, ob U eine höhere Präzedenz besitzt.

Wird festgestellt, daß U keine höhere Präzedenz besitzt und die erreichte Transition aus dem gleichen Estelle–Modul stammt, dann wird in der Zeile für u und der Spalte für die erreichte Transition in der Matrix `ResultMatrix` festgehalten, daß es ein Problem bei der Auflösung *aufgrund* von Präzedenzen gegeben hat.

Stammen die erreichte Transition und der Knoten u bzw. dessen Transition nicht aus dem selben Modul, dann wird in die entsprechende Zelle der Matrix `ResultMatrix` eingetragen, daß es ein Problem bei der Auflösung *im Zusammenhang* von Präzedenzen zwischen der Transition u und den erreichten Transitionen gegeben hat.

Weiterhin wird in den beiden Fällen in die Matrix für u und v eingetragen, daß die Auflösung *nicht erfolgreich* war.

Es werden alle Markierungen in der Zeile für v aus der Matrix `ReachableMatrix` überprüft. Dieser Vorgang wird für jedes Paar aus der Liste ausgeführt.

Die Liste kann danach gelöscht werden.

Man beachte, daß das Basissystem B eine höhere Präzedenz gegenüber einem hinzugefügten Feature F besitzen kann.

Die Laufzeitkomplexität für die Bestimmung der Auswirkungen von Präzedenzen beträgt $O(P * m(1,T))$, wobei P die Anzahl der Paare in der Liste, T die Anzahl der Transitionen der Spezifikation und $m(1,T)$ die Anzahl der notwendigen Operationen für eine Zeile der $N \times T$ Matrix `ReachableMatrix` ist.

[26] siehe Abschnitt 3.2.1

MajorList

Für die Knoten dieser Liste gilt, daß sie die Bedingung des Major–State–Preserving erfüllen.

Für die Einstufung als weniger „gefährliche“ Feature–Interaktion mit dem Minor–FI–Kriterium muß eine Überprüfung auf indirekte Datenabhängigkeiten der Knoten mit den von diesem Knoten aus erreichbaren Knoten bzw. Transitionen durchgeführt werden.

Das Problem ist, daß eine indirekte Datenabhängigkeit zwischen zwei Knoten bzw. den Transitionen nur dann exakt entschieden werden kann, wenn die *Reihenfolge*, in der die Knoten auf einem *Pfad* auftauchen, berücksichtigt wird.

Im schlimmsten Fall müssen alle möglichen Pfade von dem einen Knoten zu den erreichbaren Knoten betrachtet werden.

Da aber im Graphen Schleifen vorkommen und Knoten mehrere direkte Nachfolger haben können, die später im Pfad wieder zu einem Knoten zusammengeführt werden können, ist es zu aufwendig, alle Pfade in angemessener Zeit zu durchlaufen.

Das Problem mit den Schleifen läßt sich zwar beheben, indem man für Knoten innerhalb einer Schleife die maximal möglichen Abhängigkeiten bestimmt. In diesem Fall wird jedoch nicht berücksichtigt, daß keine Abhängigkeiten mit Variablen mehr bestehen, wenn auf eine Variable eine Konstante zugewiesen wird. Dies führt i.a. zu einer sehr ungenauen Aussage über indirekte Datenabhängigkeit, speziell wenn alle Knoten des Graphen Teil einer einzigen Schleife sind.

Zuletzt bleibt noch das Problem, daß Teilpfade in einem Knoten zusammengeführt werden können. Dieses Problem ließe sich mit einem Backtracking–Verfahren lösen. Dies ist aber aufgrund der exponentiellen Laufzeitkomplexität bzgl. der Größe des Graphen nicht für Graphen mit vielen tausenden von Knoten und Kanten realistisch durchführbar.

Deshalb wird im Werkzeug die Suche nach den tatsächlichen Datenabhängigkeiten zwischen zwei Transitionen bzw. Knoten auf einem Pfad zur Suche nach den *prinzipiell möglichen* Abhängigkeiten abgeschwächt, d.h. nach der Frage, ob es eine Datenabhängigkeit zwischen den Transitionen auf einem Pfad gibt, ohne die Reihenfolge der Transitionen auf dem Pfad zu berücksichtigen.

Die Bestimmung der prinzipiell möglichen Abhängigkeiten zwischen mehreren Transitionen erfolgt, indem ein Graph (**Abhängigkeitsgraph**) über die Elemente der Output–Dependency–Pattern[27] aufgebaut wird.

Es werden nur die Elemente des erweiterten Zustandsraumes betrachtet, weil ansonsten zwischen zwei Transitionen zum Beispiel sofort über den Hauptzustand eine Datenabhängigkeit bestehen könnte, wenn zwei Transitionen bzw. ihre Instanzen im Transitionsgraphen auf einem Pfad liegen, die Transitionen aus dem selben Estelle–Modul stammen und eine der Transitionen den Hauptzustand ändert.

Die Knoten des Abhängigkeitsgraphen sind die Variablen, Nachrichten etc. der Spezifikation. Die gerichteten Kanten des Graphen verbinden zwei Knoten, wenn eine entsprechende Abhängigkeit zwischen den Variablen etc. durch ein Element des Output–Dependency–Pattern gegeben ist. Zusätzlich werden die Kanten mit der Transitions–ID der Transition benannt, aus dem das für den Übergang verantwortliche Element stammt.

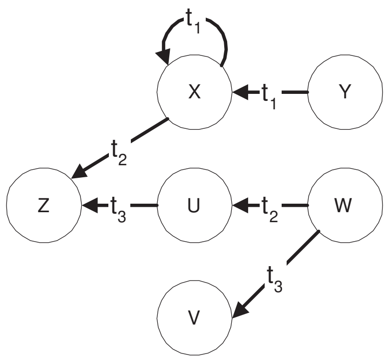

Abbildung 3.4: Abhängigkeitsgraph

Abbildung 3.4 zeigt einen Abhängigkeitsgraphen für drei Transitionen mit folgenden Output–Dependency–Pattern:

t_1: { x$\leftarrow\{x, y\}$ }

t_2: { z$\leftarrow\{x\}$, u$\leftarrow\{w\}$ }

t_3: { z$\leftarrow\{u\}$, v$\leftarrow\{w\}$ }

[27] siehe Abschnitt 2.3.3

Für das Beispiel aus Abbildung 3.4 folgt, daß zwischen den Transitionen (t_1, t_2) und (t_2, t_3) eine prinzipiell mögliche Abhängigkeit besteht. Zwischen den Transitionen (t_1, t_3) ist keine Abhängigkeit möglich. Dabei wird davon ausgegangen, daß die Instanzen der Transitionen t_2 und t_3 von einer Instanz der Transition t_1 im Transitionsgraphen erreichbar sind.

Damit reduziert sich die Bestimmung der Erfüllbarkeit des Minor–FI–Kriteriums auf die Bestimmung der von einem Knoten (im folgenden als **Start–Knoten** bzw. **Start–Transition** bezeichnet) aus erreichbaren Knoten bzw. Transitionen und der Bestimmung der prinzipiell möglichen Datenabhängigkeiten zwischen den beteiligten Transitionen.

Die von einem Knoten aus erreichbaren Transitionen sind unter der Knoten–ID in der Matrix `ReachableMatrix` zusammengefaßt.

Die Frage nach den prinzipiell möglichen Datenabhängigkeiten zwischen Transitionen auf einem Pfad im Transitionsgraphen wird über eine Tiefensuche über die Kanten des Abhängigkeitsgraphen beantwortet. Dazu muß geprüft werden, ob einfache Pfade im Abhängigkeitsgraphen existieren mit den Bedingungen,

- daß ausgehend von allen Kanten im Abhängigkeitsgraphen, die mit der Transitions–ID der Start–Transition gekennzeichnet sind, Kanten mit den Kennzeichnungen der anderen Transitionen erreicht werden können, und
- daß als Kanten nur diejenigen benutzt werden dürfen, die mit einer entsprechenden Transitions–ID der vom Start–Knoten aus im Transitionsgraphen erreichbaren Transitionen gekennzeichnet sind.

Die Auswahl der entsprechenden Kanten im Abhängigkeitsgraphen und das Festhalten, welche Transitionen eine Abhängigkeit besitzen, erfolgt in der `visit`–Funktion für die Tiefensuche im Abhängigkeitsgraphen.

Der Aufbau des Abhängigkeitsgraphen ist mit den Informationen aus der Zwischenform[28] ohne große Probleme möglich, da alle Kanten für diesen Graphen bereits implizit über die Elemente der Output–Dependency–Pattern angegeben sind.

Eine Schwäche des Verfahrens ist, daß die Reihenfolge der Knoten auf einem Pfad im Transitionsgraphen nicht beachtet wird, d.h. für einen Pfad mit

[28] siehe Abschnitt 3.5 bzw. 4.1.1

der Folge von Transitionen (t_1, t_2, t_3) kann eine Datenabhängigkeit zwischen t_1 und t_3 bestehen, obwohl die Transition t_2 zum Beispiel alle Variablen zurücksetzt.

Das Verfahren garantiert aber, daß alle (unter Berücksichtigung der Reihenfolge) tatsächlich vorhandenen Datenabhängigkeiten zwischen Transitionen erkannt werden.

Ein wesentlicher Vorteil des Verfahrens ist, daß die Laufzeitkomplexität einer Tiefensuche auf dem Abhängigkeitsgraphen um Größenordnungen kleiner als die Laufzeitkomplexität $O(N + E)$ einer Tiefensuche im Transitionsgraphen ist.

Ist V die Anzahl für die Variablen, Nachrichten etc. und p die durchschnittliche Anzahl der Elemente der Output–Dependency–Pattern in der Spezifikation, dann ergibt sich die Laufzeitkomplexität zu $O(V + T * p)$, wenn T die Anzahl der Transitionen darstellt.

Der Ablauf des Verfahrens im Werkzeug gestaltet sich nun folgendermaßen:

- Jeder Knoten der Liste `MajorList` wird nacheinander zu einem Start–Knoten erklärt.

- Für jeden dieser Start–Knoten werden mittels der Matrix `ReachableMatrix` die jeweils im Transitionsgraphen erreichbaren Transitionen ermittelt.

- Dann wird mit diesen Start–Knoten bzw. deren Transitions–ID und den Transitions–ID's der von einem Start–Knoten aus erreichbaren Transitionen eine Tiefensuche im Abhängigkeitsgraphen durchgeführt.

  Das Ergebnis dieser Tiefensuche wird in einer Matrix `IndirektDepMatrix` abgelegt, deren Zeilenindex die Knoten–ID der Knoten im Transitionsgraphen und deren Spaltenindex die Transitions–ID repräsentiert, d.h. die Matrix gibt an, zwischen welchen Transitionen eine prinzipiell mögliche Datenabhängigkeit besteht.

  Diese Matrix ist notwendig, da eine Transition b des Basissystems von Transitionen (z.B. f und g) aus hinzugefügten Features beeinflußt werden kann, so daß eine indirekte Datenabhängigkeit zwischen f und g vorhanden ist[29].

[29] siehe Abschnitt 2.3.3 über Datenabhängigkeit zwischen Transitionen

- Sind alle Knoten der Liste bearbeitet, d.h. die Matrix `IndirektDepMatrix` wurde aufgebaut, wird jede Spalte der Matrix `ReachableMatrix`, die eine Transition b_i des Basissystems repräsentiert, herangezogen.

- Für jeden Eintrag in dieser Spalte, d.h. für einen Knoten im Transitionsgraphen, wird in der Matrix `IndirektDepMatrix` unter der Knoten–ID nachgeschaut, ob eine prinzipiell mögliche Datenabhängigkeit besteht.

  Ist eine prinzipiell mögliche Datenabhängigkeit vorhanden, dann wird in einem temporären Feld über die Transitions–ID's eine Markierung gesetzt.

- Sind nach Abarbeitung einer Spalte der Matrix `ReachableMatrix` in dem temporären Feld mehr als zwei Markierungen für Transitionen aus hinzugefügten Features gesetzt, dann besteht zwischen diesen Transitionen eine indirekte Datenabhängigkeit. Dieses Ergebnis wird in eine temporäre Matrix übertragen, deren Zeilen– und Spaltenindizes den Transitions–ID's entsprechen.

- Sind alle Spalten der Matrix `ReachableMatrix` abgearbeitet, dann stehen in der temporären Matrix und der Matrix `IndirektDepMatrix` die potentiell möglichen Datenabhängigkeiten zwischen den Transitionen.

  Die Zeilen der beiden Matrizen werden dann in die Matrix `ResultMatrix` übertragen, d.h. es werden entsprechende Markierungen für die Paare von Transitionen, die das Minor–FI–Kriterium *sicher* erfüllen bzw. die eine *prinzipell mögliche* Datenabhängigkeit besitzen, angebracht.

Eine Abschätzung der Laufzeitkomplexität für die Bearbeitung der Knoten dieser Liste ergibt sich zu $O(D*(V+T*p)*m(N,T))$, wobei D die Anzahl der Knoten der Liste, $O(V+T*p)$ die Komplexität der Suche im Abhängigkeitsgraphen, T die Anzahl der Transitionen, N die Anzahl der Knoten im Transitionsgraphen und $m(N,T)$ die Anzahl der benötigten Operationen auf den $(N \times T)$ Matrizen angibt.

3.2.6 Abschätzung der Laufzeitkomplexität der Analyse

Für eine Abschätzung der Laufzeitkomplexität der Analyse werden folgende Faktoren benötigt:

N: Anzahl der Knoten im Transitionsgraphen

E: Anzahl der Kanten im Transitionsgraphen

$k = \frac{N}{E}$: Durchschnittliche Anzahl von Kinder–Knoten bzw. **Verzweigungsgrad** des Transitionsgraphen.
Man beachte, daß der Wert für k besser wird, je mehr Knoten keine Kinder haben. Dies ist besonders dann der Fall, wenn der Transitionsgraph nicht vollständig aufgebaut werden kann.

T: Anzahl von Transitionen in der Spezifikation

V: Anzahl der Komponenten des erweiterten Zustandsraumes, d.h. Anzahl von Variablen, Nachrichten, Nachrichtenparameter usw.

p: Durchschnittliche Anzahl der Elemente der Output–Dependency–Pattern

C: Anzahl der Knoten in der Liste `CollectList`.

L: Anzahl der Knoten in den Listen `Base1List`, `Base3List` und `PrecedenceList`.

D: Anzahl von Knoten in der Liste `MajorList`.

$m(V, V)$: Als Abschätzung für die Anzahl der Operationen zum Aufbau der Matrix `IndependentMatrix` aus der Vorbereitungsphase, Abschnitt 3.2.3.

H: Als Abschätzung der benötigten Überprüfungen der Hauptzustände, Zustandsvariablen und Interaktionspunkte beim Aufbau der Matrix `IndependentMatrix`.

$m(1, T)$: Als Abschätzung der Anzahl von Operationen für einen Knoten bzw. für ein Paar von Knoten aus den Listen `Base1List`, `Base3List` und `PrecedenceList`.

$m(N, T)$: Als Abschätzung der Anzahl von Operationen für einen Knoten aus der Liste `MajorList`.

Damit ergibt sich als Abschätzung für die einzelnen Teile des Analyseverfahrens:

1. Für den Aufbau der Matrix `IndependentMatrix` aus der Vorbereitungsphase:
$$O((T^2 - T) * (m(V, V) + H)$$
2. Für die Bestimmung der Paare in der ersten Analysephase:
$$O((N + E) + N * \frac{(k-1)*k}{2})$$
3. Für die Bearbeitung der `CollectList`, d.h. für die insgesamt durchzuführenden Tiefensuchen im Transitionsgraphen:
$$O(C * (N + E))$$
4. Für die Bearbeitung der Knoten bzw. Paare von Knoten in den Listen `Base1List`, `Base3List` und `PrecedenceList`:
$$O(L * M(T, T))$$
5. Für die Erkennung von indirekter Datenabhängigkeit:
$$O(D * (V + T * p) * M(N, T))$$

Unter den Bedingungen und Annahmen

- $T \ll N$, d.h. die Anzahl der Instanzen von Transitionen im Transitionsgraphen ist sehr viel größer als die Anzahl der Transitionen,
- $C \leq N$, d.h. es muß maximal von jedem Knoten aus eine Tiefensuche gestartet werden,
- $D \leq \frac{N}{2}$, d.h. es können maximal $\frac{N}{2}$ Knoten die Bedingung des Major–State–Preserving erfüllen,
- $L \leq N$, d.h. es können maximal N Knoten in einer Liste auftauchen,
- daß die Laufzeitkomplexitäten der Operationen auf den Matrizen vernachlässigbar klein sind[30] und
- daß die Laufzeitkomplexität einer Tiefensuche im Transitionsgraphen weit größer als die einer Tiefensuche im Abhängigkeitsgraphen ist[31]

[30] Die meisten der Operationen können auf Vergleiche von Bit–Pattern zurückgeführt werden.

[31] weil die Anzahl der Kanten und Knoten im Abhängigkeitsgraphen wesentlich geringer als im Transitionsgraphen ist.

gilt, daß die „worst–case“ Laufzeitkomplexität der Analyse im wesentlichen durch die Laufzeitkomplexität

$$O(N * (N + E))$$

für die Tiefensuchen im Transitionsgraphen in der zweiten Analysephase bestimmt wird.

Das folgende Beispiel soll dies verdeutlichen:

- Annahme:
  Sei die Anzahl der Knoten im Transitionsgraphen $N = 100.000$ und der durchschnittliche Verzweigungsgrad $k = 10$
- Dann ergibt sich für $N * (N + E) = N^2 * (1 + k)$ mit $E = N * k$, daß im „worst–case“ Fall jeder Knoten und jede Kante im Transitionsgraphen $1,1 * 10^{11}$ mal besucht werden wird.
- Werden für das Besuchen eines Knotens bzw. einer Kante zwei Mikrosekunden benötigt, dann wird für diesen Teil der Analyse eine Zeit von ca. 60 Stunden benötigt.

3.3 Auswirkungen von Nebenläufigkeit zwischen Estelle–Modulen auf die Analyse

Die durchschnittlichen Anzahl k von Kindern eines Knotens bestimmt die Anzahl der zu analysierenden Situationen im Graphen, da jedes Paar (bzw. ein Knoten des Paares), dessen Knoten aus zwei verschiedenen Features stammen, in mindestens eine der Listen aufgenommen werden muß.

Für fast alle in den Listen aufgenommenen Knoten muß eine Tiefensuche im Transitionsgraphen durchgeführt werden[32].

Die durchschnittliche Anzahl k von Kindern besagt, daß es im Durchschnitt k Möglichkeiten in einem Knoten gibt, einen nächsten Knoten zu wählen, d.h. man hat im Durchschnitt k Möglichkeiten nach Erreichen eines abstrakten globalen Zustandes die Ausführung mit dem Schalten einer Transition fortzusetzten.

Damit kann man k auch als Maß für Indeterminismus bzw. Nebenläufigkeit ansehen.

[32] D.h. der Knoten ist der Liste `CollectList` hinzugefügt worden.

Da ein verteiltes System in Estelle durch asynchron zueinander laufende Estelle–Module spezifiziert wird und diese Asynchronität durch Nebenläufigkeit in Estelle dargestellt wird, kann man prinzipiell sagen, daß sich das Laufzeitverhalten der Analyse verbessert, je besser die einzelnen Estelle–Module synchronisiert werden.

Die Synchronisation von Estelle–Modulen erfolgt in der Regel durch Austausch von Nachrichten. Deshalb werden Nachrichten beim Aufbau des Transitionsgraphen berücksichtigt[33].

Man beachte, daß die Synchronisation der Estelle–Module durch Nachrichten im wesentlichen die Transitionen mit `When`–Klauseln beeinflußt, während spontane Transitionen und Transitionen mit `Delay`–Klauseln im abstrakten globalen Automaten nur über den Hauptzustand bzw. die Zustandsvariablen beeinflußt werden.

Wird zum Beispiel in einer Spezifikation für ein Telefonsystem eine spontane Transition zur Simulation dafür eingesetzt, daß der Benutzer den Telefonhörer jederzeit auflegen kann, oder daß er auch während eines Gespräches jederzeit eine Taste am Telefon drücken kann, dann wird diese spontane Transition in fast jedem Knoten des Transitionsgraphen eine Verzweigung zu einer Instanz der spontanen Transition erzeugen.

Allgemein gilt, daß, sobald ein Knoten erreicht ist, in dem spontane Transitionen bzw. Transitionen mit `Delay`–Klauseln als nächstes schaltbar werden, für diesen Knoten und alle seine Nachfolgeknoten jeweils eine Verzweigung zu einer Instanz dieser Transitionen gebildet werden muß, bis die Bedingungen für das Schalten der spontanen Transitionen bzw. der Transitionen mit `Delay`–Klauseln nicht mehr erfüllt sind.

Dies entspricht der Interleaving–Semantik von Estelle.

Aufgrund dieser Semantik können Teilpfade im Transitionsgraphen entstehen, deren Knoten die Bedingungen für das Schalten der spontanen Transitionen bzw. Transitionen mit `Delay`–Klauseln nicht beeinflussen. Jeder Knoten dieser Teilpfade hat dann Verzweigungen, die zu Instanzen der spontanen Transitionen bzw. Transitionen mit `Delay`–Klauseln führen.

Dies bedeutet, daß der Verzweigungsgrad k und die Anzahl N von Knoten des Transitionsgraphen durch die Anzahl der spontanen Transitionen und Transitionen mit `Delay`–Klauseln in der Spezifikation wesentlich beeinflußt werden.

[33]siehe Abschnitt 3.1.1.

Deshalb sind Maßnahmen zur Eindämmung der durch spontane Transitionen und Transitionen mit `Delay`–Klauseln hervorgerufene Erhöhung des Verzeigungsgrades im Transitionsgraphen (also der Nebenläufigkeit) notwendig. In Abschnitt 4.3 sind zwei Erweiterungsmöglichkeiten zur Eindämmung der Nebenläufigkeit bzw. der Anzahl von Knoten im Transitionsgraphen angegeben.

3.4 Anforderungen an einen Spezifikationsstil in Estelle

Damit der Transitionsgraph aus Abschnitt 3.1 aufgebaut und die Analyse durchgeführt werden kann, sind einige Anforderungen an den Spezifikationsstil notwendig.

Einige der Anforderungen ergeben sich aus den Randbedingungen aus Abschnitt 2.4.

Die wesentliche Randbedingung ist die Notwendigkeit, eine *Wertverfolgung* für die Elemente des erweiterten Zustandsraumes zu vermeiden, damit das Problem einer möglichen Zustandsexplosion verhindert werden kann.

Die wesentlichen Anforderungen an eine Spezifikation sind:

1. Für den Aufbau des Graphen ist ein definierter abstrakter globaler Anfangszustand notwendig, d.h. die Initialisierungstransitionen der Estelle–Module müssen die Hauptzustände und alle Zustandsvariablen der einzelnen Module entsprechend initialisieren.

   Die Elemente des erweiterten Zustandsraumes müssen nicht initialisiert werden, da für sie keine Wertverfolgung stattfindet.

2. Damit der Aufbau des Transitionsgraphen erfolgen kann und damit während des Aufbaues effizient bestimmt werden kann, an welche Estelle–Module eine Transition eine Nachricht versendet, ist es in diesem ersten Ansatz zur Implementierung der FI–Kriterien erforderlich, eine triviale Verbindungsstruktur zwischen den einzelnen Estelle–Modulen zu spezifizieren.

   Trivial bedeutet in diesem Zusammenhang, daß für jede Verbindung eine eigene `Connect`–Anweisung benutzt wird und daß in den `Connect`–Anweisungen jeder Interaktionspunkt eines Moduls über die zugehörige Modulvariable explizit referenziert wird. D.h. für jede Instanz eines Moduls muß eine eigene Modulvariable angelegt werden.

In einem zweiten Ansatz könnte man die Anforderung an eine triviale Verbindungsstruktur durch eine einfache, statische[34] Verbindungsstruktur ersetzen, da es durch die statische Modulstruktur[35] möglich sein sollte, die Verbindungen zwischen Modulen zu ermitteln und explizit dem Werkzeug zur Verfügung zu stellen.

Dazu ist es allerdings erforderlich, daß Werte für einige Elemente des erweiterten Zustandsraumes berechnet werden müssen, zum Beispiel, wenn die `Connect`–Anweisungen in einer Schleife vorhanden sind.

3. Eine weitere Anforderung an die Spezifikation ist, daß die Bestimmung der Partner–Module bei einer Kommunikation zwischen Modulen nicht von aktuellen Werten eines Elementes des erweiterten Zustandsraumes abhängig sein darf, da es durch die abstrahierte Darstellung des Zustandes eines Systems nicht möglich ist, den aktuellen Wert zu berechnen.

   Dies bedeutet für eine Estelle–Spezifikation, daß auf Felder von Interaktionspunkten, `Output`–Anweisungen in Schleifen oder sonstigen Anweisungen innerhalb eines Transitionsblockes, die eine dynamische Bestimmung des jeweiligen Partner–Moduls erforderlich machen, verzichtet werden muß.

   Dies erscheint im ersten Moment eine erhebliche Einschränkung zu sein, es besteht aber zum Beispiel für `Output`–Anweisungen in einem `If--Then--Else`–Anweisungsblock die Möglichkeit, diese Transition in zwei Transitionen aufzuspalten: Eine für den `Then`–Teil und eine für den `Else`–Teil.
   Die Bedingung muß bei der Aufspaltung geeignet in eine `Provided`–Klausel umgewandelt werden.

   Die Umwandlung der Bedingung in eine `Provided`–Klausel ist unproblematisch, da beim Aufbau des Transitionsgraphen `Provided`–Klauseln nicht berücksichtigt werden und eine Aufspaltung der ursprünglichen Transition mit der Aufspaltung der Berechnungsfolge durch den `If--Then--Else`–Anweisungsblock übereinstimmt[36].

[34] d.h. die Verbindungsstruktur und Modulstruktur werden nur in den Initialisierungstransitionen aufgebaut.

[35] Dies wird durch den Verzicht auf dynamisches Kreieren und Terminieren von Objekten aus Abschnitt 2.4 gewährleistet.

[36] Das spezifizierte System nimmt je nach Ausführung des `Then`– oder `Else`–Teils einen anderen globalen Zustand an.

Für die anderen problematischen Anweisungen, wie Schleifen oder einem `Case`–Anweisungsblock, gibt es ähnliche Möglichkeiten.

Man beachte aber, daß die Aufspaltungen einer Transition in neue Transitionen die Anzahl der möglichen Pfade bzw. den Verzweigungsgrad im Transitionsgraphen erhöht.

4. Die Module der Estelle–Spezifikation sollten als asynchron zueinander laufenden Subsysteme attributiert werden, da zum Zeitpunkt des Aufbaus des Transitionsgraphen Vater–Sohn–Vorrangbeziehungen nicht (oder nur in Spezialfällen) beachtet werden können, so daß geschachtelte Modul–Strukturen schlechter durch den Transitionsgraphen repräsentiert werden.

   Ein einfaches Beispiel soll diese Tatsache verdeutlichen:

   > Gegeben sei ein System, in dem Vater– und Sohn–Module jeweils eine Transition mit einer `Provided`–Klausel besitzen.
   >
   > Gibt es einen globalen Zustand, in dem die beiden Transitionen feuern können, dann muß nach der Estelle–Semantik über die Vater–Sohn Vorrangbeziehung die Vater–Transition ausgewählt und gefeuert werden, während die Sohn–Transition nicht feuern darf, d.h. es darf keine Berechnungsfolge geben, in der ausgehend von diesem globalen Zustand die Sohn–Transition zu einem nächsten globalen Zustand führt.
   >
   > Für die Auswahl und das priorisierte Feuern der Vater–Transition ist es aber notwendig, den augenblicklichen globalen Zustand des Systems zu kennen.
   >
   > Dies erfordert, daß die `Provided`–Klauseln der beiden Transitionen ausgewertet werden müssen. Dazu ist es notwendig, den Wert jedes Elementes des Zustandsraumes zu verfolgen, mit der Gefahr einer Zustandsexplosion.
   >
   > Diese Gefahr wurde aber durch den Übergang zu abstrakten globalen Zuständen und der Randbedingung, keine Wertverfolgung durchzuführen, verhindert, so daß eine Auswertung der Elemente der `Provided`–Klausel nicht erfolgen kann.
   >
   > Deshalb muß beim Aufbau davon ausgegangen werden, daß auch die Sohn–Transition im entsprechenden abstrakten globalen Zustand ausgewählt und gefeuert werden kann.

Man beachte, daß aus dem gleichen Grund Prioritäten zwischen Transitionen nicht oder nur in Spezialfällen für den Aufbau herangezogen werden können.

Prinzipiell muß alles verboten werden, was eine Berechnung von Werten für Elemente des erweiterten Zustandsraumes beim Aufbau des Transitionsgraphen erforderlich machen würde.

Insgesamt schränken diese Anforderungen an den Spezifikationsstil die Möglichkeiten zur Spezifikation eines Intelligenten Netzwerkes in Estelle ein. Erste Erfahrungen ([Mi96]) zeigen jedoch, daß es weiterhin möglich ist, reale Systeme geeignet zu spezifizieren.

3.5 Zwischenform

Die Zwischenform ist eine Estelle–ähnliche, textuelle Form der Ausgangsspezifikation.

Der Übergang von der Estelle–Spezifikation zur Zwischenform stellt einen Abstraktionsschritt dar, da diese Zwischenform sich auf die für den Aufbau des Transitionsgraphen und für die Analyse benötigten Informationen aus der Ausgangsspezifikation beschränkt.

Durch das höhere Abstraktionsniveau braucht das in dieser Arbeit beschrieben Werkzeug sich nicht mit für den Aufbau des Transitionsgraphen und für die Analyse unwichtigen Teilen der Estelle–Spezifikation (wie zum Beispiel Funktionen, Prozeduren, Modul–Attributierungen, Typ–Definitionen etc.) zu beschäftigen.

Außerdem ist das Werkzeug dadurch unabhängig von anderen Werkzeugen, wie zum Beispiel der PET–Klassenbibliothek[37], so daß interne Änderungen in anderen Werkzeugen keine Auswirkungen auf dieses Werkzeug haben können.

Weiterhin können über die Zwischenform leicht Erweiterungen gegenüber Estelle eingebracht werden. Ein Beispiel ist die Einführung von synchroner Kommunikation für einzelne Interaktionspunkte. Dadurch kann das Problem der Nebenläufigkeit für die Analyse reduziert werden. Abschnitt 4.3 behandelt diese Erweiterungsmöglichkeit.

Darüber hinaus bietet die Zwischenform die Möglichkeit, einige Parameter für den Aufbau des Transitionsgraphen einzustellen. So kann zum Beispiel

[37]siehe Abschnitt 4.1.1 über die technische Umgebung des Werkzeuges

für jeden Interaktionspunkt die Länge der zugehörigen Warteschlange individuell eingestellt werden.

Die Zwischenform läßt sich ohne Probleme per Hand erstellen. Sie ist jedoch so konzipiert, daß sie maschinell durch ein Frontend[38] generiert werden kann.

Die wesentlichen Aufgaben eines solchen Frontends sind:

- Für jede Instanz eines Estelle–Moduls eine Modul–Struktur in der Zwischenform zu generieren.

  Dabei können die Modulvariablen in der Spezifikation als Namen für die Modul–Strukturen benutzt werden.

- Die Verbindungsstruktur für die Zwischenform aufzubereiten.

- Die Nachrichten und Nachrichtenparameter aus den Kanaldefinitionen der Estelle–Spezifikation herauszufinden und mit global eindeutigen Namen zu versehen.

- Die Transitionen der Estelle–Spezifikation in die entsprechenden Transitionen der Zwischenform umzusetzen. Dazu muß das Frontend bestimmen, welche Auswirkungen eine Transition auf den Hauptzustand bzw. die Zustandsvariablen haben kann und welche Nachrichten in den `Output`–Anweisungen an andere Estelle–Module verschickt werden können.

  Außerdem könnte das Frontend die Aufspaltung einer Transition mit `Output`–Anweisungen in einem `If--Then--Else`–Anweisungsblock durchführen.

  Weiterhin muß das Frontend die Output–Dependency–Pattern über die Elemente des erweiterten Zustandsraumes für die einzelnen Transitionen erstellen.

Die aufwendigste Aufgabe dürfte die Erstellung der Output–Dependency–Pattern darstellen. Dazu sind folgende Design–Entscheidungen zu treffen:

1. Wie wird mit Record–Strukturen verfahren? Wird jede Komponente des Records als eine eigenständige „Variable“ betrachtet, so daß eine feinere Datenabhängigkeitsanalyse möglich ist, oder wird ein Record immer als ganzes betrachtet?

   Man beachte, daß die Datenabhängigkeitsanalyse durch die Verfeinerung zeitaufwendiger wird.

[38] siehe Abschnitt 4.1.1

2. Können Elemente eines Feldes (`Array`) als eigenständige „Variablen" betrachtet werden?

   Im Normalfall nicht, da auf diese Felder über einen Index zugegriffen wird, dessen Wert dazu jeweils bekannt sein müßte.

3. Welche Abhängigkeiten gelten für Variablen in Schleifen, in `If--Then--Else`–Anweisungsblöcken etc.?

   Man wird am besten die maximal möglichen Abhängigkeiten bilden. Man muß aber immer die Abhängigkeiten von den Variablen der Bedingungen berücksichtigen.

4. Wie werden die zu einer Transition, zu einer Funktion etc. lokalen Variablen behandelt? Welchen Einfluß haben die lokalen Variablen auf die innerhalb eines Moduls existierenden globalen Variablen?

5. Wie ist der Aufruf einer Funktion bzw. einer Prozedur zu behandeln ?

   Man beachte hierbei auch die Art und Weise, wie Parameter übergeben werden können („call–by–referenc" bzw. „call–by–value").

6. Welche Abhängigkeiten können durch Nachrichtenparameter entstehen bzw. wie können Nachrichtenparameter beeinflußt werden ?

   Man beachte, daß Nachrichten an beliebig vielen Stellen in einer Anweisungsfolge eines Transitionsblockes mittels `Output`–Anweisungen verschickt werden können. Dann können die Parameter der Nachricht für jedes Auftreten der `Output`–Anweisungen unterschiedlich von anderen Elementen des erweiterten Zustandsraumes abhängig sein.

7. Was gilt für Nachrichten die keine Parameter haben ?

   Ihre Existenz kann zum Beispiel vom Wert der in einer `If`–Bedingung referenzierten Variablen abhängig sein.

8. Auf welche Weise müssen die Variablen in den Klauseln einer Transition berücksichtigt werden ?

   Von diesen Variablen hängen alle anderen in dem Transitionsblock benutzten Variablen ab.

9. Wie ist die Abhängigkeit der Variablen von Modul–Parametern ?

Die Liste läßt sich sicher noch erweitern.

Prinzipiell stellt sich immer die Frage, wie ein Element des erweiterten Zustandsraumes von einem anderen abhängt, wenn man die Anweisungsfolge in einem Transitionsblock Schritt für Schritt abarbeitet. Dabei müssen noch die in den Transitionsklauseln referenzierten Elemente des erweiterten Zustandsraumes berücksichtigt werden.

Kapitel 4

Details der Implementation

Dieses Kapitel beschreibt einige Details der Implementation. Der erste Abschnitt 4.1 des Kapitels beschäftigt sich mit dem Einsatz des Werkzeuges. In diesem Abnschnitt werden das technische Umfeld, der Aufruf des Werkzeuges und die Ausgaben des Werkzeuges vorgestellt. Im zweiten Abschnitt 4.2 steht die interne Struktur der Implementation im Vordergrund und im dritten Abschnitt 4.3 werden Erweiterungsmöglichkeiten des Werkzeuges besprochen.

4.1 Einsatz des Werkzeuges

Das Werkzeug dient, wie in den vorangegangenen Kapiteln dargestellt, zur Erkennung von Feature–Interaktionen in Intelligenten Netzwerken. In den folgenden Unterabschnitten wird der Einsatz des Werkzeuges näher erläutert.

4.1.1 Technische Umgebung

Wie aus der Abbildung 4.1 zu erkennen ist, baut das hier entwickelte Werkzeug auf anderen Werkzeugen auf.

Ausgehend von einer Estelle–Spezifikation mit Präprozessor–Anweisungen („<spec>.epp“) werden über einen Estelle-Präprozessor ([BrGo94a], [BrGo94b]) die Transitionen der einzelnen Features textuell ein- oder ausgeschlossen.

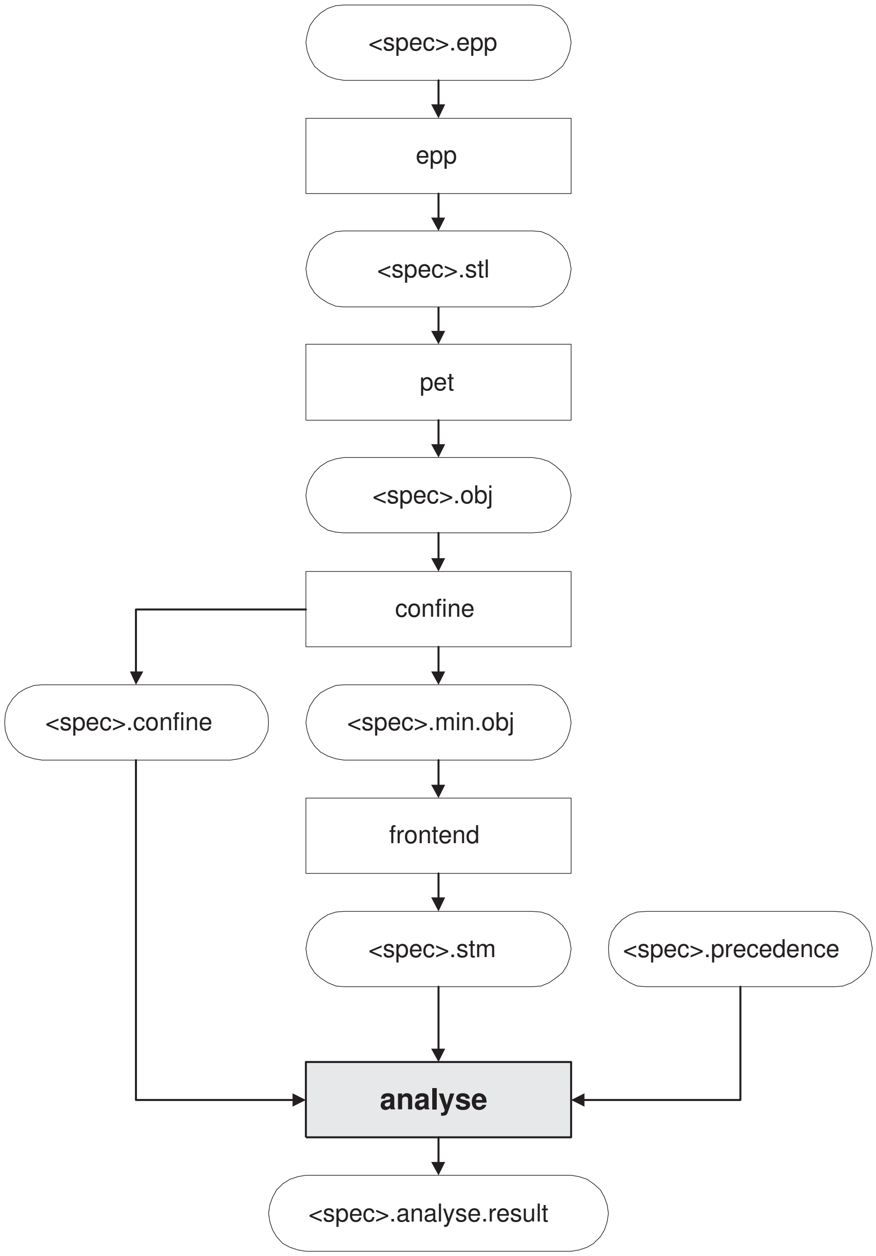

Abbildung 4.1: Technische Umgebung

Die resultierende Estelle–Spezifikation wird mittels des Compiler–Frontends **PET**[1] aus dem PET-DINGO-Toolkit ([SiSt91b], [SiSt91a]) in ein Objektformat („<spec>.obj“) übersetzt.

Danach erfolgt mittels des Werkzeuges CONFINE[2] eine Vorverarbeitung und erste Analyse der Spezifikation.

Das resultierende Objektformat wird nach der Verarbeitung durch CONFINE manuell oder mit Hilfe eines entsprechenden (noch zu entwickelnden) Werkzeuges in das in Abschnitt 3.5 erläuterte Zwischenformat („<spec>.stm“) übersetzt.

Basierend auf diesem Zwischenformat, den Ergebnissen von CONFINE und den Präzedenzen, wird der Transitionsgraph aufgebaut, dieser mittels der Ergebnisse von CONFINE und den Präzedenzen anhand der FI–Kriterien für Feature–Interaktionen untersucht[3] und das Ergebnis der Analyse in die Ausgabedatei („<spec>.analyse.result“) geschrieben.

4.1.2 Aufruf des Werkzeuges

Ohne Angabe eines Argumentes liefert das Werkzeug eine kurze Hilfestellung über seine Benutzung und die möglichen Optionen.

Ansonsten wird das Werkzeug auf der Kommandozeile mit drei Dateien als Argumente aufgerufen: Die zu analysierenden Zwischenform („<spec>.stm“), die Präzedenzmatrix in Form von Listen von Features („<spec>.precedence“) und die durch eine Analyse mit CONFINE gewonnenen Ergebnisse („<spec>.confine“).

In der folgenden Aufstellung werden die möglichen Optionen und ihre Wirkung kurz beschrieben. Das Werkzeug benutzt das für die jeweiligen Optionen angegebene Default–Verhalten, wenn eine Option nicht explizit angegeben wird.

Die folgenden Optionen dienen als Schalter. Mit „`-OPTION[+|-]`“ wird die entsprechende Option „eingeschaltet“ (+) bzw. „ausgeschaltet“(–).

- **-egraph**
  Ausgabe des Erreichbarkeitsgraphen
  Mit dieser Option wird der Erreichbarkeitsgraph(d.h. alle abstrakten

[1] **P**ortable **E**stelle **T**ranslator
[2] siehe Abschnitt 2.6
[3] siehe Kapitel 2 und 3

globale Zustände und die in einem Zustand als nächstes zu schaltenden Transitionen) ausgegeben.

–: Default, nicht ausgeben.

+: Ausgabe in „<spec>.egraph.result“

- **-tgraph**
  Ausgabe des Transitionsgraphen
  Damit werden alle Knoten des Transitionsgraphen zusammen mit ihren Kinder–Knoten ausgegeben.

  –: Default, keine Ausgabe

  +: Ausgabe in „<spec>.tgraph.result“

- **-msg**
  Berücksichtigung von Nachrichten beim Aufbau des Transitionsgraphen

  –: Beim Aufbau wird der Austausch von Nachrichten zwischen Estelle–Modulen nicht berücksichtigt.

  +: Default, Nachrichten werden beim Aufbau berücksichtigt.

- **-reach**
  Ausgabe der Matrix ***ReachableMatrix***[4]
  Damit werden die von einem Knoten im Graphen aus erreichbaren Transitionen ausgegeben.

  –: Default, keine Ausgabe der Matrix

  +: Ausgabe der Matrix in „<spec>.reachable.result“

Mit den folgenden Optionen kann mit „`-OPTION n`“ eine Größe auf den angegebenen Wert n gesetzt werden.

- **-clw** n
  Grenzwert für Ausgabe einer Warnung
  Überschreitet die Anzahl der Elemente in der Liste `CollectList`[5] diesen Grenzwert(d.h. es müssen mehr als n Tiefensuchen im Transitionsgraphen ausgeführt werden) dann gibt das Werkzeug eine Warnung aus.

[4]siehe Abschnitt 3.2.5 über die zweite Analysephase.
[5]siehe Abschnitt 3.2.5 über die zweite Analysephase

n: Grenzwert

Default: Als Default wird die Hälfte der Anzahl von Knoten im Transitionsgraphen benutzt.

- **-max_trans_node** n
  Maximale Anzahl von Knoten im Transitionsgraphen[6]

  n: neuer Wert

  Default: $n = 100.000$

- **-max_path_len** n
  Maximale Länge eines einfachen Pfades[6]

  n: neuer Wert

  Default: $n = 100.000$

- **-max_strans_path** n
  Maximale Anzahl von hintereinander ausgewählten spontanen Transitionen[6]

  n: neuer Wert

  Default: $n = 2$

- **-max_dtrans_path** n
  Maximale Anzahl von hintereinander ausgewählten Transitionen mit ***Delay****–Klauseln*[6]

  n: neuer Wert

  Default: $n = 2$

- **-queue_len** n
  Länge der Warteschlangen für Interaktionspunkte mit dem ***Individual****–Attribut*[6]
  Wird von den Angaben für einzelne Interaktionspunkte in der Zwischenform überschrieben.

  n: neuer Wert

  Default: $n = 1$

- **-common_queue_len** n
  Länge der Warteschlangen für Interaktionspunkte mit dem ***Common****–Attribut*[6]

Wird von den Angaben für die entsprechenden Interaktionspunkte in der Zwischenform überschrieben.

n: neuer Wert

Default: $n = 1$

Es sei darauf hingewiesen, daß die optionalen Ausgabedateien „<spec>.egraph.result“ und „<spec>.tgraph.result“ je nach Umfang und Komplexität der Ausgangsspezifikation sehr groß[7] werden können und deshalb die Optionen „**-egraph**“ und „**-tgraph**“ nur mit Vorsicht zu benutzen sind.

4.1.3 Ausgaben des Werkzeuges

Das Werkzeug gibt während eines Laufes auf den Standard–Ausgabestrom („stdout“) folgende Informationen aus:

- Den aktuellen Arbeitsabschnitt, in dem sich das Werkzeug befindet,
- einen rotierenden Balken zur Anzeige, daß das Werkzeug noch am „Leben“ ist, und
- am Ende eines Arbeitsabschnittes, welche Zeit für diesen benötigt worden ist.

Weiterhin werden einige statistische Informationen angezeigt. So wird zum Beispiel nach dem Aufbau des Transitionsgraphen ausgegeben, wieviele Knoten erzeugt worden sind.

Die einzelnen Arbeitsschritte sind:

1. Einlesen des Zwischenformats, der Ergebnisse von CONFINE und der Präzedenzmatrix.
2. Aufbau des Transitionsgraphen
3. Löschen der Attribute der abstrakten globalen Zustände, d.h. alle Informationen über die Hauptzustände, Zustandsvariablen etc.

[6] siehe Abschnitt 3.1.2

[7] mehrere Megabyte

4. Vorbereitungsphase
5. Paar–Analysephase
6. Pfad–Analysephase, unterteilt in
   - Tiefensuchen im Transitionsgraphen
   - Bearbeitung der Listen
7. Ausgabe des Ergebnisses in die Datei „<spec>.analyse.result“

Die Ausgabedatei besteht aus einer Folge von FI-Beschreibungsgruppen. Jede dieser Gruppen besteht aus mehreren Zeilen nach folgendem Schema:

```
<Modul> <Transition> <Feature>
      <Modul> <Transition> <Feature> <Flags>
         ⋮          ⋮           ⋮          ⋮
      <Modul> <Transition> <Feature> <Flags>
```

Die 1. (nicht eingerückte) Zeile jeder Gruppe beschreibt eine Transition, die Feature–Interaktionen mit den in den folgenden (eingerückten) Zeilen angegebenen Transitionen hat. Die Flags geben dabei genauer an, um welche Art von Feature–Interaktion es sich jeweils handelt.

Folgende Flags können auftreten:

B1: Es handelt sich um eine Feature–Interaktion nach Basiskriterium 1 zwischen der ersten Transition und der Transition aus der Liste von Transitionen.

B2: Es besteht zwischen den beiden Transitionen eine Feature–Interaktion nach Basiskriterium 2.

B3: Die beiden Transitionen erfüllen das Basiskriterium 3.

IN: Die beiden Transitionen erfüllen das Independent–FI–Kriterium.

MS: Es gibt eine Situation im Transitionsgraphen, so daß die beiden Transitionen *sicher* das Minor–FI–Kriterium erfüllen.

MP: Die erste Transition erfüllt die Bedingungen des Major–State–Preservings des Minor–FI–Kriteriums, aber es besteht eine *prinzipiell mögliche*[8] Datenabhängigkeit zwischen dieser und der Transition aus Liste.

PE: Die Einführung einer Präzedenz zwischen den beiden Transitionen führt zu neuen Feature–Interaktionen[9].

PA: Es besteht zwischen den beiden Transitionen eine Feature–Interaktion *aufgrund* der Einführung einer Präzedenz[10].

PZ: Zwischen den beiden Transitionen besteht eine Feature–Interaktion *im Zusammenhang* mit der Einführung einer Präzedenz[10].

Man beachte, daß für eine Transition in der Liste mehrere Flags gesetzt sein können. Dies entsteht einmal durch die Überschneidungen zwischen den einzelnen FI–Kriterien und zum anderen dadurch, daß durch den Übergang von den Knoten im Graphen zu den Transitionen in der Ausgabedatei die Informationen über die Situation im Transitionsgraphen verloren gehen. Dafür ist der Umfang des Ausgabeprotokolls geringer.

Weiterhin gibt das Werkzeug einige statistische Informationen

- wie Anzahl der Knoten im Transitionsgraphen,
- durchschnittlicher Verzweigungsgrad,
- wieviele Knoten in einen bereits existierenden abstrakten globalen Zustand oder in einen Null–Zustand führen und
- an wievielen Knoten der Aufbau des Transitionsgraphen nicht fortgesetzt wurde, weil die Warteschlangen voll sind oder die maximale Pfadlänge erreicht wurde, etc.

in die Datei „<spec>.info.result“ aus.

Außerdem gibt das Werkzeug mit der Option „-reach+“ in die Datei „<spec>.reachable.result“ jede Zeile bzw. jeden Knoten aus der Matrix `ReachableMatrix` zusammen mit den von diesem Knoten aus erreichbaren Transitionen aus.

[8] siehe Abschnitt 3.2.5

[9] siehe Abschnitt 2.5 und Abschnitt 3.2.1.

[10] siehe Abschnitt 3.2.1 und Abschnitt 3.2.5.

4.2 Interne Struktur

Dieser Abschnitt beschreibt die interne Struktur der Implementation soweit, daß der Leser einen Überblick über die einzelnen Teile des Quelltext erhält und gegebenenfalls Erweiterungen in das Werkzeug einbringen kann.

Der gesamte Quelltext ist in C++ geschrieben und ist objektorientiert strukturiert. Dies sollte es dem Leser erleichtern, sich im Quelltext zurechtzufinden.

Zur weiteren Erleichterung der Lesbarkeit des Quelltextes sind in Abschnitt 4.2.1 einige allgemeine Konventionen aufgeführt, die während der Implementation berücksichtigt wurden.

Die darauf folgenden Abschnitte beschäftigen sich schließlich näher mit den wesentlichen Komponenten des Werkzeuges.

Insgesamt ist das Werkzeug in die folgenden Module aufgeteilt:

1. **RMain**
   Hauptmodul des Werkzeuges, ist verantwortlich für die Steuerung des Werkzeuges.

2. **RAnalyse**
   Beinhaltet die Suchroutinen und Algorithmen für die Erkennung von Feature–Interaktionen.

3. **RGraph**
   Ist für den Aufbau des Graphen, die Ausgabe des Erreichbarkeitsgraphen und des Transitionsgraphen verantwortlich.

4. **RConfine**
   Das Modul ist für das Einlesen der Ergebnisse von CONFINE[11] verantwortlich.

5. **RPrecedence**
   Dieses Modul liest die Präzedenzmatrix[12] ein.

6. **RData**
   Aufgabe dieses Moduls ist die Erfassung und Bereitstellung der Zwischenform in einer für den Aufbau und die Analyse des Transitionsgraphen geeigneten Form.

[11] siehe Abschnitt 2.6
[12] siehe Abschnitt 2.5

7. **RMisc**
   Hier liegen Hilfsklassen für das System, wie zum Beispiel zur formatierten Ausgabe in eine Datei.

8. **XTools**
   Dieses Modul enthält die notwendigen Klassen für Listen, Tabellen, Stacks, Matrizen etc., meistens in Form von Template–Definitionen. Dabei ist für die Entwicklung einiger der Template–Klassen die Klassen–Bibliothek von CONFINE ([Th95]) zugrundegelegt worden.

Daraus ergeben sich die in Abbildung 4.2 gezeigte Hierarchie und (zusammen mit dem Makefile für den GNU–Compiler „g++ V 2.7.2“) folgende Dateien:

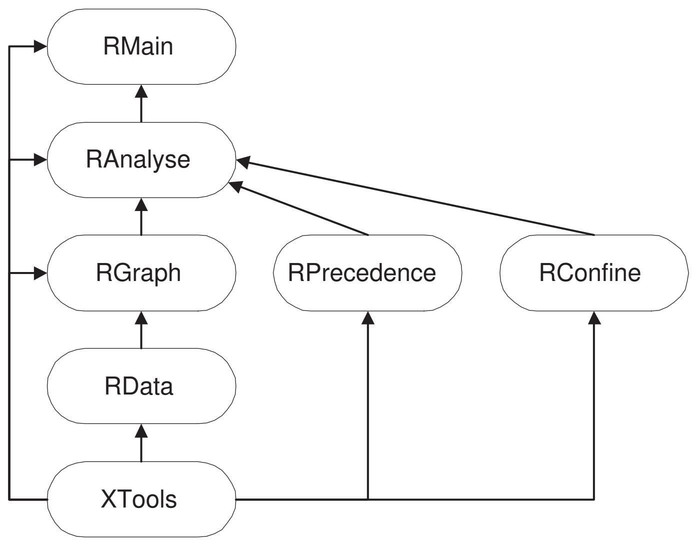

Abbildung 4.2: Modulhierachie

makefile	
RMain.hxx	RMain.cxx
RAnalyse.hxx	RAnalyse.cxx
RGraph.hxx	RGraph.cxx
RConfine.hxx	RConfine.cxx
RPrecedence.hxx	RPrecedence.cxx
RData.hxx	RData.cxx
RMisc.hxx	RMisc.cxx
XTools.hxx	

4.2.1 Allgemeine Konventionen

In diesem Abschnitt werden die allgemeinen Konventionen aufgeführt, die beim Schreiben des Werkzeuges berücksichtigt wurden. Diese Stilregeln dienen vor allem dazu, einen besseren Überblick über das Programm und die Arbeitsweise des Programmes zu erhalten.

Eine der benutzten Stilregeln bezieht sich auf die Namensgebung von Variablen. Für Variablen gilt die einfache Konvention, daß sie entsprechend ihres C++–Basistyps mit einem der folgenden Präfixe versehen werden:

b	bool	Boolean–Variablen (äquivalent zu int)
n	int	vorzeichenbehaftete Ganzzahlen
u	unsigned	vorzeichenlose Ganzzahlen
c	char	Zeichen
uc	unsigned char	Byte
sz	char[]	nullterminierte Strings

Zeiger auf Variablen und Strukturen werden zusätzlich mit einem „p“ gekennzeichnet, so daß zum Beispiel ein Zeiger auf einen Integer folgende Gestalt hat: „int* pnStuff“

Eine weitere Stilregel ist die Erweiterung der Variablennamen um die Art der zugrundeliegenden Strukturen. Zum Beispiel wird eine Tabelle von Transitionen entsprechend als „TransTable“, eine Liste von Transitionen als „TransList“ benannt.

Weiterhin werden alle Klassen durch ein „R“ als Präfix vor dem eigentlichen Klassennamen gekennzeichnet. So ist zum Beispiel „RTrans“ eine Klasse, die für die interne Repräsentation einer Transition verantwortlich ist.

Eine Ausnahme von dieser Regel stellen die Templateklassen und einige Hilfsklassen des Moduls „XTools“ dar. Diese Klassen werden mit einem „X“ vor dem eigentlichen Klassennamen gekennzeichnet, da es sich um allgemeine Klassen für Listen, Tabellen etc. handelt, die als Grundlage für größere Strukturen dienen.

Die Idee bei diesen Konventionen ist es, Typ und Herkunft einer Variablen bereits am Namen erkennen zu können und Namenskonflikte bei späteren Erweiterungen vorab bereits zu vermeiden.

Dadurch und durch die systematische Kommentierung des Quelltextes sollte zusammen mit den Beschreibungen aus Kapitel 3 eine Einarbeitung in die interne Arbeitsweise wesentlich erleichtert werden.

4.2.2 Erfassung und Aufbereitung der Zwischenform

Das Modul „RData“ dient zur Erfassung der Zwischenform.

Für das Einlesen der Zwischenform ist die Klasse *RParse* verantwortlich. Diese Klasse enthält für jedes Schlüsselwort der Zwischenform eine Parsing–Funktion, die direkt die entsprechenden Klassen, wie zum Beispiel die Klasse *RModule*, initialisiert.

Für die Unterstützung der Parsing–Funktionen enthält sie zusätzlich einen Zeiger auf das gerade eingelesene Wort („pCurrToken“) und den Typ des Wortes („nCurrType“). Dieser Typ beschreibt, ob das Wort ein Schlüsselwort oder ein Name ist.

Das Prinzip des Parsers ist, daß der Zeiger „pCurrToken“ auf das jeweils nächste zu bearbeitende Wort zeigt. Dazu wird nach Aufruf einer Parsing–Funktion als erstes ein neues Wort gelesen und vor dem Verlassen der Funktion der Zeiger „pCurrToken“ auf das nächste zu bearbeitende Wort gesetzt.

Die Klasse *RParse* implementiert einen einfachen Parser, der im wesentlichen das Initialisieren der entsprechenden Klassen für die Repräsentation der einzelnen Konstrukte der Zwischenform übernimmt. Auf das Überprüfen der semantischen Korrektheit der Zwischenform ist verzichtet worden, weil davon ausgegangen wurde, daß diese Überprüfung von dem Werkzeug übernommen wird, das für die Erstellung der korrekten Zwischenform aus einer semantisch korrekten Estelle–Spezifikation zuständig ist.

Der größere Teil der in diesem Modul vorhandenen Klassen bezieht sich auf die interne Repräsentation der einzelnen Konstrukte der Zwischenform.

Das Prinzip bzw. der Aufbau dieser Klassen ist relativ einfach.

Als Beispiel für diesen Aufbau sei kurz die Klasse *RModule* skizziert, welche ein Objekt vom Typ Estelle–Modul repräsentiert. Da ein Objekt dieses Typs als untergeordnete Elemente eine Menge von Interaktionspunkten, Zuständen, Transitionen etc. besitzen kann, enthält die entsprechende Klasse *RModule* eine Tabelle von Interaktionspunkten (Klasse *RIp*), eine Tabelle von Zuständen (Klasse *RState*) usw. Der Index der jeweiligen Tabellen wird entsprechend zur Identifikation der Elemente benutzt. Zum Beispiel wird eine Transition eines Moduls durch den Index der Modul–Tabelle und den Index der Transitionstabelle des jeweiligen Moduls identifiziert. Dies entspricht einer natürlichen Abbildung der Objekte auf ihre zugeordneten Klassen und wird durch die Paradigmen wie Kapselung in objektorientierten Sprachen unterstützt.

Für den Aufbau des Transitionsgraphen ist diese Darstellung vorteilhaft, da alle Informationen für ein Estelle–Modul bzw. für eine Transition sofort greifbar sind. Dies ist zum Beispiel ein Vorteil beim Bestimmen der als nächstes zu schaltenden Transitionen beim Aufbau des Transitionsgraphen. Ein weiterer Vorteil ist die einfache Erweiterbarkeit der Zwischenform um neue Konstrukte.

Für die Matrizenoperationen während der Analyse ist diese Darstellung schlechter geeignet. Deshalb werden alle in der Zwischenform angegebenen Transitionen in eine Tabelle vom Typ `RTransDict`[13] zusammengefaßt (ihre Modul–Identifikation und ihre Transitions–Identifikation innerhalb eines Moduls), so daß der Index in dieser Tabelle als Kennung (Transitions–ID) für eine Transition benutzt werden kann.

Die Variablen, Nachrichten und Nachrichtenparameter werden aus dem gleichen Grund wie die Transitionen in eine Tabelle vom Typ `RVarDict` zusammengefaßt. Dadurch ist es möglich, die Output–Dependency–Pattern als Matrizen darzustellen, so daß die Datenabhängigkeitsanalyse bzw. der Aufbau des Abhängigkeitsgraphen aus Abschnitt 3.2.5 für die Anwendung des Minor–FI–Kriteriums effizient erfolgen kann.

Zusammengefaßt werden diese Klassen in einer Klasse `RData`. Diese enthält die notwendigen Zugriffsfunktionen auf die entsprechenden Tabellen und sonstigen benötigten Informationen, so daß zum Beispiel für den Aufbau des Transitionsgraphen lediglich eine Instanz der Klasse `RData` übergeben werden muß.

4.2.3 Einlesen der Ergebnisse von CONFINE

Die Aufgabe des Moduls „RConfine“ ist das Einlesen der Ergebnisse einer Analyse der Spezifikation durch das Werkzeug CONFINE[14].

Dazu muß CONFINE mit der Estelle–Spezifikation und mindestens mit den Optionen „-short+ -indet+ -over+“ aufgerufen werden.

Das Ergebnis der Analyse durch CONFINE ist eine Ausgabe des Analyseprotokolls auf den Standard–Ausgabestrom. Dieses Ergebnis muß in eine Datei umgeleitet werden, so daß sie dem hier beschriebenen Werkzeug als Argument zur Verfügung gestellt werden kann.

[13] Der Teil `Dict` stammt von „Dictionary“

[14] siehe Abschnitt 2.6

Das Modul „RConfine“ liest diese Datei ein und baut basierend auf der Tabelle aus dem Modul „RData“ vom Typ `RTransDict` eine Matrix vom Typ `RIndMatrix` über die Transitionen der Spezifikation auf. In der Matrix wird festgehalten, welche Transitionen nach CONFINE miteinander einen Indeterminismus zeigen.

4.2.4 Einlesen der Präzedenzmatrix

Das Modul „RPrecedence“ ist verantwortlich für das Einlesen der Präzedenzmatrix. Es erwartet die Präzedenzen als Paare (F, G) von Feature–Namen. Dabei muß gelten, daß $F \succ G$ gilt, d.h. F hat eine höhere Präzedenz gegenüber dem Feature G.

Zusammen mit den Feature–Namen aus der Zwischenform baut dieses Modul intern dann eine Matrix vom Typ `RPrecMatrix` auf, so daß über die Feature–Namen darauf zugegriffen werden kann.

4.2.5 Aufbau des Graphen

Das Modul „RGraph“ stellt die Klassen für den Aufbau des Transitionsgraphen zur Verfügung. Die wesentlichen Klassen des Moduls sind:

- `RGraph` als Repräsentation und zur Steuerung des Aufbaus des gesamten Graphen.
- `RTransNode` für die Knoten und Kanten des Graphen.
- `RGlobalState` als Klasse zur Darstellung der abstrakten globalen Zustände.
- `RLocalState` für die lokalen Komponenten eines abstrakten globalen Zustandes, d.h. zur Repräsentation eines Estelle–Moduls.
- `RStateVar` für die Darstellung einer Zustandsvariablen.
- `RQueue` als Repräsentation der Warteschlangen für die Interaktionspunkte eines Estelle–Moduls.

Die beiden wesentlichen Member–Funktionen der Klasse `RGraph` sind die Funktionen `build()` und `decide()`. Die Funktion `build()` ist für den Aufbau des Graphen verantwortlich und verwendet die Funktion `decide()`. Diese entscheidet darüber, ob die Transition in einen Null–Zustand oder einen bereits existierenden abstrakten globalen Zustand etc. führt.

Das Prinzip des Aufbaus ist in Abschnitt 3.1.2 näher erläutert.

Beide Funktionen benutzen Member–Funktionen der anderen Klassen, um die Auswirkungen des Schaltens einer Transition auf einen abstrakten globalen Zustand und die in einem abstrakten globalen Zustand schaltbar gewordenen Transitionen eines Estelle–Moduls zu bestimmen.

Weiterhin setzen die beiden Funktionen Markierungen bzw. Flags in einem Knoten. Die Bedeutungen der Flags sind:

M: Mit dem Schalten der zugehörigen Transition ist die maximal zulässige Anzahl von Knoten im Graphen erreicht bzw. überschritten worden.

P: Mit dem Schalten der Transition ist die maximal erlaubte Pfadlänge für einen einfachen Pfad erreicht worden.

Q: Die zugehörige Transition kann nicht geschaltet werden, da die Warteschlangen der Empfangsmodule die Nachrichten nicht aufnehmen können.

S: Die spontane Transition ist bereits mehrmals hintereinander auf einem einfachen Pfad ausgeführt worden.

D: Die Transition mit `Delay`–Klausel ist bereits mehrmals hintereinander auf einem einfachen Pfad ausgeführt worden.

E: Mit dem Schalten der Transition ist ein im Transitionsgraphen bereits existierender abstrakter globaler Zustand erreicht worden.

N: Das Schalten der Transition führte in einen Null–Zustand.

Diese Markierungen werden bei der Ausgabe des Erreichbarkeits– bzw. des Transitionsgraphen mit ausgegeben.

4.2.6 Analyse des Graphen

Das Modul „RAnalyse“ enthält die Klassen für die eigentliche Erkennung und Einstufung von Feature–Interaktionen. Siehe dazu den Abschnitt 3.2 über das Analyseverfahren.

Die wesentlichen Klassen sind:

- **RDeepSearchBase**
  Die Basisklasse für die in Abschnitt 3.2.2 beschriebene Tiefensuche.

- **RIndependentMatrix**
  Für die Matrix `IndependentMatrix`, d.h. welche Transitionen die Independent–Bedingung erfüllen.

- **RReProzessLists**
  Diese Klasse enthält die einzelnen Listen von Knoten bzw. Paaren von Knoten für die Weiterverarbeitung in der zweiten Analysephase.

- **RResultMatrix**
  Für die Speicherung der Ergebnisse der Analyse.

- **RPairAnalyse**
  Für die erste Analysephase. Die Klasse ist von der Klasse **RDeepSearchBase** abgeleitet.

- **RReachableMatrix**
  Als Klasse für die Matrix über die Knoten und Transitionen, d.h. welche Transitionen sind von einem Knoten aus erreichbar. Sie bekommt die Liste `CollectList` übergeben, führt für jeden der Knoten in der Liste eine Tiefensuche durch, und speichert das Ergebnis in der Matrix. Die Klasse ist von der Klasse `RDeepSearchBase` abgeleitet.

- **RDependencyGraph**
  Als Repräsentation des Abhängigkeitsgraphen für die Datenabhängigkeitsanalyse und zur Entscheidung der prinzipiellen Datenabhängigkeit zwischen Transitionen.

- **RPathAnalyse**
  Für die zweite Analysephase. Sie bearbeitet die Listen der Klasse `RReProzessLists`.

- **RAnalyse**
  Diese Klasse steuert den Ablauf der Analyse, d.h. zuerst Vorbereitungsphase, dann die Paar–Analyse und schließlich die Pfad–Analyse, sammelt die Ergebnisse ein und besitzt Funktionen zur Ausgabe des Ergebnisses.

Weiterhin enthält das Modul Klassen für die Bildung der ansonsten benötigten Matrizen und Felder.

Alle diese Klassen besitzen entsprechend ihrer Aufgabenstellung Zugriffsfunktionen und Verarbeitungsfunktionen.

Das Prinzip des Ablaufes der Analyse und das Zusammenspiel der einzelnen Klassen sollte mit Abschnitt 3.2 über das Suchverfahren leicht aus dem Quelltext zu ersehen sein.

4.3 Mögliche Erweiterungen

In diesem Abschnitt werden Erweiterungsmöglichkeiten zur Verringerung der Nebenläufigkeit bzw. zur Verringerung der Anzahl von Knoten im Transitionsgraphen kurz vorgestellt.

Die *erste Erweiterungsmöglichkeit* betrifft die Einführung von synchroner Kommunikation für bestimmte Verbindungen zwischen Estelle–Modulen.

Die Idee bei dieser Erweiterung ist, daß nach dem Absenden einer Nachricht in den empfangenden Estelle–Modulen nur diejenigen Transitionen ausgewählt werden dürfen, die am entsprechenden Interaktionspunkt auf den Eingang der Nachricht warten, und daß im sendenden Estelle–Modul solange keine Transitionen schaltbar werden bis die Nachricht angenommen worden ist.

Damit können spontane Transitionen und Transitionen mit `Delay`–Klauseln im Sender–Modul und in den Empfangs–Modulen nicht als nächstes zu schaltende Transitionen bestimmt werden. Dies hat zur Folge, daß der Verzweigungsgrad und die Anzahl von Knoten im Transitionsgraphen reduziert wird[15].

Nachteilig an dieser Erweiterung ist, daß sie eine Estelle–Erweiterung beinhaltet und entsprechend eine Anpassung der Semantik von Estelle notwendig ist. Außerdem müßte stärker die „Deadlock"–Gefahr beachtet werden.

Für die Implementation der Erweiterung müßten die Module für die Erfassung der Zwischenform „RData" und für den Aufbau des Graphen „RGraph" geändert werden. Die Änderungen an dem Modul „RData" betreffen im Wesentlichen die Änderung des Parsers, da ein neues Schlüsselwort zu verarbeiten wäre, und die Repräsentation der Verbindungen. Im Modul „RGraph" wären vor allem Änderungen bei der Bestimmung der als nächstes schaltbaren Transitionen durchzuführen und die Erkennung, daß eine synchrone Kommunikation vorliegt, zu implementieren.

Eine Änderung des Analyseverfahrens selbst wäre nicht notwendig, d.h. das Modul „RAnalyse" bliebe unverändert.

[15]siehe Abschnitt 3.3

Die *zweite Erweiterungsmöglichkeit* beschäftigt sich mit Symmetrieeigenschaften.

In einem spezifizierten Telefonsystem für zwei Benutzer können diese beiden Benutzer die gleichen Möglichkeiten haben, ein Telefongespräch zu initiieren und durchzuführen, d.h. beide können zum Beispiel den Telefonhörer abheben, beide können eine Telefonnummer wählen etc.

Für den Transitionsgraphen dieses Sytems bedeutet dies, daß zwei Teilgraphen für die Aufnahme und Durchführung eines Gespräches existieren, die sich nur dadurch unterscheiden, daß die Aufnahme eines Gespräches von dem jeweils anderen Benutzer durchgeführt wird.

Die Idee der Erweiterung wäre nun, daß beim Aufbau des Transitionsgraphen symmetrische abstrakte globale Zustände aufeinander abgebildet würden. Für ein symmetrisch spezifiziertes Telefonsystem mit zwei Benutzern erhielte man dadurch eine Halbierung der Anzahl von Knoten im Transitionsgraphen.

Dies könnte automatisch erreicht werden. Dazu müßte man für das Wiedererkennen von bereits existierenden abstrakten globalen Zuständen[16] die Eigenschaft, daß zwei abstrakte globale Zustände symmetrisch sind, verwenden. Damit würden die abstrakten globalen Zustände aufeinander abgebildet. Die notwendigen Änderungen müßten im Module „RGraph“ vollzogen werden.

Das Problem bei dieser Erweiterung ist, zu definieren, wann zwei abstrakte globale Zustände symmetrisch sind bzw. welche Symmetrieeigenschaft das System erfüllt.

Kein Problem gibt es, wenn das System eine globale Symmetrieachse besitzt, d.h. der globale Automat des Systems ist symmetrisch.

Die in [Mi96] entwickelte Spezifikation eines Intelligenten Netzwerkes hat möglicherweise eine globale Symmetrieachse, da diese Spezifikation eine symmetriche Verbindungsstruktur aufweist, und weil die nach der Instanziierung der Estelle–Module entstehende Struktur symmetrisch ist. Es bleibt die Frage, ob diese beiden Eigenschaften hinreichende Bedingungen für das Vorhandensein einer globalen Symmetrieachse darstellen.

---

[16] siehe Abschnitt 3.1.2

Kapitel 5

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Werkzeug zur Unterstützung einer Analyse auf Feature–Interaktionen eines in Estelle spezifizierten Intelligenten Netzwerkes beschrieben. Das Werkzeug untersucht dabei anhand der in [Bre95] entwickelten FI–Kriterien (siehe auch Abschnitt 2.2) die Estelle–Spezifikation und macht Aussagen über die „Gefährlichkeit“ der Feature–Interaktionen mittels der FI–Kriterien. Weiterhin analysiert das Werkzeug eine Spezifikation daraufhin, ob die Einführung einer Präzedenz zur Auflösung einer Feature–Interaktion neue Interaktionen erzeugt (siehe Abschnitt 2.5).

Dazu führt das Werkzeug eine Erreichbarkeitsanalyse auf einem zum Erreichbarkeitsgraphen äquivalenten Transitionsgraphen (siehe Abschnitt 3.1) durch. In diesem Graphen stehen die Transitionen im Vordergrund, während im Erreichbarkeitsgraphen dies die erreichbaren Zustände eines Systemes sind. Der Transitionsgraph ist aus der Beobachtung enstanden, daß für die Anwendung der FI–Kriterien der erreichte globale Zustand letztlich keine Rolle spielt.

Das Problem einer Zustandsexplosion bei einer Erreichbarkeitsanalyse wird durch den Übergang von globalen zu abstrakten globalen Zustände verhindert (siehe Kapitel 2).

Allerdings hat sich gezeigt, daß ein anderer Faktor die Erreichbarkeitsanalyse auf einem Erreichbarkeitsgraphen bzw. einem Transitionsgraphen wesentlich beeinflußt. Dieser Faktor ist die Nebenläufigkeit der Estelle–Module eines verteilten Systems wie einem Intelligenten Netzwerk. Im besonderen sind in diesem Zusammenhang die Auswirkungen von spontanen Transitionen und Transitionen mit `Delay`–Klauseln zu nennen (siehe Abschnitt 3.3).

Mögliche Erweiterungen des Werkzeuges zur Reduzierung der Nebenläufigkeit sind in Abschnitt 4.3 aufgeführt. Dies sind die Einführung einer synchronen Kommunikation für einige Verbindungen zwischen Estelle–Modulen und die Idee, Symmetrieeigenschaften einer Spezifikation auszunutzen.

Im Zusammenhang mit der Nebenläufigkeit ist auch die Erweiterung der abstrakten globalen Zustände durch Hinzunahme von Nachrichten bzw. Warteschlangen für die Interaktionspunkte zu nennen. In der Regel dienen diese Nachrichten zur Synchronisation der Estelle–Module (siehe Abschnitt 3.1.1).

Neben der Erreichbarkeitsanalyse muß für die Anwendung der FI–Kriterien eine Datenabhängigkeitsanalyse auf dem Transitionsgraphen durchgeführt werden. Obwohl bei dieser Analyse keine Werte für Variablen, Nachrichtenparameter etc. zu berechnen sind, hat sich herausgestellt, daß die Entscheidung über Datenabhängigkeiten zwischen zwei Transitionen nur unter Einschränkungen durchzuführen ist. Ohne diese Einschränkungen wäre eine Analyse aufgrund der Laufzeitkomplexität nicht durchzuführen. Abschnitt 3.2.5 behandelt unter anderem diese Problematik.

Insgesamt läßt sich zu den FI–Kriterien sagen, daß durch die Kombinationen der Kriterien auch dann eine Erreichbarkeitsanalyse von einer Transition aus einem Feature aus gestartet werden muß, wenn sie zum Beispiel mit einem anderen Kriterium als „harmlos“ eingestuft worden ist.

Die Implementation des Werkzeuges erfolgte in der objektorientierten Sprache C++. Als Compiler ist der GNU–Compiler „g++ V 2.7.2“ benutzt worden.

Der Test auf korrekten Ablauf des Werkzeuges ist mit einigen kleinen und sehr speziellen Testspezifikationen durchgeführt worden.

Aufgrund der hohen Nebenläufigkeit konnte eine in Arbeit befindliche Testspezifikation ([Mi96]) eines Intelligenten Netzwerkes nicht für eine ausführliche Analyse herangezogen werden.

Literaturverzeichnis

[Bre95] J. Bredereke *Automata–Theoretic Criteria for Feature–Interactions in Telecommunications Systems.* Interner Bericht Nr. 273/95, Fachbereich Informatik, Universität Kaiserslautern (Dezember 1995).

[Bre96] J. Bredereke *Detection of Feature Interactions in IN by Verification.* Software Concepts & Tools (1996). In Vorbereitung.

[BrGo94a] J. Bredereke, R. Gotzhein *A Case Study on Specification, Detection and Resolution of IN Feature Interactions with Estelle.* Interner Bericht Nr. 245/94, Fachbereich Informatik, Universität Kaiserslautern (Mai 1994).

[BrGo94b] J. Bredereke, R. Gotzhein *Specification, Detection and Resolution of IN Feature Interactions with Estelle.* In „Seventh International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols — FORTE '94, Proceedings“, IFIP TC6/WG6.1, Bern, Schweiz (Oktober 1994).

[Ho89] D. Hogrefe *Estelle, LOTOS und SDL. Standard–Spezifiktionssprachen für verteilte Systeme.* Springer (1989).

[ISO89] ISO/TC 97/SC 21, ISO 9074 *Information Processing Systems — Open Systems Interconnection — Estelle: A Formal Description Technique Based on an Extended State Transition Model.* (1989).

[ITU93] ITU-T *Q12xx–Series Intelligent Network Recommendations* (1993).

[Mi96] T. Michels *Spezifikation zusätzlicher Leistungsmerkmale für ein Telefonvermittlungssystem* Projektarbeit, FB Informatik, Univ. Kaiserslautern (1996)

[Sed95] R. Sedgewick *Algorithmen in C++*. Addison–Wesley (1995).

[SiSt91a] Sijelmassi, R. und Strausser, B. *The Distributed Implementation Generator: an overview and user guide*. Tech. Rep. NCSL/SNA—91/3, U.S. Dept. of Commerce, National Institute of Standards and Technology, Gaithersburg, MD (Jan. 1991).

[SiSt91b] Sijelmassi, R. und Strausser, B. *The Portable Estelle Translator: an overview and user guide*. Tech. Rep. NCSL/SNA—91/2, U.S. Dept. of Commerce, National Institute of Standards and Technology, Gaithersburg, MD (Jan. 1991).

[Sto92] B. Stoustrup *Die C++ Programmiersprache*. 2. Auflage, Addison–Wesley (1992)

[Th95] J. Thees *CONFINE — Entwurf und Implementierung eines Werkzeuges zur Analyse von Feature–Interaktionen in Estelle–Spezifikationen*. Fachbereich Informatik, Universität Kaiserslautern (April 1995).

[ThBr95] J. Thees, J. Bredereke *Ein Werkzeug zur Analyse von Feature–Interaktionen in IN*. Fachgespräch 1995 in R.Gotzhein, J.Bredereke *5. GI/ITG Workshop on Formal Description Techniques for Distributed Systems*, S. 199–208 Fachbereich Informatik, Universität Kaiserslautern (Juni 1995).

[Tur92] K. Turner *Using Formal Description Techniques — An Introduction to Estelle, LOTOS and SDL*. Wiley (1992).

FB Informatik
AG Rechnernetze

Professor Dr. Reinhard Gotzhein

Aufgabe und Betreuung: Diplom–Informatiker Jan Bredereke

Erklärung

Ich versichere hiermit, die vorliegende Arbeit selbständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Kaiserslautern, den 9. Dezember 1996