Universität Paderborn
Die Universität der Informationsgesellschaft

Institut für Informatik
Arbeitsgruppe Softwaretechnik
Warburger Str. 100
33098 Paderborn

Optimierung von Genauigkeitswerten
unscharfer Regeln

DIPLOMARBEIT
für den integrierten Studiengang Informatik
im Rahmen des Hauptstudiums II

von
Carsten Reckord
Bergstraße 53
33415 Verl

vorgelegt bei
Prof. Dr. Wilhelm Schäfer
und
Prof. Dr. Gregor Engels

Mai 2004
Eidesstattliche Erklärung

Ich versichere, daß ich die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe sowie ohne Benutzung anderer als der angegebenen Quellen angefertigt habe. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Paderborn, den 26. Mai 2004

______________________________
Carsten Reckord
Inhaltsverzeichnis

KAPITEL 1 EINFÜHRUNG ................................................................. 1
  1.1 PROBLEM BESCHREIBUNG ....................................................... 2
  1.2 ZIELSETZUNG ........................................................................ 3
  1.3 AUFBAU DER ARBEIT ......................................................... 4

KAPITEL 2 DER INFERENZPROZESS .............................................. 5
  2.1 DESIGN PATTERN SPEZIFIKATION ........................................ 5
    2.1.1 Der abstrakte Syntaxgraph ............................................... 5
    2.1.2 Die Spezifikationssprache ............................................... 6
  2.2 DAS PATTERN DEPENDENCY NET ....................................... 12
    2.2.1 Struktur des Pattern Dependency Net ................................ 12
    2.2.2 Der Erkennungsprozess .................................................... 16
  2.3 DIE FUZZY EVALUATION ...................................................... 21
    2.3.1 Struktur des Fuzzy Petrinetzes ......................................... 21
    2.3.2 Auswertung des Fuzzy Petrinetzes .................................... 23
  2.4 BENUTZERINTERAKTION ..................................................... 24

KAPITEL 3 ADAPTION VON WISSEN ............................................. 29
  3.1 MASCHINELLES LERNEN ....................................................... 29
  3.2 ADAPTION VON FUZZY-REGELN UND UNSCHARFEN PETRINETZEN .... 32
  3.3 REGRESSION UND GRADIENT DESCENDANT TECHNIKEN .......... 33
  3.4 NEURONALE NETZE UND BACKPROPAGATION ....................... 34
    3.4.1 Das Fuzzy Neural Net ..................................................... 34
    3.4.2 Konstruktion der Lernmuster .......................................... 39
    3.4.3 Der Backpropagation-Algorithmus ................................... 41
  3.5 PROBLEME VON GRADIENT DESCENDANT VERFAHREN .......... 42

KAPITEL 4 DER LÖSUNGSANSATZ: STATISTISCHE ADAPTION IM FPN ................................................................. 45
  4.1 ENTWURFSKRITERIEN .......................................................... 45
  4.2 STATISTISCHE ADAPTION .................................................... 46
4.3 ADAPTION DER VERTRAUENSWERTE .................................................... 48
  4.3.1 Senken des Vertrauenswertes .......................................................... 50
  4.3.2 Erhöhen des Vertrauenswertes ......................................................... 56
  4.3.3 Bestätigen des Vertrauenswertes ...................................................... 58
4.4 STATISTISCHE AUSWERTUNG ............................................................ 58
  4.4.1 Sammeln der Änderungsanforderungen .......................................... 59
  4.4.2 Zusammenführen der Lernmuster ..................................................... 59
  4.4.3 Adaption der Vertrauenswerte ......................................................... 60
4.5 ERWEITERUNGEN ............................................................................. 62
  4.5.1 Bereinigung der Datensätze ............................................................... 62
  4.5.2 Wahl der Ausgabestellen ................................................................. 64
  4.5.3 Situationsspezifisches Wissen und gedächtnisbehaftetes Lernen ........ 65
4.6 ADAPTION DER SCHWELLWERTE ..................................................... 66
4.7 VERALLGEMEINERUNG .................................................................... 69

KAPITEL 5 EVALUATION ............................................................................ 71
  5.1 VORGEHEN ......................................................................................... 71
  5.2 VERGLEICH MIT FUZZY NEURAL NETS ............................................ 73
  5.3 FAZIT .................................................................................................. 77

KAPITEL 6 ZUSAMMENFASSUNG UND AUSBlick ......................................... 79
  6.1 ZUSAMMENFASSUNG ........................................................................ 79
  6.2 AUSBlick .......................................................................................... 81
    6.2.1 Statistische Adaption .................................................................... 81
    6.2.2 Optimierungsmöglichkeiten im Inferenzprozess ......................... 82

ANHANG A LITERATURVERZEICHNIS ...................................................... 83

ANHANG B ABBILDUNGSVERZEICHNIS .................................................. 85
Eines der zentralen Aufgabenfelder in der Softwareentwicklung ist das **Reengineering**. Im Gegensatz zu der vollständigen Neuentwicklung eines Softwaresystems ist das Ziel des Reengineering die Weiterentwicklung existierender Systeme, um sie etwa an geänderte Anforderungen anzupassen oder um neue Funktionalität zu ergänzen. Insbesondere große Softwaresysteme werden so häufig von unterschiedlichen Entwicklern über mehrere Generationen hinweg entwickelt und gepflegt.

Um ein solches sogenanntes **Legacy-System** erfolgreich weiterentwickeln und warten zu können, ist zunächst ein gutes Verständnis von Aufbau und Funktionsweise des Systems erforderlich. Hierzu kann aber in der Praxis häufig nicht auf eine vorhandene Design-Dokumentation zurückgegriffen werden, da diese - sofern überhaupt vorhanden - zumeist nur das initiale Design des Systems reflektiert, während spätere Änderungen und Erweiterungen zumeist undokumentiert bleiben. Auch die ursprünglichen Entwickler stehen insbesondere bei älteren Systemen häufig nicht mehr zur Verfügung, so daß für die Einarbeitung eines neuen Entwicklers in das System zumeist nur der Systemquelltext genutzt werden kann.


Design Pattern bieten hier eine gute Möglichkeit, auch komplexere Zusammenhänge zwischen mehreren Elementen kompakt auszudrücken und erlauben so dem Entwickler einen schnelleren und besseren Überblick über das Gesamtsystem. Design Pattern (oder Entwurfsmuster), die erstmals von Gamma, Helm, Johnson und Vlissides in [GHJV95] vorgestellt worden sind, sind Standard-Lösungsansätze für häufig wiederkehrende Designaufgaben. Ursprünglich als Entwurfsprinzip auf hohem Abstraktionsniveau für das Forward Engineering konzipiert, abstrahieren sie von einzelnen Elementen hin zu einem übergeordneten Design-Prinzip. Umgekehrt kann aber auch von dem Design Pattern auf die Funktionen der daran beteiligten Systemkom-
1.1 Problembeschreibung


Ein weiteres Problem stellt die Größe der zu untersuchenden Systeme dar. Typische Software- systeme liegen häufig in Größenordnungen von einigen hunderttausend bis Millionen Zeilen Quelltext, woraus sich besonders hohe Anforderungen an die Skalierbarkeit des verwendeten Erkennungsmechanismus ergeben.


Das Variantenproblem wird durch die Verwendung von Unschärfe in der Spezifikation gelöst. So ist es möglich, durch eine einzelne, möglichst allgemein gehaltene Musterspezifikation mehrere Varianten eines Pattern abzudecken.


Da die Vertrauenswerte der gefundenen Patternausprägungen dem Benutzer als Grundlage für die weitere Auswertung und mögliche Kontrollen der Ergebnisse dienen, haben sie entscheidenden Anteil an der Qualität der Patternerkennung. Die Festlegung der für die Fuzzy-Auswertung entscheidenden Parameter und damit der Qualität der Vertrauenswerte ist zur Zeit jedoch dem Benutzer überlassen. Die optimalen Werte für diese Parameter hängen direkt von der Qualität
1.2 Zielsetzung

Im Rahmen dieser Diplomarbeit wird ein Adoptionsverfahren entwickelt, das die Anpassung der Fuzzy-Parameter für die Design Pattern Erkennung nach [Wen01] ermöglicht. Grundlage für die Adaption sollen die manuellen Anpassungen der Fuzzy-Werte durch den Benutzer während eines Erkennungsprozesses sein. Das Ziel ist somit die Realisierung der in Abbildung 1.1
fehlenden Rückkopplung von den Benutzerkorrekturen zu den Patternspezifikationen mit Hilfe dieses Adoptionsverfahren.

Eine Adaption der Fuzzy-Parameter auf Basis konnektionistischer Methoden (Fuzzy Neural Nets, FNN, [Str99]) hat sich in einem ähnlichen Projekt ([Jah99]) aufgrund der Laufzeitkomplexität des Lernenden-Netzes als nicht anwendbar herausgestellt, da die Auswertung des Netzes ein Vielfaches der übrigen Analysezeit benötigt hat. Ähnliches läßt sich für andere klassische iterative Lernansätze herleiten. Daher wird bei dem zu entwickelnden Adoptionsverfahren besonderes Augenmerk auf die Skalierbarkeit hinsichtlich Laufzeit und Speicherbedarf gelegt, damit das Verfahren auch für Analysen von realistischen Projektgrößen im Umfang von mehreren hunderttausend LOC (Lines Of Code) noch anwendbar bleibt.

### 1.3 Aufbau der Arbeit


Thema von Kapitel 4 ist der Lösungsansatz dieser Arbeit. Hier wird zunächst beispielhaft die Änderung einzelner Fuzzy-Parameter untersucht, bevor anschließend die Auswirkungen auf Teilpattern und darauf basierend das entgültige Lernverfahren präsentiert werden.


In Kapitel 6 werden schließlich die Ergebnisse dieser Arbeit zusammengefaßt und im Ausblick offene Fragen und weitere lohnenswerte Ansätze für die rechnergestützte Optimierung des Inferenzprozesses aufgegriffen.
KAPITEL 2

Der Inferenzprozess


2.1 Design Pattern Spezifikation


2.1.1 Der abstrakte Syntaxgraph


Die Verwendung eines abstrakten Syntaxgraphen zur Repräsentation von Quelltext hat zahlreiche Vorteile gegenüber der direkten Verwendung des Quelltextes. So wird im ASG von der For-
matierung des Quelltextes abstrahiert. Einfache syntaktische Variationen im Quellcode, wie etwa \texttt{i++} und \texttt{i=i+1} oder unterschiedliche Schleifenkonstrukte, werden im ASG einheitlich dargestellt. So ist etwa eine Unterscheidung zwischen for- und while-Schleifen im ASG nicht mehr erforderlich. Neben der Vereinfachung der Darstellung bietet die ASG-Darstellung aber auch zusätzliche Informationen über Variablenabläufe, Namensauflösung und mehr, die während des Parsens aus dem Quelltext gewonnen werden.

Abbildung 2.1 zeigt beispielhaft Auszüge eines Java-Quelltextes und des entsprechenden abstrakten Syntaxgraphen in Form eines Objektdiagrammes. Der Syntaxgraph des Methodenrumpfes ist hier zur besseren Darstellung etwas vereinfacht abgebildet. Detaillierte Informationen über das verwendete Metamodell und den abstrakten Syntaxgraph sind in [FNT98] zu finden.

2.1.2 Die Spezifikationssprache


Design Pattern nach Gamma

Abbildung 2.2 zeigt das Klassendiagramm des Composite Pattern nach Gamma et al. Das Composite Pattern beschreibt eine Baumstruktur, in der Baum- und Blattknoten auf gleiche Weise behandelt werden können.


Das Variantenproblem

Eine formale Spezifikation von Design Pattern und eine darauf aufbauende Mustererkennung werden erheblich erschwert durch die Vielzahl möglicher Varianten eines Patterns.

Durch die informale Patternbeschreibung nach Gamma et al. sind dem Entwickler für die Umsetzung eines Patterns viele Freiräume gelassen, um das Pattern an die eigenen Anforderun-
2.1 Design Pattern Spezifikation

gen anzupassen. Hierdurch entstehen viele strukturelle Variationen desselben Patterns, die im Erkennungsprozess und damit auch bereits bei der formalen Spezifikation des Patterns zu berücksichtigen sind.

So ist etwa auch das Klassendiagramm aus Abbildung 2.2 lediglich als eine Beispielausprägung zu betrachten. Eine häufig gefundene strukturelle Variation des Composite-Patterns, bei der die Rollen der ursprünglichen Klassen Component und Composite in der Klasse Composite vereinigt sind, ist in Abbildung 2.3 dargestellt.

Eine weitere Schwierigkeit für die Spezifikation und Mustererkennung stellt die Umsetzung höherer objektorientierter Konstrukte bei der Implementierung dar. Insbesondere strukturelle Beziehungen zwischen den Klassen eines Patterns sind in diesem Zusammenhang problematisch, da die meisten objektorientierten Programmiersprachen etwa für bidirektionale Assoziationen oder zu-n-Beziehungen keine Sprachelemente besitzen. Solche Beziehungen werden zumeist durch Referenzattribute beziehungsweise Containerklassen und entsprechende Zugriffsmethoden abgebildet. Die konkreten Implementierungen können jedoch stark voneinander abweichen. Abbildung 2.4 zeigt beispielsweise zwei unterschiedliche Möglichkeiten, die zu-n-Assoziation children aus Abbildung 2.3 zu implementieren. In Abbildung 2.4a wird hierzu ein Standardcontainer-Objekt verwendet (Zeile 3), auf das über Zugriffsmethoden addToChildren, removeFromChildren und iteratorOfChildren elementweise zugegriffen werden kann (Zeilen 4-12). Die Implementierung aus Abbildung 2.4b verwendet dagegen ein Array (Zeile 3), das dem Benutzer über die beiden Zugriffsmethoden setChildren und getChildren (Zeilen 4-8) zur Verfügung gestellt wird. Die Organisation der Elemente in dem Array bleibt dabei dem Benutzer überlassen.

Aufgrund der hohen Anzahl möglicher Design- und Implementierungsvarianten ist die separate Spezifikation aller möglichen Variationen eines Design Patterns kaum zu realisieren. Daher wird eine Spezifikationssprache benötigt, die einen möglichst großen Teil dieser Variationen in wenigen Spezifikationen auszudrücken vermag.

Patternsspezifikation

Die formale Spezifikation eines Patterns erfolgt mit Hilfe von Graphtransformationenregeln¹ auf dem abstrakten Syntaxgraph. Die verwendete Notation orientiert sich an der Notation von UML

¹ Eine formale Definition für die hier verwendete Form von Graphtransformationenregeln ist in [Zün01] zu finden.
2.1 Design Pattern Spezifikation

Abbildung 2.4: Implementierung von zu-n-Assoziationen

Abbildung 2.5: Formale Spezifikation des Composite-Patterns

Kollaborationsdiagrammen. Eine mögliche Spezifikation des Composite Patterns ist in Abbildung 2.5 dargestellt. Es ist zu erkennen, daß sich die Spezifikation in zwei Teile gliedert.


Der fett dargestellte Rahmen des MultiDelegation Patterns weist es als einen Trigger für das Composite Pattern aus. Trigger werden in dem im nächsten Abschnitt beschriebenen Inferenzalgorithmus zur Steuerung der Patternsuche verwendet. Wird während der Inferenz ein Objekt vom Typ des Triggers gefunden, so wird die Analyse der betreffenden Pattern ausgelöst, die die-
sen Trigger haben. Daher ist es erforderlich, daß in jeder Musterspezifikation ein Element als Trigger markiert ist.
Es ist zu erkennen, daß in der Patternspezifikation die Klassen Leaf und Client nicht repräsentiert sind. Um eine Vielzahl von Mustervarianten abdecken zu können, sollte die Spezifikation stets nur die minimal erforderlichen Teile eines Pattern umfassen. Da die Klasse Client in der Patternbeschreibung nur der Veranschaulichung einer Einbindung des Patterns dient, kann sie bei der Spezifikation weggelassen werden. Die Klasse Leaf ist ebenfalls nicht entscheidend für ein Vorkommen des Musters, da die entscheidende Baumstruktur bereits durch die Component- und Composite-Klasse definiert wird. Die Existenz konkreter Blattklassen ist hingegen für das Composite-Pattern nicht entscheidend. So sind etwa häufig in Softwarebibliotheksten Instanzen des Composite-Patterns zu finden, zu denen erst der Benutzer der Bibliothek die konkreten Blattklassen implementiert. Einer Musterspezifikation, die die Leaf-Klasse umfaßt, würde solche Instanzen jedoch nicht mit einschließen.

Vertrauens- und Schwellwerte
Durch eine möglichst allgemein gehaltene Patternspezifikation und die Beschränkung auf die Hauptbestandteile eines Pattern ist es möglich, mit einer Spezifikation eine Vielzahl von Mustervarianten abzudecken. Andererseits wächst hierdurch aber auch die Chance, die spezifizierte Struktur im ASG zu finden, ohne daß es sich tatsächlich um ein Patternvorkommen handelt. Um dem Reverse Engine die Identifikation solcher False Positives zu erleichtern, ist es sinnvoll, ihm statt einer absoluten Aussage über ein Patternvorkommen eine Gütebewertung über die Analyseergebnisse zu geben, auf deren Basis er ausgewählte Ergebnisse genauer untersuchen kann. Diese Gütebewertung geschieht durch eine Fuzzy Evaluation, die jeder gefundenen Musterausprägung einen Fuzzy-Wert zuweist und in Abschnitt 2.3 noch beschrieben wird. Die Parameter, die als Grundlage für diese Evaluation dienen, sind die Vertrauens- und Schwellwerte, die allen Musterspezifikationen zugeordnet sein müssen.
Der Vertrauenswert ist ein prozentualer Wert zwischen 0 und 100, der das Vertrauen des Reengineers in die Genauigkeit der Musterspezifikation ausdrückt. Eine exakte Spezifikation, die nie False Positives produziert, erhält demnach einen Wert von 100.


**Das Pattern-Klassendiagramm**


2.2 Das Pattern Dependency Net


2.2 Das Pattern Dependency Net

Nachdem im letzten Abschnitt die Musterspezifikationssprache vorgestellt worden ist, befaßt sich dieser Abschnitt mit dem Algorithmus zur Erkennung der spezifizierten Muster.

2.2.1 Struktur des Pattern Dependency Net


Aufbau des Pattern Dependency Net

Namen des Prädikates. Axiome stellen hingegen Basisfakten dar, die unmittelbar aus dem geparsen Quelltext beziehungsweise dem ASG hervorgehen. Sie repräsentieren ausgewählte ASG-Elemente, die als Ausgangspunkte für die Patternsuche dienen. Im PDN werden sie durch ein Rechteck mit fett gezeichnetem Rand dargestellt.


Sowohl einfache Prädikate als auch Axiome können durch gerichtete Implikationskanten mit nachfolgenden Implikationen verbunden sein. Die Richtung der Implikationskanten wird durch eine offene Pfeilspitze angegeben. Der Vorbereich einer Implikation enthält die Prädikate zu allen Pattern, die in der zugehörigen Musterspezifikation als Teilpattern enthalten sind. Außerdem werden alle ASG-Objekte, die als Trigger markiert sind, durch gleichnamige Axiome im Vorbereich repräsentiert. Ist ein Axiom oder Prädikat im Vorbereich der Implikation in der Spezifikation als Trigger festgelegt, so ist die zugehörige Implikationskante im PDN eine Triggerkante, die fett dargestellt wird. Analog werden Prädikate zu optionalen Elementen der Spezifikation durch gestrichelt dargestellte optionale Kanten mit der Implikation verbunden.


Neben den Implikationskanten gibt es außerdem noch Vererbungskanten, die in der UML-üblichen Notation die im Musterklassendiagramm definierte Vererbungshierarchie zwischen den Pattern reflektieren.

2.2 Das Pattern Dependency Net

Auf die Ebene der ASG-Elemente, die hier durch die Axiome UMLMethod, UMLAttr und UMLGeneralization vertreten sind.

Abbildung 2.7: Ausschnitt eines Pattern Dependency Nets

Anpassung der Syntax


Das GFRN, wie es in [Jah99] beschrieben ist, enthält einige Elemente, die sich im praktischen Einsatz bei der Design Pattern Erkennung als nicht erforderlich oder unpraktisch erwiesen haben. Da diese Elemente sowohl die Musterspezifikation als auch den Erkennungsprozess unnötig komplizieren, sind sie mittlerweile aus dem Netz entfernt worden. Das resultierende Netz stellt durch diese Einschränkungen somit eine spezielle Variante des ursprünglichen GFRN dar. Um diesen Umstand zu verdeutlichen, ist daher der Name Patten Dependency Net...
für das hier verwendete Netz eingeführt worden. Auf die Änderungen des PDN gegenüber dem ursprünglichen GFRN soll im Folgenden kurz eingegangen werden.

Die gravierendste Vereinfachung gegenüber dem ursprünglichen GFRN stellt der Verzicht auf negative Implikationskanten dar. Negative Implikationskanten drücken einen gegenseitigen Ausschluß zwischen den beteiligten Prädikaten aus. Im Kontext des Datenbank-Reengineering sind sie verwendet worden, um die Auswirkungen widersprüchlichen Wissens zu modellieren.


Formale Beschreibung


Definition 2.1: Pattern Dependency Net

Ein Pattern Dependency Net ist ein Tupel \((P, I, E_I, E_G, cf, th)\), für das gilt:
- \(P = P_A \cup P_p\) ist die endliche Menge der Prädikate, aufgeteilt in Axiome und einfache Prädikate.
- \(I\) ist die endliche Menge der Implikationen.
Das Pattern Dependency Net

2.2.1 Das Pattern Dependency Net

E₁ ⊆ (P × I) ∪ (I × P₁) ist die Menge der gerichteten Implikationskanten von Prädikaten zu Implikationen und von Implikationen zu einfachen Prädikaten. Es gilt:

∀p ∈ P₁, p ∈ E₁
∀i ∈ I, q, r ∈ P : (q, i) ∈ E₁ ∧ (i, r) ∈ E₁
∀(i, p) ∈ E₁ : (j, p) = E₁ ∧ (i, s) = E₁ ∀j ∈ I ∖ {i}, s ∈ P₁ ∖ {p₁}

E₁ ⊆ (P × P₁) ist die Menge der Vererbungskanten. Es gilt für p₁, p₂ ∈ P:

(p₁, p₂) ∈ E₁ ⇒ (p₁, q) = E₁ ∀q ∈ E₁ ∖ {p₂}

• cf : I → [0, 100] ordnet jeder Implikation einen Vertrauenswert (engl. Confidence) zu.
• th : I → [0, 100] ordnet jeder Implikation einen Schwellwert (engl. Threshold) zu.

Definition 2.2: Vor- und Nachbereich

Für i ∈ I und p ∈ P gilt:

pre : P₁ → I mit pre(p) = j ⇔ (j, p) ∈ E₁ ordnet jedem einfachen Prädikat die eindeutig bestimmte Implikation in seinen Vorbereich zu.

pre : I → P₁ mit pre(i) = {q ∈ P₁ | (q, i) ∈ E₁} ordnet jeder Implikation ihren Vorbereich zu. Es gilt pre(i) ≠ ∅ ∀ i ∈ I.

pre : P₁ → P₁ mit prep(p) = {q ∈ P₁ | (q, j) ∈ E₁ ∧ j = pre(p)} ordnet jedem Prädikat alle direkten Vorgänger-Prädikate zu.

post : P₁ → P₁ mit post(p) = {j ∈ I | (i, j) ∈ E₁} ordnet jedem Prädikat seinen Nachbereich zu.

post : I → P₁ mit post(i) = q ⇔ (i, q) ∈ E₁ ordnet jeder Implikation das eindeutig bestimmte Prädikat in ihrem Nachbereich zu.

post : P₁ → P₁ mit post(p) = {q ∈ P₁ | (j, q) ∈ E₁ ∧ j ∈ post(p)} ordnet jedem Prädikat alle direkten Nachfolger-Prädikate zu.

Definition 2.3: Transitiver Abschluß und Kreise

pre : P₁ → P₁ mit prep(p) = {q ∈ P₁ | q ∈ pre(p) ∨ ∃ r ∈ prep(p) : q ∈ prep(r)} . Er enthält alle Prädikate aus dem Teilgraph unterhalb von p.

post : P₁ → P₁ mit post(p) = {q ∈ P₁ | q ∈ post(p) ∨ ∃ r ∈ post(p) : q ∈ post(r)} . Er enthält alle Prädikate aus dem Teilgraph oberhalb von p.

cycl : P₁ → P₁ mit cycl(p) = {q ∈ P₁ | q ∈ post(p) ∧ p ∈ post(p)} ordnet jedem Prädikat p die Menge der Prädikate zu, die mit p auf einem Kreis im PDN liegen.

2.2.2 Der Erkennungsprozess

Mit Hilfe des Pattern Dependency Net kann nun der Erkennungsprozess definiert werden. Grundlage für den Erkennungsprozess sind ein Katalog mit Musterspezifikationen und das daraus generierte PDN. Der Musterkatalog legt fest, nach welchen Mustern zu suchen ist und gibt mit den Graphtransformationsregeln der Musterspezifikationen die Suchkriterien für das Pat-
2.2 Das Pattern Dependency Net

tern Matching vor. Das PDN enthält alle für den Erkennungsprozess wichtigen Abhängigkeiten zwischen den Spezifikationen und steuert damit die Reihenfolge, in der die Anwendung der Spezifikationen erfolgt.


Um die Integration des Reverse Engineers zu unterstützen, ist daher ein halbautomatischer, inkrementeller Erkennungsprozess realisiert worden, der schnell zu guten Zwischenergebnissen führt und jederzeit vom Benutzer unterbrochen und nach eventuellen Korrekturen ohne den Verlust von Zwischenergebnissen fortgesetzt werden kann. Änderungen durch den Benutzer, auf die in Abschnitt 2.4 eingegangen wird, gehen dabei unmittelbar in die weiteren Analysen ein.

Die Inferenzmaschine

Der Mustererkennungsprozess wird gesteuert durch die Inferenzmaschine. Diese sucht nach Instanzen der im Musterkatalog spezifizierten Pattern.


Formal gilt:

**Definition 2.4: Rang**

a) Der Rang eines Prädikates \( p \) ist gegeben durch die Funktion \( \text{rang}: P \rightarrow \mathbb{N} \) mit

\[
\text{rang}(p) = \begin{cases} 
0 & \text{falls } p \in P_A \\
\text{rang}(p) > \text{rang}(q) & \forall q \in \text{pre}(p) \setminus \text{cycl}(p) \\
\text{rang}(p) = \text{rang}(q) & \forall q \in \text{cycl}(p) 
\end{cases}
\]

b) Der Rang einer Implikation \( i \) ist gegeben durch die Funktion \( \text{rang}: I \rightarrow \mathbb{N} \) mit

\[
\text{rang}(i) = \text{rang}(	ext{post}(i))
\]

Diese Definition läßt verschiedene Möglichkeiten zu, die Ränge nicht voneinander abhängiger Prädikate festzulegen und so etwa die Erkennungsreihenfolge zu optimieren. Im Folgenden wird soweit nicht anders angegeben stets der natürliche Rang verwendet.

**Definition 2.5: Natürlicher Rang**

Der natürliche Rang \( \text{rang}_0 \) eines Prädikates \( p \) ist gegeben durch

\[
\text{rang}_0(p) = \text{Max}\{r'(q) | q \in \text{cycl}(p) \cup \{p\} \} \text{ mit} \\
r'(p) = 0 \text{ für } p \in P_A \text{ und} \\
r'(p) = \text{Max}\{\text{rang}_0(q) | q \in \text{pre}_P(p) \setminus \text{cycl}(p) \} + 1 \text{ für } p \in P_P
\]

Mit Hilfe des Ranges kann nun der Erkennungsalgorithmus beschrieben werden. Abbildung 2.8 zeigt eine Momentaufnahme des Algorithmus. Das schwarze Oval zeigt die Annotation einer Generalisierung, die bereits in der Bottom-Up-Analyse erkannt worden ist. Die grauen Ovale symbolisieren noch nicht abgeschlossene Top-Down-Analysen der entsprechenden Pattern. Die Zahlen oberhalb der Ovale sind die Rangnummern der Prädikate, die sich nach Definition 2.5 für das PDN aus Abbildung 2.7 ergeben. Die gerichteten Kanten verdeutlichen die Reihenfolge des Analyseprozesses und die Kantenbeschriftungen bezeichnen die Kontextobjekte, die für die Anwendung der Graphtransformationenregeln von der Inferenzmaschine bereitgestellt werden.
Das Pattern Dependency Net

2.2 Das Pattern Dependency Net

Der Inferenzprozess

Bottom-Up Strategie


Zur Initialisierung der Bottom-Up Queue werden zunächst alle ASG-Objekte ermittelt, die Axiomen im PDN entsprechen. Für jedes dieser Objekte werden im PDN die Implikationen aus seinem Nachbereich bestimmt, mit denen es über eine Triggerkante verbunden ist. Für jede gefundenene Implikation wird ein Tupel bestehend aus der Implikation und dem ASG-Objekt als Kontext in die Bottom-Up Queue eingetragen.


2.2 Das Pattern Dependency Net

Es ist also zu erkennen, daß die Priorisierung hochrangiger Implikationen schnell zur Analyse komplexerer Pattern führt.


Top-Down Strategie

Kann eine Implikation im Bottom-Up Modus nicht schließen, weil für die Elemente aus ihrem Vorbereich noch keine Annotationen im abstrakten Syntaxgraph existieren, so wird das Tupel aus Implikation und Kontext in die Top-Down Queue eingetragen. Hierbei handelt es sich ebenfalls um eine Prioritätswarteschlange, die jedoch nach aufsteigendem Rang sortiert ist, so daß Tupel mit niedrigem Rang bevorzugsweise aus der Prioritätswarteschlange geholt werden. Im Top-Down Modus wird dann versucht, die unerfüllten Vorbedingungen der Implikation zielgetrieben herzuleiten. Diese Herleitung kann im Gegensatz zu anderen Top-Down Verfahren effizient ausgeführt werden, weil durch den vorgegebenen Kontext der zu analysierende Suchraum auf einige wenige Elemente beschränkt ist.

Bei der Abarbeitung der Top-Down Queue wird wiederum sukzessive das erste Element in der Queue betrachtet. Kann die zugehörige Implikation nicht schließen, weil Bedingungen aus ihrem Vorbereich nicht erfüllt sind, werden die Implikationen zu den fehlenden Pattern mit einem entsprechenden Kontext in die Top-Down Queue eingetragen. Der erforderliche Kontext kann aus dem Kontext der fehlgeschlagenen Implikation und der zugehörigen Patternspezifikation ermittelt werden. Hierfür kann es unter Umständen mehrere Möglichkeiten geben. In diesem Fall wird für jeden möglichen Kontext ein Tupel in die Liste eingetragen. Im Beispiel ergeben sich sowohl für die MultiDelegation als auch für die Association jeweils die UMLClass-Objekte c1 und c2 als mögliche Kontexte.

Im Gegensatz zum Bottom-Up Modus wird ein Tupel aus der Top-Down Queue erst entfernt, wenn die abgelegte Implikation entweder erfolgreich schließen konnte oder die Suche nach einem fehlenden Muster aus dem Vorbereich der Implikation für alle ermittelten Kontexte fehlgeschlagen ist. Durch die Priorisierung niederrangiger Analysen ist gewährleistet, daß die Suche so schnell wie möglich auf Basisfakten des ASG zurückgeführt wird. Dadurch wird ein schnelles Fehlschlagen nicht erfüllbarer Implikationen erreicht. Ist ein Pattern erfolgreich erkannt worden, so werden die dadurch getriggerten Implikationen für die spätere Auswertung in die Bottom-Up Queue eingetragen.

2.3 Die Fuzzy Evaluation

Nachdem die Mustererkennung angehalten hat, folgt als nächstes die Bewertung der Analyseergebnisse. Da die Musterspezifikationen, die die Basis der Analysen bilden, sehr allgemein formuliert sind und False Positives unter den Ergebnissen nicht ausgeschlossen werden können, wird jeder Annotation zu einer Patterninstanz ein Fuzzywert zwischen 0 und 100 zugeordnet. Dieser Fuzzywert soll dem Reverse Engineer als Anhaltspunkt für die Verlässlichkeit des Ergebnisses dienen und ihn bei der Auswahl der Patterninstanzen unterstützen, für die eine weitere Analyse sinnvoll erscheint. Die Berechnung dieser Fuzzy-Werte erfolgt durch ein Fuzzy Petrinetz\[Wen01, Jah99\], das während des Erkennungsprozesses im Hintergrund mitaufgebaut wird.

2.3.1 Struktur des Fuzzy Petrinetzes

Das Fuzzy Petrinetz (FPN) ist ein klassisches Stellen-Transitions-Netz\[Jah99\]. Die Stellen im Fuzzy Petrinetz entsprechen Analyseergebnissen, während die Transitionen Implikationsregeln repräsentieren.


Durch die in Abschnitt 2.2 beschriebenen Anpassungen ist auch im FPN sichergestellt, daß jede Transition genau eine Stelle in ihrem Nachbereich hat. Umgekehrt hat auch jede Stelle, die kein Axiom repräsentiert, genau eine Transition in ihrem Vorbereich. Da Zyklen im PDN zudem im Erkennungsprozess eine endliche Rekursion bewirken, die im FPN keine Kreise erzeugt, und auf negative Implikationskanten, die Rückkopplungskreise im FPN bewirken können, verzichtet worden ist, ist das resultierende FPN zyklenfrei und streng vorwärtsgerichtet.

Jede Stelle im FPN besitzt ein Fuzzy Belief Marking, bei dem es sich um eine natürliche Zahl zwischen 0 und 100 handelt. Das Fuzzy Belief Marking (FBM) entspricht nach der Auswertung des FPN dem Fuzzy-Wert der zugehörigen Annotation. Die Berechnung der Fuzzy Belief Mar-
2.3 Die Fuzzy Evaluation


Abbildung 2.9 zeigt ein Beispiel-FPN zu dem PDN aus Abbildung 2.7. Es ist zu erkennen, daß die Struktur des FPN der des PDN stark ähneln. Da die Stellen des FPN konkrete Instanzen der im PDN enthaltenen Pattern widerspiegeln und die Transitionen für erfolgreiche Anwendungen der Implikationen stehen, kann das FPN als Instanzgraph mit dem PDN als Typgraph aufgefasst werden.
Formal ist ein Fuzzy Petrinetz wie folgt definiert.

**Definition 2.6: Fuzzy Petrinetz**

Ein Fuzzy Petrinetz ist ein Tupel \((S, T, F, cf, th, fbm)\) mit:

- \(S\) ist eine endliche Menge von Stellen
- \(T\) ist eine endliche Menge von Transitionen mit \(T \cap S = \emptyset\)
- \(F \subseteq (S \times T) \cup (T \times S)\) ist die Kantenmenge des FPN und wird als Flußrelation bezeichnet
- \(cf: T \to [0, 100]\) ist die Konfidenzfunktion, die jeder Transition einen Vertrauenswert zuweist
- \(th: T \to [0, 100]\) ordnet jeder Transition einen Schwellwert zu.
- \(fbm: S \to [0, 100]\) ist die Markierungsfunktion, die jeder Stelle ein Fuzzy Belief Marking zuordnet. Hierbei wird zwischen dem errechneten und dem benutzerdefinierten FBM unterschieden. Es gilt:
  \(fbm_c: S \to [0, 100]\) ordnet jeder Stelle das errechnete FBM zu.
  \(fbm_u: S \to [0, 100] \cup \{-1\}\) ordnet jeder Stelle ein benutzerdefiniertes FBM oder den Sonderwert -1 zu, wobei -1 die Abwesenheit eines benutzerdefinierten Wertes symbolisiert.

Damit ergibt sich für eine Stelle \(s\):

\[
fbm(s) = \begin{cases} 
  fbm_u(s) & \text{falls } fbm_u(s) \neq -1 \\
  fbm_c(s) & \text{sonst}
\end{cases}
\]

- \(typ_s: S \to P\) und \(typ_t: T \to I\) ordnet jeder Stelle und jeder Transition die zugehörigen Elemente des PDN zu
- \(inst_p: P \to P(S)\) und \(inst_i: I \to P(T)\) ordnet jedem Prädikat und jeder Implikation die zugehörigen Instanzen im FPN zu

Definition 2.2 und Definition 2.3 zu Vor- und Nachbereichen gelten entsprechend, wobei Prädikate durch Stellen und Implikationen durch Transitionen zu ersetzen sind.

### 2.3.2 Auswertung des Fuzzy Petrinetzes

Zu Beginn des Auswertungsprozesses sind alle Stellen des Fuzzy Petrinetzes, die Axiome des PDN repräsentieren, mit einem Fuzzy Belief Marking von 100 markiert. Dies drückt die absolute Sicherheit der Axiome aus, die ja für Fakten im ASG stehen. Alle anderen Stellen sind mit einem FBM von 0 initialisiert.

2.4 Benutzerinteraktion


**Definition 2.7: Fuzzy Truth Token**

Das Fuzzy Truth Token zu einer Transition \( t \) wird berechnet durch \( \tilde{ftt}(t) = \min \{ \text{fbm}(s) | s \in \text{pre}(t) \} \) mit

\[
\tilde{ftt}(t) = \begin{cases} 
\min \{ cf(t), \tilde{ftt}(t) \} & \text{falls } \tilde{ftt}(t) \geq \text{th}(t) \\
0 & \text{sonst}
\end{cases}
\]

Zur Berechnung des FTT wird also zunächst das Minimum über die Fuzzy Belief Markings aller Stellen im Vorbereich der Transition gebildet. Ist dieser Wert größer als der Schwellwert der Transition, so ist die Transition aktiv und das FTT ergibt sich aus dem Minimum des berechneten Wertes und des Schwellwertes der Transition. Die Werte aus dem Vorbereich einer Transition und dem Vertrauenswert, die das Minimum des FTT prägen, werden als dominante Werte bezeichnet. Sie haben eine wichtige Bedeutung für die in Kapitel 3 und Kapitel 4 vorgestellten Lernverfahren.

In der zweiten Phase schalten alle aktiven Transitionen und die errechneten FBMs nehmen die Werte der Fuzzy Truth Tokens aus dem Vorbereich der Stellen an\(^1\). Ist für eine Stelle ein Fuzzy-Wert vom Benutzer vorgegeben, so wird dieser statt des errechneten Wertes als FBM angenommen und für die weitere Auswertung des FPN verwendet.

Abbildung 2.10 zeigt das Fuzzy Petrinetz aus Abbildung 2.9 nach der vollständigen Auswertung. Sobald die Auswertung des FPN abgeschlossen ist, werden die neu ermittelten Werte an die Annotationen weitergegeben und mit diesen zusammen in den jeweiligen Klassendiagrammen angezeigt. Anschließend ist der Inferenzprozeß abgeschlossen und der Reverse Engineer kann die gefundenen Pattern analysieren und die Fuzzy-Bewertung der Annotationen seinen Schätzungen entsprechend anpassen. Dies wird im folgenden Abschnitt beschrieben.

2.4 Benutzerinteraktion


---

2.4 Benutzerinteraktion


Abbildung 2.10: Fuzzy Petrinetz nach der Auswertung

Abbildung 2.11: Inferenzprozess Statechart

Um dem Benutzer die nötige Zeit für seine Analysen zu geben, kann er den automatischen Erkennungsprozess jederzeit unterbrechen und später fortsetzen, ohne dabei Zwischenergebnisse zu verlieren. Dies ist möglich, da alle bis dahin gefundenen Muster Zwischenergebnisse darstellen, die durch entsprechende Annotationen im ASG festgehalten sind und durch die weitere automatische Analyse nicht mehr verändert werden. Auch die Daten einer unvollständigen Top-Down-Analyse gehen bei einer Unterbrechung nicht verloren, da alle für die Fortsetzung relevanten Informationen in den entsprechenden Warteschlangen festgehalten sind.

Ist der Erkennungsprozess abgeschlossen oder wird er vom Benutzer unterbrochen, so erfolgt automatisch die Fuzzy-Bewertung der bis dahin gefundenen Ergebnisse wie in Abschnitt 2.3 beschrieben. Anhand der berechneten Fuzzy-Werte kann der Benutzer nun die Muster für seine weiteren Untersuchungen auswählen. Stellt er bei diesen Untersuchungen fest, daß ein Muster falsch erkannt oder bewertet worden ist, so kann er dies korrigieren und damit die Erkennungsgenauigkeit der darauf aufbauenden Analysen verbessern. Bei der Fuzzy-Bewertung erhalten Muster in der Regel einen Fuzzy-Wert unterhalb des Maximalwertes von 100. Ist sich der Benutzer sicher, daß ein Muster korrekt erkannt worden ist, so kann er diesen Wert auf 100 erhöhen. Stellt er hingegen fest, daß es sich bei dem gefundenen Muster um ein False Positive handelt, so muß er das Ergebnis zurückweisen. Hierzu setzt er den Fuzzy-Wert des Musters auf 0. Ist sich der Benutzer hinsichtlich der Korrektheit des Musters nicht sicher, kann er natürlich entsprechende Werte zwischen 0 und 100 vergeben.

Die Anpassung der Fuzzy-Werte durch den Benutzer ist für diese Arbeit von vorrangigem Interesse, da sie die Grundlage für die Ausnivellierung der Fuzzy-Parameter darstellt, die Gegenstand der weiteren Arbeit ist. Daneben hat der Benutzer jedoch auch noch die Möglichkeit, selbst neue Annotationen zu erzeugen, falls ein Muster nicht von dem Erkennungsmechanismus gefunden worden ist. Auch diese Interaktion ist mit dem Algorithmus aus Abschnitt 2.2 einfach zu realisieren. Es muß lediglich für die neue Annotation eine Stelle im FPN erzeugt und die Bottom-Up Queue um die von dem Muster getriggerten Implikationen ergänzt werden. Werden häufig Muster nicht erkannt oder gibt es andere Probleme mit den Analyseergebnissen, so kann der Benutzer an dieser Stelle auch eine erneute Kontrolle der Musterspezifikationen vornehmen und diese gegebenenfalls anpassen. Da sich hierdurch die Grundlage der Mustererkennung ändert, kann in diesem Fall jedoch der Erkennungsprozess nicht wieder aufgenommen werden, sondern muß neu gestartet werden.
2.4 Benutzerinteraktion
Adaption von Wissen


3.1 Maschinelles Lernen

An dieser Stelle soll zunächst geklärt werden, was unter (maschinellem) Lernen zu verstehen ist. Außerdem werden weitere Begriffe aus dem Bereich des Lernens erläutert, um eine Basis für die weiteren Ausführungen zu schaffen.

Lernen

Um den Begriff des maschinellen Lernens zu erläutern, ist es hilfreich, zunächst generell den Begriff des Lernens zu betrachten. Allgemein beschreibt Lernen die Vermehrung des eigenen Wissens. Diese Definition ist jedoch zu unspezifisch, um sie in einen algorithmischen Ansatz zu übertragen. Eine spezifischere Definition von Lernen beschreibt das Erlernen von Verhaltensweisen:

„Lernen beschreibt das Anpassen von Verhalten auf der Basis gesammelter Erfahrungen“

Hieraus läßt sich unmittelbar eine Definition von maschinellem Lernen ableiten. Verhalten wird in diesem Kontext im Allgemeinen durch Algorithmen beschrieben. Die Erfahrungen, die zur Anpassung eines Algorithmus herangezogen werden können, sind Datensätze aus der Problemdomäne des betrachteten Algorithmus. Cherkassky und Mulier [CM98] definieren daher maschinelles Lernen wie folgt:

„Maschinelles Lernen bezeichnet den Prozess zur Verbesserung eines Algorithmus auf der Basis seiner Eingabedaten.“

Weiterhin unterteilen sie maschinelle Lernansätze in drei Problembereiche: Klassifikationsprobleme, Regressionsprobleme und die Schätzung von Verteilungsdichten.

Ansätze zum Schätzen von Verteilungsdichten dienen zur Vorhersage von Ereignissen in einem kontinuierlichen Ereignisraum anhand der Verteilung bisher beobachteter Ereignisse.

**Regression**

(Parametrische) Regression bezeichnet den Prozess zur Abschätzung der Parameter einer qualitativ bekannten Funktion auf der Basis rauschbehafteter Trainingsdaten. In Regressionsproblemen kann die Ausgabe des Systems beschrieben werden als Vektor $y$ von Zufallszwerten, der interpretiert werden kann als die Summe einer deterministischen Funktion $f$ mit unbekanntem Parametervektor $\omega_0$ und einem zufälligen, mittelwertfreien Fehler:

$$y = f(x, \omega_0) + \varepsilon \quad (3.1)$$

Lernen eines Regressionsproblem entspricht somit dem Finden von Parametern $\omega^*$, für die die Abweichung zwischen $f(x, \omega^*)$ und $y$ für die gegebenen Trainingsdaten der Form $(x, y)$ minimal ist. Da die Menge der Trainingsdaten nicht notwendigerweise alle Eigenschaften der Systemfunktion erkennen läßt, muß dabei auch bei einem optimalen Lernerfolg nicht unbedingt $\omega^* = \omega_0$ gelten. Lernmethoden für die Lösung des Regressionsproblems werden in Abschnitt 3.3 und Abschnitt 3.4 behandelt.

**Supervised and Unsupervised Learning**


**System**


Lernmuster


Trainingsdaten und Testdaten


Lernmaschine

Die Aufgabe der Lernmaschine ist die Nachbildung der unbekannten Systemfunktion zu einem gegebenen System. Hierzu steht der Lernmaschine im Allgemeinen eine Menge von Funktionen zur Verfügung, die aufgrund von a priori-Wissen über die zu bewältigende Lernaufgabe ausgewählt worden ist. Das Problem, das die Lernmaschine zu lösen hat, ist damit die Auswahl der Funktion aus der von ihr unterstützten Funktionsmenge, die die tatsächliche Systemreaktion, gegeben durch eine endliche Menge von Trainingsdaten, am besten approximiert.

Adaption

Als Adaption oder Training wird die Anpassung der Systemabschätzung aufgrund neuer Trainingsdaten bezeichnet. Insbesondere bezieht sich dies auf die Anpassung der Parameter $\omega$ in einem Regressionsproblem für einen neuen Satz von Trainingsdaten.

Der Zusammenhang zwischen dem System, der Lernmaschine und den Trainingsdaten ist in Abbildung 3.1 für ein typisches Regressionsproblem dargestellt. Die Lernmaschine erzeugt für eine Eingabe $x$ aus den Trainingsdaten mit den Parametern $\omega$ eine Ausgabe $\hat{y}$. Durch den Vergleich mit der tatsächlich vom System produzierten Ausgabe $y$ kann $\omega$ nun für $(x,y)$ angepasst werden, um die Systemfunktion $f(x,\omega_0)+\epsilon$ besser zu approximieren.

Abbildung 3.1: Lernen eines Regressionsproblems
3.2 Adaption von Fuzzy-Regeln und unscharfen Petrinetzen

Die Lernmaschine läßt sich hierbei, wie in Abbildung 3.2 zu sehen ist, in der Regel noch weiter unterteilen in die Systemfunktion, die das Ein-Ausgabe-Verhalten der Lernmaschine beschreibt, und den Lernalgorithmus, der anhand der Ist- und Sollwerte die Systemfunktion anpasst, so daß sie das tatsächliche System besser approximiert.

Abbildung 3.2: Die Lernmaschine im Detail

3.2 Adaption von Fuzzy-Regeln und unscharfen Petrinetzen

Mit Hilfe der soeben definierten Begriffe und der Beschreibung der Fuzzy-Bewertung aus Kapitel 2 kann nun die in der Einführung beschriebene Lernaufgabe eingeordnet und präzisiert werden.


Zur Lösung eines solchen Problems existieren verschiedene Ansätze, von denen einige im nachfolgenden Abschnitt diskutiert werden.

### 3.3 Regression und Gradient Descendant Techniken

Es existiert eine Vielzahl unterschiedlicher Lösungsansätze für das in Abschnitt 3.2 vorgestellte Regressionsproblem. Eine Arbeit, die eine zu der vorliegenden Arbeit fast identische Problemstellung behandelt, stammt aus dem Bereich des Datenbank-Reengineering. Im Kontext des in Kapitel 2 schon erwähnten Varlet-Projektes zur Rekonstruktion relationaler Datenbankschemata beschreibt Christoph David Strebin in seiner Diplomarbeit „Adaption unsicheren Wissens auf Basis konnektionistischer Methoden“ [Str99] einen Ansatz zur Adaption von Vertrauens- und Schwellwerten in Generic Fuzzy Reasoning Nets, aus denen die hier verwendeten Pattern Dependency Nets entstanden sind.


Das Regressionsproblem ist in Abschnitt 3.1 beschrieben worden als Optimierungsproblem zur Minimierung der Abweichung einer Funktion \( f(x, \omega) \) zu den tatsächlichen Werten \( y = f(x, \omega_0) + \epsilon \).

Für die Herleitung eines möglichst optimalen Parametersatzes \( \omega^* \) aufgrund der Lernmuster \((x,y)\) wird dabei in der Regel nicht die Funktion \( f \) betrachtet, sondern eine Fehlerfunktion \( e \), die die Abweichung zwischen \( y \) und \( f \) für alle Lernmuster in Abhängigkeit von dem Parametervektor \( \omega \) ausdrückt. Die am häufigsten betrachtete Fehlerfunktion ist der mittlere quadratische Fehler (Mean Square Error, MSE), der definiert ist als Summe

\[
\varepsilon(\omega) = \frac{1}{|TD|} \sum_{(x,y) \in TD} (y - f(x, \omega))^2
\]  

(3.2)

über die Menge TD der Trainingsdaten. Die Optimierung von \( f \) bezüglich \( \omega \) kann damit zurückgeführt werden auf die Minimierung der Fehlerfunktion \( \varepsilon(\omega) \) [Gal93].

Die Fehlerfunktion \( \varepsilon(\omega) \) ist in der Regel nichtlinear und kann daher meist nicht oder nur sehr schwer mit analytischen Methoden minimiert werden. Daher müssen numerische Approximationsverfahren angewendet werden. Gradient Descendent Algorithmen gehören hierbei zu den wichtigsten und meistangewendeten Verfahren [Wer94, And97]. Bekannte Algorithmen dieser Familie sind etwa Newton-Raphson, Polak-Ribiere oder der in Abbildung 3.4 vorgestellte Backpropagation-Algorithmus [And97]. Die Gemeinsamkeit dieser Verfahren ist die iterative Appro-
3.4 Neuronale Netze und Backpropagation

...ximation des gesuchten Minimums mit Hilfe des Gradienten. Der Gradient der Fehlerfunktion ist gegeben durch

\[ \nabla e(\omega) = \frac{\partial}{\partial \omega} e(\omega) \]  

(3.3)

und beschreibt einen Vektor, der in Richtung des steilsten Anstiegs von \( e \) zeigt. Das heißt durch Anpassung von \( \omega \) in Richtung von \( \nabla e(\omega) \) würde der Fehler am stärksten wachsen. Umgekehrt verringert der entgegengesetzte Vektor \( -\nabla e(\omega) \) für hinreichend kleine Anpassungsschritte \( \Delta \omega \) den Fehler in der Umgebung von \( \omega \). Dies wird in den Gradientenverfahren ausgenutzt, um sich iterativ an ein Minimum anzunähern. Ausgehend von einem beliebig gewählten Startwert \( \omega_1 \) wird der Parametervektor iterativ angepaßt:

\[ \omega_{i+1} = \omega_i - \rho \nabla e(\omega_i) \]  

(3.4)

Der Parameter \( \rho \) gibt dabei die Schrittweite der Anpassung an. Je nach verwendetem Algorithmus ist er fest gewählt oder wird bei jeder Iteration dynamisch errechnet. Die Iteration wird solange wiederholt, bis die Differenz aufeinanderfolgender Werte hinreichend klein ist. Dann hat \( e(\omega) \) (näherungsweise) ein Minimum angenommen.

3.4 Neuronale Netze und Backpropagation


3.4.1 Das Fuzzy Neural Net

Grundlagen neuronaler Netze

Ein neuronales Netz besteht aus einer Menge von Berechnungseinheiten, die als Neuronen bezeichnet werden. Sie sind durch gerichtete Kanten miteinander verbunden, denen Gewichte...
3.4 Neuronale Netze und Backpropagation


Die Berechnung der Aktivierung $a_i$ eines Neurons $u_i$ erfolgt in zwei Schritten [Gal93]. In einem ersten Schritt wird durch die Netzeingabefunktion $\text{NET}_i:([0, 1] \times \mathbb{R})^{n+1} \rightarrow \mathbb{R}$ aus den einge- henden Aktivierungen und Gewichten und dem Bias des Neurons die Netzeingabe $\text{net}_i$ bestimmt, wobei $n$ die Anzahl eingehender Kanten von $u_i$ ist. Aus $\text{net}_i$ wird dann durch die Aktivierungsfunktion $f_i: \mathbb{R} \rightarrow [0, 1]$ die Aktivierung $a_i$ des Neurons berechnet. In herkömmlichen neuronalen Netzen ist die Netzeingabefunktion zumeist für alle Neurone definiert als gewichtete Summe über die eingehenden Kanten von Neuronen $u_j$ mit

$$\text{net}_i = \sum_{j=0}^{n} w_{i,j} a_j$$

(3.5)

Die Aktivierungsfunktion $f_i$ wird auch häufig als Komprimierungsfunktion (squashing function)

3.4 Neuronale Netze und Backpropagation

bezeichnet, da sie die Ausgabe der Netzeingabefunktion auf das Intervall \([0,1]\) komprimiert. Der Aufbau eines Neurons ist in Abbildung 3.3 noch einmal schematisch dargestellt [Gal93].

Abbildung 3.3: Ein Neuron

Aufgabe des in Abschnitt 3.4.3 beschriebenen Backpropagation-Algorithmus ist es, die Kanten-

Struktur des Fuzzy Neural Nets

Damit das verwendete neuronale Netz die korrekten Vertrauens- und Schwellwerte lernen kann, muß es die Berechnungsvorschriften der Fuzzy Petrinetze nachbilden, aus denen die Lernmuster stammen. Hierbei ergibt sich das Problem, daß sich die Fuzzy Petrinetze aus unterschiedlichen Analyseprozessen voneinander unterscheiden, selbst wenn sie auf demselben PDN basieren. Das neuronale Netz muß indes für alle Lernmuster strukturell unverändert bleiben. Das in [Str99] vorgestellte Fuzzy Neural Net (FNN) orientiert sich in seiner Struktur daher nicht an den Fuzzy Petrinetzen, sondern an dem konstanten PDN. Aufbau und Verarbeitungsfunktionen des FNN sind dabei so gewählt, daß das FNN die Berechnungsvorschrift der Fuzzy Petrinetze für die verwendeten Lernmuster möglichst gut approximiert. Bei diesem Ansatz kann zu jedem Pattern des PDN nur eine Instanz zugleich im FNN repräsentiert werden, während im FPN typischerweise viele Stellen zu Instanzen desselben Patterns zu finden sind. Die Trainingsdaten für das FNN entstehen daher durch Aufteilung der Patterninstanzen eines FPNs auf mehrere Lernmuster, wie in Abschnitt 3.4.2 noch beschrieben wird.

Abbildung 3.4: Implikation im FNN

Implikation, indem er das Berechnungsergebnis der Minimumsfunktion auf 0 senkt. Ein Wert von 1 aktiviert die Implikation.


Ein Problem bei der Generierung des FNN stellen Kreise im PDN dar. Sowohl Kreise, die nur durch Implikationskanten erzeugt werden, als auch solche, die unter Berücksichtigung von Vererbungskanten entstehen, induzieren Zyklen im FNN. Hierdurch wird die Forderung nach einem Feed-Forward-Netz für den Backpropagation-Algorithmus verletzt. In [Str99] werden solche Kreise durch eine Umformung des GFRN aufgelöst, die auf einer Umkehrung der Implikationsrichtung basiert. Diese Umkehrung ist dort möglich, da die Implikationsbedingungen des GFRN aus quantorenanlogischen Ausdrücken bestehen, die durch Äquivalenzumformungen geeignet umgestellt werden können. Da in den hier verwendeten PDNs stattdessen nicht umformbare Graphtransformationen benutzt werden, ist dieses Verfahren nicht übertragbar. Da das Lernen mit neuronalen Netzen hier lediglich als Vergleichsansatz dient, ist dieses
Problem im Rahmen dieser Arbeit jedoch nicht weiter betrachtet worden. Stattdessen wird für die weitere Betrachtung der neuronalen Netze von einem kreisfreien PDN als Grundlage ausgegangen.

**Die Verarbeitungsfunktionen des Fuzzy Neural Net**

Um die Funktion eines Fuzzy Petrinetzes nachbilden zu können, müssen die Verarbeitungsfunktionen der Neuronen die Minimums- und die Schwellwertfunktion realisieren. Daneben wird, wie erläutert, noch die Maximumsfunktion für die Disjunktion unterschiedlicher Implikationskonfigurationen benötigt. Da für den Gradientenabstieg des Backpropagation-Algorithmus partiell differenzierbare Funktionen benötigt werden, werden nach [Str99] für das Fuzzy Neural Net die Funktionen des FPN durch stetige Funktionen approximiert. Für das Minimum und das Maximum sind dies die t-Norm $T$ und die dazu duale t-Conorm $\perp$ mit

\[
T(a, b) = 1 - [(1 - a)^p + (1 - b)^p]^{1/p} \tag{3.6}
\]

\[
\perp(a, b) = [a^p + b^p - a^p b^p]^{1/p} \tag{3.7}
\]

und für den Schwellwert $t$ die Sigmoid-Funktion $S$ mit

\[
S(a, t) = \frac{a}{1 + e^{-p(a-t)}} \tag{3.8}
\]


**3.4.2 Konstruktion der Lernmuster**


Durch die Anlehnung des neuronalen Netzes an die Struktur des PDN ergibt sich das Problem, daß das FNN nur jeweils eine Instanz jedes Patterns zugleich repräsentieren kann. Das FPN, das wie in Abschnitt 3.2 beschrieben die Systemeingabe repräsentiert, enthält dagegen typischerweise mehrere Instanzen desselben Patterns. Um das Fuzzy Neural Net als Lernmaschine für das Fuzzy-Bewertungssystem einsetzen zu können, ist es daher erforderlich, die Informationen des FPN auf geeignete Lernmuster für das FNN aufzuteilen, so daß die Berechnungsergebnisse des FNN weiterhin näherungsweise mit denen des FPN übereinstimmen. Die möglichen Abhängigkeiten zwischen Stellen des FPN sind im FNN statiisch realisiert. Bei der Abbildung des FPN auf das FNN besteht die Aufgabe daher darin, das FNN mit Hilfe der Aktivierungsneuronen entsprechend der im FPN vorgefundenen Situation zu konfigurieren. Die entscheidenden Eingabe-
3.4 Neuronale Netze und Backpropagation

werte für ein Lernmuster bestehen daher aus entsprechenden Werten für die Aktivierungsneuronen. Daneben gehören auch die Werte zu Axiomen des PDN zur Eingabemenge. Sie sind jedoch fest mit 1.0 festgelegt und unabhängig von dem Lernmuster.

In [Str99] wird ein Verfahren zur Aufteilung der Patterninstanzen und Abhängigkeiten eines FPN auf mehrere Lernmuster beschrieben. Hierzu werden zunächst die Patterninstanzen ausgewählt, für die das Netz trainiert werden soll. Dies sind zunächst alle Instanzen, für die gesicherte Daten in Form von benutzerdefinierten Fuzzy-Werten vorliegen. Desweiteren werden die übrigen Instanzen aller Pattern verwendet, zu denen mindestens eine Instanz mit benutzerdefinierten Werten existiert, da sie gegebenenfalls Gegenbeispiele zu den benutzerdefinierten Werten bilden, die im Lernprozess zu berücksichtigen sind.


Aus den gefundenen Bindungen ergeben sich unmittelbar die Lernmuster für das neuronale Netz. Die Ausgabewerte für ein Lernmuster sind dabei durch die Fuzzy-Werte aller zu Beginn ausgewählten Ausgabe-Patterninstanzen gegeben. Die Werte für die Aktivierungsneuronen, die die Eingabe des Lernmusters bilden, werden durch die gefundenen Bindungen festgelegt. Für jede gebundene Patterninstanz wird hierzu die Transition aus dem Vorbereich der Stelle im FPN betrachtet. Das Aktivierungsneuron für die zugehörige Implikation wird aktiviert, indem es im Lernmuster einen Eingabewert von 1.0 erhält. Entsprechend wird auch das Aktivierungsneuron
3.4 Neuronale Netze und Backpropagation

für den im FPN vorgefundenen Vorbereich der Transition mit einem Eingabewert von 1.0 aktiviert. Alle Aktivierungsneuronen, die nicht durch gebundene Patterninstanzen aktiviert werden, erhalten einen Eingabewert von 0.

3.4.3 Der Backpropagation-Algorithmus

Der Backpropagation-Algorithmus gehört zur Familie der in Abschnitt 3.3 beschriebenen Gradient Descendant Algorithmen. Er adaptiert die Kantengewichte und Bias-Werte. Sein Name stammt von dem besonders effizienten Verfahren, mit dem der Gradient der Fehlerfunktion berechnet wird. Bei diesem Verfahren wird die Gewichtsanpassung schrittweise für alle Gewichte des neuronalen Netzes berechnet, wobei das Netz beginnend mit den Ausgabeneuronen rückwärts zu den Eingabeneuronen durchlaufen wird. Dabei können große Teile der Gradientenberechnung für ein Gewicht auf bereits vorher berechnete Werte zurückgeführt werden. Diese wiederverwertbaren Teile der Gradientenberechnung werden für jedes Neuron \( u_i \) als Fehlersignal \( \delta_i \) berechnet mit

\[
\delta_i = -\frac{\partial e(w)}{\partial \text{net}_i} \quad (3.9)
\]

Für Ausgabeneuronen ist \( \delta_i \) direkt gegeben durch

\[
\delta_i = (y_i - a_i) \cdot f'(\text{net}_i) \quad (3.10)
\]

Für innere Neuronen kann \( \delta_i \) mit Hilfe der bereits berechneten Werte \( \delta_j \) der Nachfolger-Neuronen bestimmt werden. Für ein Neuron \( u_i \) mit \( n \) Nachfolger-Neuronen \( u_j \) gilt:

\[
\delta_i = \sum_{j=1}^{n} \frac{\partial \text{NET}}{\partial a_i} \delta_j \cdot f'(\text{net}_i) \quad (3.11)
\]

Abbildung 3.6: Fuzzy Petrinetz nach der Auswertung

Bindungen:
\{a,b1,c1,d\}
\{a,b1,c2,d\}
\{a,b2,c1,e\}
\{a,b2,c2,e\}
3.5 Probleme von Gradient Descendant Verfahren

Die Gewichtskorrektur nach Gleichung (3.4) kann damit wegen (3.9) ausgedrückt werden als

\[ w_{j,i}^{(k+1)} = w_{j,i}^{(k)} + \rho \delta_i f'(net_i) \cdot \frac{\partial NET_i}{\partial w_{j,i}^{(k)}} \]

(3.12)

Es muß also nur die partielle Ableitung für die Netzeingabefunktion für das anzupassende Gewicht neu berechnet werden, wodurch die Berechnung der Gewichtskorrektur besonders effizient wird.


```
repeat
    foreach Sample s do
        evaluate FNN for s.input
        compare FNN.output with s.output
        backpropagate error:
            calculate \( d_i \) for all Neurons \( u_i \) with (3.11)
            update weights \( w \) with (3.12)
        until weight correction \( \Delta w \approx 0 \)
```

Abbildung 3.7: Backpropagation als Pseudo-Code

3.5 Probleme von Gradient Descendant Verfahren

Bei der Anwendung von Gradient Descendant Verfahren zum Lernen von Regressionsproblemen ergeben sich zahlreiche Schwierigkeiten, die sie insbesondere für das hier betrachtete Adoptionsproblem als nicht geeignet erscheinen lassen.

Differenzierbare Fehlerfunktion

Eine wichtige Voraussetzung für die Anwendung von Gradient Descendant Verfahren ist die Differenzierbarkeit der Fehlerfunktion. Für das hier betrachtete Problem stellt dies ein Problem dar, da die im Fuzzy Petrinetz verwendeten Minimums- und Schwellwertoperationen nicht differenzierbar sind. Die Abbildung auf differenzierbare Funktionen hat unweigerlich einen Näherungsfehler zu Folge. Da eine gute Annäherung an die nicht-stetigen Funktionen gleichzeitig
3.5 Probleme von Gradient Descendant Verfahren

Eine Verschlechterung der Konvergenzgeschwindigkeit bewirkt (siehe [Str99]), ist hier zwischen der Konvergenzgeschwindigkeit und der Qualität der Ergebnisse abzuwägen.

**Lokale Minima**


**Feste Systemfunktion**

Ein besonderes Problem im Kontext der in dieser Arbeit zu lösenden Lernaufgabe stellt die approximierte Systemfunktion \( f \) dar. Alle herkömmlichen Ansätze zur Lösung des Regressionsproblems gehen davon aus, daß die zu approximierende Funktion für alle Lernmuster dieselbe ist und lediglich eine Veränderung der Parameter stattfindet. Bei der Fuzzy-Evaluation der Design Pattern wird diese Funktion durch das Fuzzy Petrinetz realisiert. Da das Petrinetz von den erkannten Pattern abhängt, ist die Systemfunktion aber nach jedem Erkennungsprozeß eine andere und nicht für alle Lernmuster konstant. Lediglich die im Fuzzy Petrinetz verwendeten Parameter sind für alle Lernmuster dieselben. Um dennoch ein herkömmliches Regressionsverfahren wie Backpropagation auf diese Lernaufgabe anwenden zu können, muß also die vom Fuzzy Petrinetz vorgegebene Systemfunktion in der Lernmaschine durch eine gleichbleibende Funktion ausgedrückt werden. Dies geschieht wie beschrieben durch die Abbildung der PDN-Struktur auf das FNN. Hierdurch ergibt sich jedoch das Problem, daß die Lernmuster, die ursprünglich durch die vom Inferenzprozeß erzeugten FPNs gegeben sind, auf diese Struktur abgebildet werden müssen wie in Abschnitt 3.4.2 beschrieben. Dies führt zum einen zu weiteren Approximationsfehlern im Lernprozeß. Zum anderen hat die Abbildung zur Folge, daß zur Berücksichtigung aller entscheidenden Kombinationen von Ein- und Ausgabestellen des FPN bei der Abbildung aus jedem FPN eine große Anzahl von Lernmustern erzeugt wird. Im ungünstigsten Fall kann die Zahl der Lernmuster exponentiell mit der Tiefe des FPN wachsen.

**Konvergenzgeschwindigkeit und Divergenz**

Ein weiteres Problem stellt die Konvergenz von Gradient Descendant Ansätzen dar.

manuelle Konstruktion des neuronalen Netzes voraus, die bei dem hier betrachteten Ansatz nicht gegeben ist.

Im praktischen Einsatz zeigt sich, daß durch die aus der Abbildung auf das FNN resultierende extrem hohe Anzahl an zu verarbeitenden Lernmustern zusammen mit Ill-Conditioning Problemen der Backpropagation-Ansatz bereits für FPNs mit mehr als etwa hundert Stellen nicht mehr sinnvoll einsetzbar ist. Daher wird im folgenden Kapitel ein Lernansatz vorgestellt, der das gegebene Regressionsproblem mit einem nicht-iterativen Algorithmus auf Basis statistischer Überlegungen effizient löst.
KAPITEL 4

Der Lösungsansatz:
Statistische Adaption im FPN

Gegenstand dieses Kapitels ist der Entwurf eines heuristischen Adoptionsalgorithmus zur Anpassung der Vertrauens- und Schwellwerte eines Musterkataloges auf der Basis der vom Benutzer vorgenommenen Korrekturen. Hierzu werden in Abschnitt 4.1 zunächst die Anforderungen aus Kapitel 1 und die in Kapitel 3 identifizierten Probleme herkömmlicher Ansätze analysiert. Auf dieser Basis wird dann in Abschnitt 4.2 ein Lösungsansatz entwickelt, der in den folgenden Abschnitten weiter ausgeführt wird. Abschnitt 4.3 behandelt die Aufbereitung der Daten eines einzelnen Lernmusters, bevor diese in Abschnitt 4.4 in die Adaption der Vertrauenswerte eingehen. Dasselbe Verfahren wird anschließend auf die Adaption der Schwellwerte angewandt.

4.1 Entwurfskriterien


Ein weiteres wichtiges Kriterium ist die Realisierung der Systemfunktion nach Abbildung 3.2. Ein geeignetes Berechnungsnetz ist bereits in Form des Fuzzy Petrinetzes vorhanden. Es ist daher wünschenswert, dieses Netz direkt im Lernalgorithmus zu verwenden, anstatt wie im Fall...
4.2 Statistische Adaption


Besonderes Augenmerk ist hierbei auf die Tatsache zu richten, daß jeder der anzupassenden Parameter in den Fuzzy Petrinetzen mehrfach vorkommen kann. Dies, zusammen mit der hohen Knotenzahl der FPNs, stellt für herkömmliche Regressionsansätze, wie in Kapitel 3 beschrieben, ein großes Problem dar. Andererseits bedeutet es aber auch, daß für jeden dieser Parameter eine große Zahl von Korrekturdaten, und damit wichtigen Lerninformationen, zur Verfügung steht. Der im Folgenden beschriebene Ansatz soll daher dieses hohe Datenaufkommen bestmöglich ausnutzen, und so diesen Nachteil herkömmlicher Ansätze in einen Vorteil verwandeln.

Der auf diesen Kriterien basierende Ansatz wird nun im folgenden Abschnitt zunächst konzeptionell hergeleitet, bevor im weiteren Verlauf die detaillierte Ausarbeitung erfolgt.

4.2 Statistische Adaption

Der zu entwerfende Adaptionsalgorithmus hat zwei grundlegende Probleme zu lösen, das der Uneindeutigkeit und das der Widersprüchlichkeit.

Wenn der Benutzer den Fuzzy-Wert einer Patterninstanz ändert, wird diese Änderung in Form eines benutzerdefinierten Fuzzy Belief Markings an die zugehörige Stelle im FPN weitergegeben. Um diesen Wert auch für den vom FPN errechneten Belief zu erhalten, müssen gemäß Abschnitt 2.3 die Vertrauenswerte im FPN entsprechend angepasst werden. Hierbei ergibt sich das Problem der Uneindeutigkeit. Die erforderlichen Anpassungen lassen sich nicht eindeutig aus dem Korrekturwert und dem FPN ableiten. Vielmehr gibt es zahlreiche unterschiedliche Realisierungsmöglichkeiten für jede Benutzerkorrektur.


Widersprüche ergeben sich, wenn für eine anzupassende Größe mehrere unterschiedliche Korrekturwerte vorliegen. Dies ist der Fall, wenn die Korrekturen verschiedener Benutzereingaben gemäß Abbildung 4.1c durch eine gemeinsame Vorbereichsstelle realisiert werden sollen. Derartige Widersprüchlichkeiten können häufig aufgelöst werden, indem andere Realisierungs-
4.2 Statistische Adaption

Der Lösungsansatz: Statistische Adaption im FPN

möglichkeiten für die Korrekturen gewählt werden. Ist dies nicht möglich, so ist der resultierende Wert so zu wählen, daß der Fehler hinsichtlich aller beteiligten Benutzereingaben minimal ist.


Stattdessen wird im Folgenden ein heuristisches Verfahren vorgestellt, das auf der Idee der gleichzeitigen Verfolgung der möglichen Realisierungen basiert, jedoch die problematische Einzelbetrachtung der verschiedenen Realisierungsmöglichkeiten durch eine einfache und effiziente statistische Auswertung ersetzt.

Hierzu werden für jede Benutzerkorrektur die erforderlichen Zielwerte an alle Stellen und Transitionen im FPN weitergegeben, an denen eine Realisierung der erforderlichen Anpassung möglich ist. Betrachtet man die so aufbereiteten Änderungsdaten mehrerer FPNs an den zugehörigen Implikationen des zugrundeliegenden PDNs, so steht zu erwarten, daß sich an den für eine Anpassung am besten geeigneten Implikationen ein entsprechender Zielwert aus diesen Daten herauskristallisiert. Andererseits werden nachteilige Änderungen durch entsprechende Gegenbeispiele in den Änderungsdaten unterdrückt.

Das Optimierungsproblem wird somit auf mehrere lokale Probleme reduziert, die anschließend einfacher zu lösen sind. Dabei wird ausgenutzt, daß die anzupassenden Werte des PDN in den FPNs in der Regel mehrfach vorkommen, so daß eine entsprechend große und damit zuverlässige statistische Basis für die Auswertung zur Verfügung steht.

Das bereits geschilderte Problem der Uneindeutigkeit betrifft in besonderem Maße die hier angestrebte Realisierung eines Lernansatzes als One-Shot-Lösung, da eine einmal getroffene

Abbildung 4.1: Realisierungsmöglichkeiten

a) Benutzereingabe  b) lokale Korrektur  c) Teilpattern  d) mehrere Knoten
4.3 Adaption der Vertrauenswerte

Entscheidung nicht auf der Basis späterer Erkenntnisse revidiert werden kann. In diesem Zusammenhang stellt die Kopplung der Vertrauenswerte mit den Schwellwerten bei der Berechnung der Fuzzy Beliefs nach Definition 2.7 ein besonderes Problem dar. Denn die Ablehnung oder nachträgliche Akzeptierung einer Musterinstanz kann sowohl durch Anpassung des Vertrauenswertes als auch durch Änderung des Schwellwerts erreicht werden. Zudem ist es für die Aufbereitung der Korrekturdaten von Nachteil, daß Fuzzy Belief-Werte durch vorangegangene Schwellwertbildungen maskiert werden, da so die Auswirkungen von Änderungen schwer zu beurteilen sind. Daher wird für die folgenden Betrachtungen eine Vereinfachung vorgenommen, die die getrennte Behandlung von Vertrauens- und Schwellwerten erlaubt. Hierzu wird die Berechnung des Fuzzy Truth Tokens aus Definition 2.7 geändert zu

\[
\text{ftt}(t) = \min \{ \text{cf}(t) \cup \{ \text{fbm}(s) \mid s \in \text{pre}(t) \}\}
\]  

(4.1)

Zusätzlich wird ein Fuzzy Threshold Token eingeführt, das zusammen mit dem FTT berechnet wird und die Schwellwertbildung realisiert. Die Stellen des FPN halten den durch das FTT gegebenen Belief-Wert und die Schwellwertinformation des FTH getrennt. Der Fuzzy-Wert einer Musterinstanz ist damit gegeben durch den (nicht schwellwertbehafteten) Fuzzy Belief der zugehörigen Stelle falls das FTH der Stelle den Wert 1 hat und wird ansonsten zu 0. Definition 2.6 ist entsprechend anzupassen.


Im Folgenden wird deshalb der Lernansatz zunächst am Beispiel der Vertrauenswerte entwickelt, bevor das Verfahren dann in Abschnitt 4.6 auf die Schwellwerte angewendet wird. Zunächst wird im folgenden Abschnitt die Aufbereitung der Korrekturdaten eines einzelnen Lernmusters, gegeben durch ein FPN und eine Menge von Benutzerkorrekturen, betrachtet. Anschließend wird die Anpassung durch Zusammenführung und statistische Auswertung der aufbereiteten Daten erläutert.

4.3 Adaption der Vertrauenswerte

Durch die getrennte Auswertung von Vertrauens- und Schwellwerten bei der Berechnung der Fuzzy Truth Tokens können die Schwellwerte in diesem Schritt zunächst vernachlässigt werden.
Daher kann die Berechnung des Fuzzy Beliefs aus Definition 2.7 in der vereinfachten Form aus Gleichung (4.1) erfolgen. Der Vertrauenswert einer Transition wird damit bei der Berechnung des FTT nicht mehr anders behandelt, als die Werte aus ihrem Vorbereich. Für die Formulierung des Adaptionalgorithmmus bietet es sich daher zur Vereinfachung an, den Schwellwert ebenfalls genauso zu behandeln, wie die Fuzzy-Werte der Stellen im FPN. Dazu werden zusätzliche Stellen in das FPN eingefügt, die die Schwellwerte repräsentieren und als Bias-Stellen bezeichnet werden. An dem Beispiel in Abbildung 4.2 ist zu sehen, daß die neu hinzugefügte Bias-Stelle cf mit dem Schwellwert der Transition initialisiert ist und die Schwellwert-Angabe an der Transition dafür fehlt.

Für das so vereinfachte FPN lassen sich nun recht einfach zwei Bedingungen angeben, die eine Stelle s erfüllen muß, um einem vom Benutzer vorgegebenen Wert x anzunehmen. Diese Bedingungen folgen unmittelbar aus der Minimierungseigenschaft in Gleichung (4.1).

Damit s den Wert x annehmen kann, muß es mindestens eine Stelle im Vorbereich von s geben, die ebenfalls den Wert x hat. Darüber hinaus darf es keine Werte kleiner als x im Vorbereich von s geben. Es ergibt sich also:

\[ \exists s' \in \text{pre}_S(s) : \text{fbm}(s') = x \quad (4.3) \]

\[ \forall s' \in \text{pre}_S(s) : \text{fbm}(s') \geq x \quad (4.4) \]

Diese beiden Bedingungen können auch als Handlungsanweisungen für die Umsetzung der Benutzerkorrekturen aufgefaßt werden und bilden damit die Grundlage für die im Folgenden beschriebenen Adoptionsverfahren.

4.3 Adaption der Vertrauenswerte

4.3.1 Senken des Vertrauenswertes

Bei der Senkung des Vertrauenswertes ergibt sich, wie im Beispiel aus Abbildung 4.1 bereits erwähnt, das Problem, daß nicht eindeutig bestimmt werden kann, welche Werte zur Realisierung dieser Änderung angepasst werden müssen. Abbildung 4.3 zeigt nochmal einige resultierende Möglichkeiten in der Darstellung mit Bias-Stellen. Es ist zu sehen, daß die lokale Anpassung des Vertrauenswertes der Transition in Abbildung 4.3b keinen Sonderfall mehr gegenüber den Anpassungen in Abbildung 4.3c und d darstellt.


Durch die Propagation an alle Vorbereichsstellen werden jedoch viele Änderungsanforderungen an Stellen weitergegeben, auf die sie bei einer optimalen Lösung keinen Einfluß hätten. Bei der späteren Auswertung der Änderungsanforderungen beeinflussen diese falsch propagierten Werte das Endergebnis negativ. Dieser negative Einfluß kann mit Hilfe einer einfachen Heuristik stark reduziert werden. Dazu werden Änderungsanforderungen bevorzugt an solche Stellen weitergegeben, für die die erforderliche Anpassung minimal ist, aufgrund der Minimierungseigenschaft (4.1) also Stellen mit niedrigem Fuzzy Belief.

multipliziert, so daß Änderungsanforderungen an bevorzugte Stellen ein hohes Gewicht behalten, während Anforderungen an Stellen mit starker Abweichung von der Ursprungsstelle durch ein niedrigeres Gewicht penalisiert werden. Das Gewicht der Änderungsanforderungen drückt in diesem Kontext gewissermaßen die Unsicherheit darüber aus, ob eine Anforderung auf die jeweilige Stelle Einfluß haben sollte.

Diese Heuristik basiert auf der Annahme, daß mit hoher Wahrscheinlichkeit bei der initialen Festlegung der Vertrauenswerte während der Musterspezifikation nur ein verhältnismäßig geringer quantitativer Schätzfehler gemacht wird. Im Gegensatz dazu ist ein Fehler bei der qualitativen Einordnung der Musterspezifikationen zueinander, also der Einschätzung, ob eine Spezifikation qualitativ vertrauenswürdiger ist, als eine andere, deutlich unwahrscheinlicher. Diese Annahme deckt sich auch mit Erfahrungen aus dem praktischen Einsatz des Inferenzprozesses aus Kapitel 2. Die Idee der Bevorzugung der wahrscheinlichsten Lösung hat sich auch in vielen anderen Lernansätzen bewährt, insbesondere in Form sogenannter Maximum-Likelihood Schätzer [CM98], bei denen ein zu optimierender Wert so gewählt wird, daß die Wahrscheinlichkeit eines korrekten Ergebnisses hinsichtlich einer gegebenen Wahrscheinlichkeitsverteilung maximiert wird.

Die für die Gewichtung zu verwendende Gewichtsfunktion ist nicht eindeutig vorgegeben. Es gibt jedoch eine Reihe von Rahmenbedingungen, anhand derer eine geeignete Funktion ausgewählt werden kann. Die Gewichtsfunktion muß für die Differenz zwischen den Vertrauenswerten von Vor- und Nachbereichsstelle auf das Intervall [0,1] beschränkt sein und monoton fallen. Zudem ist es sinnvoll, eine Funktion zu wählen, die zu Beginn relativ flach verläuft, um kleine Abweichungen nicht zu überbewerten. Eine Funktion, die diese Kriterien erfüllt und auch häufig in Maximum-Likelihood-Ansätzen auftritt, ist die logistische Funktion

\[ \text{lf}: \mathbb{R} \rightarrow [0,1], \text{lf}(x) = \frac{1}{1 + e^{-B(x-M)}} \quad (4.5) \]

mit geeignetem Wachstumsfaktor B und Wendepunkt M. In praktischen Tests hat sich diese Funktion bewährt. Mit \( B=0.2 \) und \( M=25 \) ergibt sich die Gewichtsfunktion

\[ c:F \rightarrow [0,1], \ c((s, t)) = \text{lf}(\Delta \text{fbm}) = \frac{1}{1 + e^{0.2(\Delta \text{fbm}-25)}} \quad (4.6) \]

beziehungsweise

\[ c(s, s') = c((s, t)), t = \text{pre}(s') \in T \quad (4.7) \]

für eine Stellen-Transitions-Kante \((p, t)\) und

\[ \Delta \text{fbm} = \text{fbm}(s) - \text{fbm}(\text{post}(t)) \quad (4.8) \]

wobei zu beachten ist, daß \( \Delta \text{fbm} \) nach Gleichung (4.1) nie negativ wird. Abbildung 4.4a zeigt die Gewichtsfunktion für Differenzen \( \Delta \text{fbm} \) zwischen 0 und 100.

Mit diesen Gewichten ergibt sich damit für die zu propagierenden Änderungsanforderungen die folgende formale Gestalt.
4.3 Adaption der Vertrauenswerte

Definition 4.1: Änderungsanforderung

Eine Änderungsanforderung (engl. change request) an eine Stelle \( s \in S \) ist ein Tupel \((v|w) \in [0,100] \times \mathbb{R}^+\) aus einem Vertrauenswert \( v \) und einem zugehörigen Gewicht \( w \). Die Menge der Änderungsanforderungen \( cr = (v|w) \) zu einer gegebenen Stelle \( s \) ist vollständig beschrieben durch die Folge von Gewichten \( w_{sv} = w \).

Ein Beispiel für die Änderungspropagierung mit Gewichten ist in Abbildung 4.4b dargestellt. Die vom Benutzer durchgeführte Änderung des Fuzzy Beliefs von \( s1 \) von 70 auf 60 wird mit dem Gewicht von 1,0 initialisiert. Die Kanten von den Vorbereichsstellen von \( s1 \) zur Transition sind mit Gewichten \(^\text{1}\) versehen, die bei der Propagation des Tupels \((60|1,0)\) mit dem Gewicht der Änderung multipliziert werden. Es ist zu sehen, daß Änderungsanforderungen an die Stellen \( s2 \) und \( cf \), deren Werte näher an dem ursprünglichen Wert von \( s1 \) liegen, gegenüber der an \( s3 \) höher gewichtet werden. Es ist auch zu sehen, daß kleine Abweichungen von dem ursprünglichen Wert von \( s1 \) nur wenig „bestraft“ werden (cf), während die Gewichtsabnahme für größere Abweichungen zunimmt (s3).

![Abbildung 4.4: Gewichtete Propagation der Vertrauenswerte](image)


Abbildung 4.5 zeigt das Beispiel aus Abbildung 4.4b in einem größeren Kontext und verdeutlicht die rekursive Propagierung im FPN. Die Stelle \( s3 \) erhält hier zusätzlich zu der Änderungsanforderung von \( s1 \) mit Vertrauenswert 60 eine weitere Anforderung von \( s2 \) mit Vertrauenswert 50. Da sich die Vertrauenswerte der beiden Anforderungen unterscheiden, werden sie nicht

---

1. Zur Vereinfachung der Darstellung werden gerundete Werte verwendet
zusammengefaßt. Die Anforderungen werden von \(s_3\) an \(s_5\) weiterpropagiert. \(s_6\) erhält sie jedoch nicht, da für diese Stelle eine Benutzereingabe vorliegt. Es ist zu sehen, daß an \(s_5\) das Gewicht der Änderungsanforderung von \(s_2\) durch die Multiplikation mit dem Kantengewicht weiter abgenommen hat. Die Anforderung von \(s_1\) hat denselben Vertrauenswert, wie eine weitere von der Stelle \(s_4\) stammende. Diese beiden Anforderungen werden daher zusammengefaßt, das resultierende Gewicht entspricht der Summe der Einzelanforderungen nach Multiplikation mit den Kantengewichten:

\[
1,42 = (0,6 \cdot 0,95) + (1,0 \cdot 0,85) \quad (4.9)
\]

Nach dieser informalen Erläuterung wird der Propagierungsalgorithmus nun formal beschrieben. Dazu ist es zunächst erforderlich, den Teilgraphen des FPN zu definieren, an den die Änderungsanforderungen einer Benutzereingabe propagiert werden.

**Definition 4.2: Influenzbereich einer Änderungsanforderung**

Der Influenzbereich \(inf(s)\) einer Stelle \(s\) ist die Menge aller Stellen im Teil-FPN unterhalb von \(s\), ausgenommen solcher Teilgraphen, die von weiteren Stellen mit Benutzereingaben aufgespannt werden. Sei \(S_u\) die Menge der Stellen mit Benutzereingaben im FPN und \(S_u(s)\) die Teilmenge von \(S_u\) im Teilgraph unter \(s\)

\[
S_u = \{ s' \mid s' \in S \land \text{fbm}_u(s') \neq -1 \}, \quad (4.10)
\]

\[
S_u(s) = S_u \cap \text{pre}_S^*(s). \quad (4.11)
\]

Dann gilt für \(inf(s)\)

\[
inf: S \rightarrow P(S), \quad inf(s) = \text{pre}_S^*(s) \setminus \text{pre}_S^*(S_u(s)) \quad (4.12)
\]

Mit Hilfe des Influenzbereiches kann nun der Propagierungsalgorithmus formuliert werden.

Abbildung 4.5: Propagierung der Änderungsanforderungen
4.3 Adaption der Vertrauenswerte

Definition 4.3: Gewichtete Propagierung von Änderungsanforderungen

Seien $s_i, i \in [1, m]$ die Stellen des FPN, deren Fuzzy Belief vom Benutzer gesenkt worden ist. Ihre benutzerdefinierten Werte seien durch $v_i, i \in [1, m]$ gegeben. Die Änderungsanforderungen mit Wert $v$ an einem Knoten $s$ sind zum Zeitpunkt $t$ durch ihre Gewichte $w_{s,v,t}$ definiert.

Zum Zeitpunkt $t=0$ sind alle Gewichte mit 0 initialisiert. Für die Stellen $s_i$ mit Benutzereingaben gilt

$$w_{s_i,v,t} = \begin{cases} 1 & \text{für } t \geq i, \ v = v_i \\ 0 & \text{sonst} \end{cases} \quad (4.13)$$

Nach der Propagierung der Änderungsanforderungen der ersten $t$ Stellen gilt für jede Stelle $s$ aus dem Vorbereich der geänderten Stellen:

$$w_{s,v,t} = w_{s,v,t-1} + \sum_{s' \in \text{post}(s)} c(s,s') \cdot \Delta w_{s',v,t}, \quad s \in \bigcup_{i=1..t} \text{inf}(s_i) \quad (4.14)$$

$\Delta w_{s',v,t}$ ist die Änderung an der Stelle $s'$ durch die Propagierung der Änderungsanforderung von $s_t$, für $t>0$ gegeben durch

$$\Delta w_{s',v,t} = w_{s',v,t} - w_{s',v,t-1} \quad (4.15)$$

Aus (4.14) und (4.15) folgt, daß nach der Propagation aller $m$ Änderungsanforderungen gilt:

$$w_{s,v}^* = w_{s,v,m} = \sum_{s' \in \text{post}_S(s)} c(s,s') \cdot w_{s',v}^* \quad (4.16)$$

Außerdem folgt aus der Zyklenfreiheit des FPN (siehe Abschnitt 2.3.1), daß der Algorithmus immer in endlicher Zeit terminiert.

Die bisherigen Überlegungen zur Verarbeitung der Benutzereingaben haben sich im wesentlichen auf die Bedingung (4.3) konzentriert. Zusätzlich hierzu kann die Bedingung (4.4) herangezogen werden, um die Ergebnisse weiter abzusichern.

Für eine Benutzerkorrektur, die den Fuzzy Belief einer Stelle $s$ auf einen neuen Wert $x$ senkt, müssen nach Gleichung (4.4) alle Stellen im Vorbereich von $s$ einen Wert größer oder gleich $x$ annehmen. Erhält eine Stelle $s'$ aus dem Vorbereich von $s$ neben der aus der Anpassung von $s$ resultierenden Änderungsanforderung $cr=(x|w)$ eine weitere Änderungsanforderung $cr'=(x'|w')$, deren Wert $x'$ kleiner als $x$ ist, so steht diese Anforderung $cr'$ nach (4.4) im Widerspruch zu $cr$. Ist $x'$ hingegen größer als $x$, so ist $cr$ mit dieser Anforderung nach (4.4) konsistent und kann somit als Positivindikator für die Umsetzung von $cr'$ in $s'$ oder dessen Vorbereich interpretiert werden. Formt man Gleichung (4.4) wie folgt um

$$\forall s' \in \text{pre}_S(s): \text{fbm}(s') = x \lor \text{fbm}(s') > x \quad , \quad (4.17)$$

so ist zu erkennen, daß dies der ersten Regel entspricht, während die erste Regel bereits durch die Anpassung von $cr$ ausgedrückt wird.

Es liegt nahe, in einem solchen Fall das Gewicht der konsistenten Änderungsanforderung $cr'$ zu erhöhen. Hierzu wird entsprechend dem zweiten Teil von (4.17) zusammen mit jeder Ände-
4.3 Adaption der Vertrauenswerte

Die Lösungsansatz: Statistische Adaption im FPN

Der Anforderungsanforderung \( cr=(x|w) \) ein Endorsement-Gewicht \( e \) für das Intervall \([x, 100]\) propagiert, das bei der späteren Auswertung zur Verstärkung aller Änderungsanforderungen in \([x, 100]\) verwendet wird, die an denselben Knoten wie \( cr \) auftreten. Da die obere Intervallgrenze fest mit 100 vorgegeben ist, reicht es für die Propagierung, die untere Grenze \( x \) zu propagieren. Berücksichtigt man ferner, daß die beiden Teile von (4.17) zueinander komplementär sind, so ist es nahe liegend, auch den Wert von \( e \) als Komplement des Gewichts \( w \) der Änderungsanforderung anzunehmen:

\[
e = 1 - w \quad (4.18)
\]

Es ist zu beachten, daß die Propagierung der Endorsement-Gewichte zwar genauso erfolgt, wie für die Änderungsanforderungen beschrieben, jedoch mit sinkenden Gewichten für die Änderungsanforderungen der Wert der Endorsement-Gewichte gemäß (4.18) steigt.

Der Zusammenhang zwischen Änderungsanforderungen und Endorsement-Gewichten ist in Abbildung 4.6 beispielhaft zu sehen. Die Propagierung erfolgt genauso, wie in dem Szenario in Abbildung 4.5. Zusätzlich wird hier jedoch das Endorsement-Gewicht, dargestellt als dritter Wert des Tupels, mitpropagiert. Es ist zu erkennen, daß das Endorsement-Gewicht initial für Benutzereingaben mit 0 angesetzt ist (s1) und mit sinkendem Gewicht der Änderungsanforderung ansteigt (s3). Endorsement-Gewichte werden genauso wie die Gewichte der Änderungsanforderungen durch Addition zusammengefaßt, wenn die zugehörigen Vertrauenswerte übereinstimmen (s5). Bei der in Abschnitt 4.4 beschriebenen Auswertung verstärken die so propagierten Endorsement-Gewichte an jeder Stelle die Änderungsanforderungen, deren Vertrauenswert über dem der Endorsements liegt. Für die Stelle s5 würde etwa die Änderungsanforderung \( cr1 \) durch das mit \( cr2 \) propagierte Endorsement-Gewicht von 0,35 verstärkt werden.

Formal läßt sich dies wie folgt ausdrücken.
4.3 Adaption der Vertrauenswerte

Definition 4.4: Endorsement-Gewicht

Das Endorsement \( ew \) für ein Intervall \([v,100]\) an einer Stelle \( s \in S \) ist gegeben durch das Tupel \( ew = (v|e) \in [0,100] \times \mathbb{R}^+ \) aus einem Vertrauenswert \( v \) und einem zugehörigen Endorsement-Gewicht \( e \). Die Menge der Endorsements \( ew = (v|e) \) zu einer gegebenen Stelle \( s \) ist vollständig beschrieben durch die Folge von Endorsement-Gewichten \( e_{s,v} = e \). Analog zu Definition 4.3 gilt nach der Propagierung der \( t \)-ten Änderungsanforderung

\[
e_{s,v,t} = e_{s,v,t-1} + \sum_{s'} \Delta e_{s',v,t} + (1-c(s,s')) \cdot \Delta w_{s',v,t}
\]

mit

\[
\Delta e_{s',v,t} = e_{s',v,t} - e_{s',v,t-1}
\]

4.3.2 Erhöhen des Vertrauenswertes

Das Erhöhen der Fuzzy Beliefs durch den Benutzer wird weitestgehend analog zum Senken behandelt. Es sind wiederum zur Realisierung des benutzerdefinierten Wertes die Vertrauenswerte des FPNs geeignet anzupassen. Allerdings gibt es einige kleine Unterschiede, insbesondere in der Motivation der verwendeten Verfahren, auf die im Folgenden eingegangen werden soll.

Beim Senken der Fuzzy Beliefs ergibt sich nach Gleichung (4.3) das Problem, zu entscheiden, welche Stellen zur Umsetzung einer Benutzerkorrektur anzupassen sind. Dies ist in Abbildung 4.3 verdeutlicht. Betrachtet man im Vergleich dazu Abbildung 4.7b, so ist zu erkennen, daß durch Gleichung (4.4) die minimal anzupassenden Stellen direkt gegeben sind. Dies sind genau die Stellen mit einem Wert unter dem vom Benutzer vorgegebenen. Allerdings sind durch (4.4) die Werte nur nach unten beschränkt. Da in (4.3) nur gefordert ist, daß mindestens eine Stelle den Wert der Benutzereingabe annimmt, ist etwa auch das in Abbildung 4.7c dargestellte Szenario möglich.

\[
\Delta e_{s',v,t} = e_{s',v,t} - e_{s',v,t-1}
\]

Zur Behandlung dieser Uneindeutigkeit werden die Änderungen des Benutzers wiederum als gewichtete Änderungsanforderungen durch das FPN propagiert. Während die Gewichte beim
4.3 Adaption der Vertrauenswerte

Senken jedoch die Unsicherheit bezüglich der zu ändernden Stellen ausdrücken, beschreiben sie in diesem Fall die Unsicherheit über das Ausmaß der Erhöhung.

Eine Erhöhung über den vom Benutzer vorgegebenen Wert hinaus, wie sie Abbildung 4.7c für die Stellen s3 und cf zeigt, können allerdings nicht direkt aus dieser Benutzereingabe geschlossen werden. Stattdessen werden sie analog zum Vorgehen in Abschnitt 4.3.1 durch Endorsement-Gewichte unterstützt. Änderungsanforderungen aus Benutzereingaben mit höheren Vertrauenswerten werden hierdurch verstärkt.

Genauso kann für Vorbereichsstellen mit einem Wert über dem vom Benutzer vorgegebenen gemäß (4.4) aus der Benutzereingabe keine Indikation zum Senken dieser Stellen entnommen werden. Daher ist die für die Propagation von Erhöhungen zu verwendende Gewichtsfunktion gegenüber (4.7) leicht anzupassen, so daß Änderungsanforderungen an solche Stellen das Gewicht 0 erhalten:

\[ c^+(s, s', v) = \begin{cases} 
  c(s, s') & \text{falls } \text{fbm}(s) \leq v \\
  0 & \text{sonst} 
\end{cases} \]  

(4.21)

\( v \) ist hierbei der vom Benutzer vorgegebene Wert. Für den zweiten Fall ist zu beachten, daß gemäß (4.18) das zugehörige Endorsement-Gewicht zu 1 wird.

Für Stellen \( s_i, i \in [m + 1, n] \) mit vom Benutzer erhöhten Werten folgt analog zu Definition 4.3 nach der Propagation der Benutzereingaben an den Stellen \( s_{m+1} \) bis \( s_t \) für Stellen \( s \):

\[ w_{s,v,t} = w_{s,v,t-1} + \sum_{s' \in \text{post}(s)} c^+(s, s', v) \cdot \Delta w_{s',v,t}, s \in \bigcup_{i=1..t} \text{inf}(s_i) \]  

(4.22)

Nach der Propagierung aller Änderungsanforderungen haben damit die Erhöhungen einen Beitrag von

\[ w^+_{s,v} = w_{s,v,n} - w_{s,v,m} = \sum_{s' \in \text{post}_S(s)} c^+(s, s', v) \cdot w^+_{s',v} \]  

(4.23)

an den Gesamtgewichten \( w_{s,v,n} \). Somit lassen sich die Gesamtgewichte beschreiben durch

\[ w_{s,v} = w_{s,v,n} = w^-_{s,v} + w^+_{s,v} \]  

(4.24)

\[ w_{s,v} = \sum_{s' \in \text{post}_S(s)} (c(s,s') \cdot w^-_{s',v} + c^+(s, s', v) \cdot w^+_{s',v}) \]  

(4.25)

Die Endorsement-Gewichte werden mit der Gewichtsfunktion (4.21) wie in Definition 4.4 beschrieben propagiert.
4.4 Statistische Auswertung

4.3.3 Bestätigen des Vertrauenswertes


Hierzu werden an den bestätigten Stellen spezielle „Änderungsanforderungen“ erzeugt und an deren Vorbereiche propagiert, die das Beibehalten der bisherigen Werte signalisieren. Diese Änderungsaufforderungen sind mit dem speziellen Vertrauenswert -1 außerhalb des normalen Wertebereiches von [0,100] markiert, der an jeder FPN-Stelle vor der Speicherung der Änderungsaufforderung durch den aktuellen Fuzzy Belief der Stelle ersetzt wird. Die Propagierung erfolgt analog zu der in Abschnitt 4.3.1 beschriebenen Senkung des Vertrauenswertes, so daß der gesamte Vorbereich der bestätigten Stelle durch die Änderungsanforderungen erfaßt wird.

Abbildung 4.8a stellt die Bestätigung eines Vertrauenswertes an einem Beispiel vor. Hier ist der Wert der Stelle $s_1$ vom Benutzer auf 60 herabgesetzt worden, während er den Wert der Stelle $s_2$ mit 70 bestätigt hat. Die Bestätigung wird in Form der Änderungsanforderung (-1|1,0) an den Vorbereich von $s_2$ weiterpropagiert wie in Abschnitt 4.3.1 beschrieben. Nach der in Abbildung 4.8b dargestellten Ersetzung des Sonderwertes -1 durch die jeweiligen Fuzzy Beliefs der Stellen ist zu erkennen, daß etwa der Änderung der Stelle $s_3$ auf den Wert 60, die als Seiteneffekt den die Änderung des Wertes von $s_2$ bewirkt, entgegengewirkt wird.

4.4 Statistische Auswertung

Nachdem die im vorherigen Abschnitt beschriebene Propagierung von Änderungsanforderungen abgeschlossen ist, sind an jeder Stelle des Fuzzy Petrinetzes alle Daten vorhanden, die für die Auswertung der Benutzereingaben und anschließende Anpassung der Vertrauenswerte im PDN benötigt werden. Die Auswertung dieser Daten erfolgt in vier Schritten, die in den folgenden Teilabschnitten beschrieben werden. Am Ende des Auswertungsprozesses steht schließlich die Anpassung der Vertrauenswerte im PDN gemäß der ausgewerteten Daten.
4.4 Statistische Auswertung

4.4.1 Sammeln der Änderungsanforderungen

Da nach abgeschlossener Propagierung alle Änderungsanforderungen lokal an den Stellen des FPN verfügbar sind, ist die Struktur des FPN für die weitere Auswertung nicht mehr relevant. Die Auswertung und anschließende Anpassung der Vertrauenswerte geschieht im PDN. Hierzu müssen für jede Implikation des PDN die zugehörigen Korrekturdaten aus dem FPN gesammelt werden. Dies sind genau die Änderungsanforderungen und Endorsement-Gewichte an den Bias-Stellen des FPN, die wie in Abschnitt 4.2 beschrieben die Vertrauenswerte des PDN repräsentieren. Da bezüglich der in dieser Arbeit betrachteten Vertrauens- und Schwellwerte nur die Implikationen im PDN informationstragende Knoten sind, sind die Daten aller übrigen Stellen, die die Prädikate des PDN repräsentieren, für die weitere Betrachtung nicht von Bedeutung und stellen lediglich Zwischenergebnisse während der Propagierung dar.

Für eine Implikation $i$ ergeben sich somit aus den Änderungsanforderungen $cr=(v|w_{cf,v})$ der Bias-Stellen $cf$ zu $i$ Gewichte $w_{i,v}$ mit

$$w_{i,v} = \sum_{cf \in \text{inst}(i)} w_{cf,v} \quad (4.26)$$

und analog für Endorsement-Gewichte

$$e_{i,v} = \sum_{cf \in \text{inst}(i)} e_{cf,v} \quad (4.27)$$

In der Regel ist nach diesem Schritt kein Zugriff auf die einzelnen Daten im FPN mehr erforderlich. In diesem Fall kann ganz auf die Speicherung der propagierten Daten in den Stellen des FPN verzichtet werden. Stattdessen können diese bereits während der Propagierung an den entsprechenden Implikationen gesammelt werden. Damit ist der benötigte Speicherbedarf des Adaptionserfahrens nicht mehr abhängig von der Größe des FPN, sondern nur noch von der Anzahl der Implikationen im PDN.

In einem optionalen zweiten Schritt können die soeben gesammelten Daten von Ausreißern bereinigt werden, um die Lerngenauigkeit zu erhöhen. Dieser Schritt ist für das Lernverfahren nicht zwingend erforderlich und wird daher als Erweiterung in Abschnitt 4.5.1 erläutert.

4.4.2 Zusammenführen der Lernmuster

Nachdem für das aktuelle Lernmuster, gegeben durch das vom Benutzer korrigierte FPN, alle Daten im PDN gesammelt und gegebenenfalls bereinigt sind, können sie mit den Daten älterer Lernmuster zusammengeführt werden. Sind bereits $k-1$ ältere Lernmuster vorhanden, so resultieren für die Gesamtgewichte $w_T$ und Endorsements $e_T$ die Werte

$$w_{Ti,v,k} = w_{Ti,v,k-1} + w_{i,v} \quad (4.28)$$

$$e_{Ti,v,k} = e_{Ti,v,k-1} + e_{i,v} \quad (4.29)$$
4.4 Statistische Auswertung

Diese Zusammenführung hat den Effekt, daß sich für zusätzliche Lernmuster lediglich die Gewichte erhöhen, jedoch keine Erhöhung des Speicherbedarfs entstehen. Die Gewichte $w_T$ und $e_T$ bieten sich somit zur Speicherung der gesammelten Lerndaten an, bis eine ausreichende Menge an Daten aus unterschiedlichen Reengineering-Sitzungen für die Anpassung vorhanden ist.

Das initiale Gewicht $w_{Ti,cf(i),0}$ für den bei der Musterspezifikation vergebenen Vertrauenswert $cf(i)$ ist frei wählbar. Die Wahl dieses Gewichtes bestimmt jedoch die Lernrate, mit der sich initial die Vertrauenswerte ändern. Ein hohes initiales Gewicht bewirkt eine langsamer Veränderung der Vertrauenswerte beziehungsweise erfordert entsprechend eine größere Anzahl an Lernmustern zur Veränderung. Ein niedriges Startgewicht resultiert hingegen in einer schnellen Revision des vom Reengineer bei der Spezifikation vergebenen Wertes. In der Praxis hat sich je nach Größe der analysierten Projekte ein Wert im Bereich von 5 bis 10 bewährt.

4.4.3 Adaption der Vertrauenswerte


Zuerst muß dafür jedoch die Berechnung der endgültigen Änderungsgewichte aus den Gewichten der Änderungsanforderungen und den Endorsements erfolgen. Es ist nicht sinnvoll, die Endorsements voll auf jedes Änderungsgewicht aus ihrem Einflußintervall aufzuaddieren, da hierdurch Endorsements mit niedrigem Vertrauenswert und damit großem Einflußintervall einen übermäßig großen Einfluß erhalten und die Gesamt-Gewichtsverteilung dadurch beeinträchtigen. Die Verstärkung der Änderungsanforderungen sollte vielmehr so erfolgen, daß die Endorsements anteilig auf die von ihnen beeinflußten Änderungsgewichte verteilt werden und das Verhältnis der geänderten Gewichte zueinander nicht durch die Anwendung dieses Endorsements verändert wird. Ein einzelnes Endorsement $e_{Ti,v'}$ wirkt sich somit auf die beeinflußten Änderungsgewichte $w_{Ti,v'} \forall v'>v$, wie folgt aus:

$$w'_{Ti,v} = w_{Ti,v} + \frac{w_{Ti,v}}{\sum_{k \geq v'} w_{Ti,k}} \cdot e_{Ti,v'}$$

Hieraus folgt für ein Änderungsgewicht $w_{Ti,v}$ unter Berücksichtigung aller Endorsements $e_{Ti,v'}$ mit $v'<v$

$$w'_{Ti,v} = w_{Ti,v} \left[ 1 + \sum_{v' \leq v} \left( \sum_{u \geq v'} w_{Ti,u} \right)^{-1} \cdot e_{Ti,v'} \right].$$
4.4 Statistische Auswertung

Ist die Summe \( w_{\text{total}}(i) \) der bisherigen Gewichte \( w_{Ti,v} \) bekannt, so folgt durch Umformung die effizienter zu berechnende iterative Form

\[
w_{Ti,v}' = w_{Ti,v} \cdot [1 + \omega(i, v)]
\]

mit

\[
\omega(i, v) = \sum_{v' \leq v} w_{\text{sum}}(i, v')^{-1} \cdot e_{Ti,v'}
\]

und

\[
w_{\text{sum}}(i, v) = \sum_{u \geq v} w_{Ti,u}
\]

\[
w_{\text{sum}}(i, 0) = w_{\text{total}}(i)
\]

Mit Hilfe der nach (4.32) neu gewichteten Änderungsanforderungen kann nun für jede Implikation \( i \) ein neuer Vertrauenswert \( cf_{\text{neu}}(i) \) ermittelt werden. Dieser Vertrauenswert ist so zu wählen, daß der bezüglich der Änderungsanforderungen entstehende Fehler, ausgedrückt durch eine Fehlerfunktion \( e: [0, 100] \times [0, 100] \rightarrow \mathbb{R} \), minimal ist. Mit dem Gesamtfehler \( e(i, cf) \) eines Vertrauenswertes \( cf \) an der Implikation \( i \), gegeben durch

\[
e(i, cf) = \sum_{v=0}^{100} w_{Ti,v} \cdot e(v, cf)
\]

folgt also für \( cf_{\text{neu}} \)

\[
e(i, cf_{\text{neu}}(i)) = \text{Min}\{e(i, cf)|cf \in [0, 100]\}
\]

Wie in Kapitel 3 beschrieben, wird als Fehlermaß häufig die quadratische Fehlerfunktion verwendet:

\[
e(v, cf) = (v - cf)^2
\]

Für dieses Fehlermaß wird das Minimum in (4.36) genau dann angenommen, wenn \( cf_{\text{neu}}(i) \) gleich dem gewichteten Mittel über die Änderungsanforderungen ist. Man wählt also

\[
cf_{\text{neu}}(i) = \frac{\sum_{v=0}^{100} w_{Ti,v} \cdot v}{\sum_{v=0}^{100} w_{Ti,v}}
\]

als neuen Vertrauenswert der Implikation \( i \). Setzt man eine „vernünftige“ Propagierung der Änderungsanforderungen voraus, so wird durch diese neuen Vertrauenswerte auch der Fehler der verwendeten Lernmuster minimiert. Die Gültigkeit dieser Annahme und das Ausmaß des Lernerfolges werden in Kapitel 5 empirisch untersucht.
4.5 Erweiterungen

Die Ausführungen der vorangegangenen Abschnitte beschreiben bereits einen vollständigen Ansatz zur Anpassung der Vertrauenswerte mit Hilfe statistischer Auswertungen. In diesem Abschnitt werden zusätzlich noch zwei optionale Erweiterungen vorgestellt, die den Lernerfolg des Ansatzes weiter verbessern können, jedoch für den Adoptionsmechanismus nicht zwingend erforderlich sind.

4.5.1 Bereinigung der Datensätze

Nach der in Abschnitt 4.4.1 geschilderten Sammlung der Lerndaten an den Implikationen kann optional eine statistische Bereinigung der gesammelten Daten vor der weiteren Auswertung erfolgen. Ziel einer solchen Bereinigung ist die weitere Reduktion ungünstig propagierter Änderungsanforderungen, die sich als Ausreißer in der Menge der Datensätze manifestieren. Hierzu gibt es eine Vielzahl statistischer Standardverfahren [Sac04]. Ein häufig angewendetes Verfahren basiert auf der Standardabweichung \( \sigma \) als Maß für die Kohärenz der Datensätze. Die Standardabweichung einer Menge von gewichteten Werten \( x_k \) mit Gewichten \( w_k, k \in [1, n] \) ist definiert als Quadratwurzel aus der Varianz \( \sigma^2 \) mit

\[
\sigma^2 = \frac{\sum_{k=1}^{n} w_k \cdot (x_k - \bar{x})^2}{\sum_{k=1}^{n} w_k}
\]  (4.39)

für das gewichtete Mittel \( \bar{x} \) der Werte \( x_k \)

\[
\bar{x} = \frac{\sum_{k=1}^{n} w_k \cdot x_k}{\sum_{k=1}^{n} w_k}
\]  (4.40)

Zur Bereinigung können nun alle weit außerhalb der mit Hilfe der Standardabweichung bemesenen Hauptmenge liegenden Datensätze entfernt werden:

\[
w'_{k} = \begin{cases} 
  w_k & \text{falls } |x_k - \bar{x}| \leq k\sigma \\
  0 & \text{sonst}
\end{cases}
\]  (4.41)

Für \( k \) wird üblicherweise ein Wert im Bereich zwischen 1,5 und 2 verwendet [Sac04].

Um dieses Verfahren auf die Daten der Änderungsanforderungen anzuwenden, muß bei der Berechnung des Mittelwertes und der Standardabweichung die Veränderung der Änderungsgewichte durch das Endorsements berücksichtigt werden. Die Berechnung der Änderungsgewichte unter Berücksichtigung von Endorsements erfolgt wie in Abschnitt 4.4.3 erläutert, jedoch nur für die im ersten Schritt in Abschnitt 4.4.1 gesammelten Gewichte. Die resultierenden Gewichte für eine Implikation \( i \) seien gegeben als \( \tilde{w}_{i,v} \).

Weiterhin ist zu beachten, daß Benutzereingaben mit den Werten \( v=0 \) und \( v=100 \) nicht als unscharfe Schätzwerte interpretiert werden, sondern eine absolute Ablehnung beziehungsweise Bestätigung darstellen. Obwohl diese Werte stets am äußersten Rand des Wertebereiches liegen,
ist ihre Entfernung im Rahmen der Bereinigung daher unerwünscht. Eine Menge solcher Ablehnungen und Bestätigungen kann jedoch in Form ihres Mittelwertes wieder als unscharfer Wert \( \tilde{w}_{i,v'} \) in die Berechnung der Standardabweichung einfließen. Aus Gleichung (4.39) folgt daher insgesamt für die Varianz der Änderungsanforderungen einer Implikation \( i \)

\[
\sigma^2_i = \frac{\tilde{w}_{i,v'} \cdot (v' - v)^2 + \sum_{v=1}^{99} \tilde{w}_{i,v} \cdot (v - v)^2}{\tilde{w}_{i,v} + \sum_{v=1}^{99} \tilde{w}_{i,v}} ,
\]

(4.42)

\[
\tilde{w}_{i,v'} = \tilde{w}_{i,v} \cdot 0 + \tilde{w}_{i,100} , \quad v' = \frac{\tilde{w}_{i,100} \cdot 100}{\tilde{w}_{i,v}} .
\]

(4.43)

Mit Gleichung (4.42) kann nun die Bereinigung der ursprünglichen Gewichte \( w_{i,v} \) für \( v \in [1, 99] \) gemäß (4.41) erfolgen.

Ein weiterer Ansatz zur Bereinigung mit Hilfe der Varianz basiert auf deren Interpretation als Maß für die Datenkonsistenz. Konzentrieren sich alle Änderungsanforderungen an einer Stelle \( s \) auf einen kleinen Wertebereich, ist dies ein starkes Indiz für die Qualität dieser Anforderungen und für eine entsprechende Änderung des Wertes dieser Stelle. Eine hohe Streuung und damit eine hohe Varianz der Änderungsauflorderungen an einer Stelle \( s \) deutet hingegen darauf hin, daß für \( s \) zu viele zueinander inkonsistente Daten verfügbar sind, weshalb aus der Analyse dieser Daten keine gesicherten Schlüsse gezogen werden können. Durch eine zusätzliche Gewichtung aller Änderungsauflorderungen einer Stelle \( s \) auf Basis der Varianz von \( s \) bei der in Abschnitt 4.4.1 beschriebenen Datensammlung können Stellen mit derartigen inkonsistenten Daten zugunsten von Stellen mit konsistenten Daten im Gesamtergebnis unterdrückt werden. Es ist zu beobachten, daß hierdurch häufig solche Änderungen unterdrückt werden, die aus einer ungünstigen Propagierung im FPN resultieren.

Allerdings müssen für diesen Ansatz die Änderungsauflorderungen an den Stellen des FPN verfügbar sein, was eine Speicheroptimierung wie in Abschnitt 4.4.1 beschrieben ausschließt und dadurch insbesondere für größere Projekte eine deutliche Erhöhung des Speicherbedarf bewirkt. Aus diesem Grund ist der Ansatz in dieser Form nur für kleine bis mittlere Projekte in der Größenordnung bis zu einigen wenigen zehntausend Zeilen Quelltext sinnvoll einsetzbar. Zur Wahrung der Speichereffizienz wird daher ein Ansatz verwendet, der statt der einzelnen Stellen im FPN die Menge aller Änderungsdaten \( \tilde{w}_{i,v} \) einer Implikation nach der Varianz gewichtet:

\[
\tilde{w}_{i,v} = (1 - \sigma^2_{0i}) \cdot \tilde{w}_{i,v}
\]

(4.44)

Als Gewicht \( \sigma^2_{0i} \) wird hierbei die auf das Intervall \([0,1]\) genormte Varianz der Implikation \( i \) verwendet mit

\[
\sigma^2_{0i} = \frac{\sigma^2_i}{\sigma^2_{max}} = 50^2
\]

(4.45)

**4.5 Erweiterungen**
4.5 Erweiterungen

Dieser Ansatz benötigt keine Daten im FPN, so daß eine Speicheroptimierung nach Abschnitt 4.4.1 möglich ist. Allerdings ist er wesentlich weniger feingranular bei der Gewichtung. Statt den Einfluß einzelner inkonsistenter Stellen auf die Datenmenge einer Implikation zu reduzieren, wird die gesamte Datenmenge einer inkonsistenten Implikation zugunsten konsistenterer Daten aus weiteren Lernmustern benachteiligt.

4.5.2 Wahl der Ausgabestellen

Bei allen bisherigen Betrachtungen in diesem Kapitel sind als für die Adaption betrachtete Ausgabestellen genau die Stellen des FPN gewählt worden, für die Benutzereingaben vorhanden sind. Häufig liegen jedoch nur Benutzereingaben für Stellen vor, deren Fuzzy Belief deutlich von dem Erwartungswert des Benutzers abweicht. Dementsprechend ist zumeist die Anzahl der vom Benutzer korrigierten Stellen deutlich niedriger, als die Anzahl der Musternotationen, die tatsächlich für ihn von Interesse sind.

Insbesondere die Beliefs korrekt bewerteter Annotationen werden nur selten durch den Benutzer bestätigt. Daher stellt sich das Problem, daß es den Lernmustern an Gegenbeispielen mangelt, die unerwünschten Effekte von Benutzerkorrekturen auf andere Stellen entgegenwirken. Da nach der Definition des Regressionsproblems in Kapitel 3 auch nur Ausgabestellen in die Fehlerberechnung eingehen, haben diese Seiteneffekte bisher auch keinen negativen Einfluß auf den Fehler und damit auf den scheinbaren Lernerfolg, der diesen Fehler minimiert.


Abbildung 4.9a zeigt das Problem der Seiteneffekte an einem Beispiel. Durch die Propagierung der Änderungsaufforderung von $s1$ an die gemeinsame Vorbereichsstelle $s3$ von $s1$ und $s2$ wird bei einer Änderung des zu $s3$ gehörenden Vertrauenswertes auch $s2$ beeinflußt. Diese Beeinflussung ist möglicherweise unerwünscht, was jedoch durch eine fehlende Bewertung von $s2$ nicht erkennbar ist.

Für das in Abschnitt 4.3 vorgestellte Propagierungsverfahren bedeutet die Hinzunahme nicht bewerteter Ausgabestellen, daß diese Stellen so zu behandeln sind, als seien sie vom Benutzer bestätigt worden (siehe Abschnitt 4.3.3). Da durch die fehlende Benutzerbewertung jedoch
nicht sichergestellt ist, daß die Werte der zusätzlichen Ausgabestellen korrekt sind, ist das initiale Gewicht der Änderungsanforderungen entsprechend niedriger anzusetzen. Je nachdem, in welchem Umfang unbewertete Stellen in die Ausgabemenge aufgenommen werden, erweist sich ein Wert von 0,5 oder niedriger als angemessen.

In Abbildung 4.9b ist zu erkennen, daß die Propagierung der zusätzlichen Änderungsanforderung von \( s_2 \) (hier bereits nach der Ersetzung der aktuellen Fuzzy Beliefs) der Änderung von \( s_1 \) an \( s_3 \) entgegengesetzt, so daß eine Änderung an \( s_3 \) abgeschwächt wird.

Dieser Ansatz hat also den Vorteil, daß nicht alle Stellen vom Benutzer bewertet werden müssen, um eine ausreichende Menge an Gegenbeispielen zur Reduktion unerwünschter Seiteneffekte zu erhalten. Andererseits werden damit möglicherweise auch Änderungen behindert, für die dieser Seiteneffekt erforderlich gewesen wäre. Wäre etwa der Seiteneffekt an \( s_2 \) in obigem Beispiel erwünscht gewesen, so würde durch die Hinzunahme von \( s_2 \) in die Menge der Ausgabestellen der Lernerfolg reduziert werden. Ob und in welchem Umfang die Hinzunahme zusätzlicher Ausgabestellen sinnvoll ist, hängt daher stark davon ab, in welchem Umfang Korrekturen und Bestätigungen vom Benutzer zu erwarten sind. Kann davon ausgegangen werden, daß eine umfangreiche Bewertung der Analyseegebnisse stattfindet, kann auf die Hinzunahme weiterer Stellen wahrscheinlich verzichtet werden. Ist dagegen wie beschrieben zu erwarten, daß nur die größten Fehler vom Benutzer korrigiert werden, so sollten weitere Stellen hinzugenommen werden.

4.5.3 Situationsspezifisches Wissen und gedächtnisbehaftetes Lernen

Ein Problem bei der Sammlung von Lernmustern besteht darin, daß durch unterschiedliche Implementierungsstile und Styleguides in verschiedenen Projekten oder sogar zwischen unterschiedlichen Programmierern die optimalen Vertrauens- und Schwellwerte voneinander abweichen können. Um trotzdem von den Daten anderer Projekte profitieren zu können, schlägt Strebin in [Str99] die Unterscheidung von situationsspezifischem und Hintergrundwissen vor. Dieser Ansatz läßt sich einfach auf das hier entwickelte Verfahren übertragen.

Für jede Situation (also etwa ein neues Projekt) werden die Lernmuster wie gehabt ausgewertet. Zusätzlich wird jeder Situation ein Situationskoeffizient \( d \) aus \( (0,1] \) zugeordnet, der angibt, wie

\[ \text{Abbildung 4.9: Propagierung mit zusätzlichen Stellen} \]
gut die Daten einer analysierten Situation auf andere Situationen übertragbar sind und wie gut allgemeines Hintergrundwissen auf diese Situation übertragen werden kann.

Für eine gegebene Situation mit Koeffizienten \( d \) geht das Hintergrundwissen \( w_H \) zu einer Implikation \( i \) damit wie folgt in die Auswertung der situationsspezifischen Daten \( w_{Ti,v} \) ein:

\[
w'_{Ti,v} = d \cdot w_{H,i,v} + (1 - d) \cdot w_{Ti,v}
\]  

(4.46)

Umgekehrt werden bei Beendigung einer situationsspezifischen Analyse die gesammelten Daten gemäß dem Situationskoeffizienten zum Hintergrundwissen hinzugefügt:

\[
w'_{H,i,v} = w_{H,i,v} + d \cdot w_{Ti,v}
\]  

(4.47)

Ein weiteres Problem, das sich bei der Auswertung vieler Lernmuster ergibt, ist der mit der Anzahl verfügbarer Daten abnehmende Einfluß jedes neuen Datensatzes. Ist dies unerwünscht, zum Beispiel weil berücksichtigt werden soll, daß sich das System mit der Zeit (in kleinem Umfang) ändert, so muß ein Vergessen alter Daten realisiert werden. Nach [Str99] kann dies durch einen Gedächtniskoefizienten \( \beta \) aus \((0, 1)\) realisiert werden, über den der Anteil alter Daten bei der Zusammenführung der Lernmuster reduziert wird. Auch dieser Ansatz läßt sich direkt auf das hier vorgestellte Verfahren übertragen. Gleichung (4.28) wird damit zu

\[
w_{Ti,v,k} = (1 - \beta) \cdot w_{Ti,v,k-1} + (1 + \beta) \cdot w'_{i,v}
\]  

(4.48)

### 4.6 Adaption der Schwellwerte

Bisher ist der Adaptionalgorithmus ausschließlich am Beispiel der Vertrauenswerte vorgestellt worden. In diesem Abschnitt soll nun gezeigt werden, wie dasselbe Vorgehen zur Anpassung der Schwellwerte verwendet werden kann, bevor im nächsten Abschnitt auf Basis dieser beiden Beispiele das Verfahren verallgemeinert wird.

Der Schwellwert einer Musterspezifikation dient, wie bereits erwähnt, zur Ausfilterung von Annotationen, die mit einem niedrigen Fuzzy Belief bewertet worden sind. Der Reengineer kann den Schwellwert dazu verwenden, um eine Annotation abzulehnen, ohne ihren Fuzzy Belief zu ändern. Umgekehrt kann er auch Annotationen akzeptieren, die aufgrund eines niedrigen Beliefs vom Erkennungssystem abgelehnt worden sind. Im Gegensatz zur Adaption der Vertrauenswerte hat der Reengineer hier also nicht die Möglichkeit, einen Schwellwert direkt vorzugeben. Dies erweist sich als sinnvoll, da die Schwellwerte zum Filtern einer ohnehin bereits unscharfen Größe dienen und erheblich schwerer korrekt vom Reengineer abzuschätzen sind, als etwa die Vertrauenswerte. Das Akzeptieren und Ablehnen von Patternannotationen ist jedoch ausreichend, um daraus automatisch geeignete Ziel-Schwellwerte für die Musterspezifikationen im PDN abzuleiten.

Betrachtet man die vier in Abbildung 4.10 dargestellten Möglichkeiten von Benutzereingaben, so wird schnell klar, daß es bei deren Umsetzung in passende Schwellwerte ähnliche Probleme
hinsichtlich Unsicherheit und Widersprüchlichkeit gibt, wie in Abschnitt 4.3 für die Adaption der Vertrauenswerte geschildert. Auf die ersten beiden Fälle soll nun im Einzelnen eingegangen werden, die Bestätigung von Bewertungen wird etwas später behandelt.

Das Beispiel in Abbildung 4.10a zeigt die Ablehnung einer Annotation durch den Benutzer. Zur Realisierung dieses Szenarios ist es ausreichend, wenn der Schwellwert einer Transition im Vorbereich der abgelehnten Stelle höher ist, als das niedrigste Fuzzy Belief Marking in seinem Vorbereich (siehe Definition 2.7 und Gleichung (4.2)). Ähnlich dem Senken eines Vertrauenswertes ist jedoch aus der Benutzereingabe nicht erkennbar, welcher Schwellwert anzupassen ist, um diese Bedingung zu erfüllen.

![Abbildung 4.10: Benutzereingaben für Schwellwerte](image)

Für die in Abbildung 4.10b dargestellte Akzeptierung einer zuvor vom Bewertungssystem abgelehnten Annotation müssen hingegen zur Realisierung alle Schwellwerte im Vorbereich der akzeptierten Stelle $s$ unter den Fuzzy Beliefs ihrer jeweiligen Vorbereiche liegen. Denn nach Gleichung (4.2) würde bereits ein Fall, in dem ein Schwellwert im Teilbaum unter $s$ nicht erreicht wird, für die Ausfilterung von $s$ sorgen. Analog zum Erhöhen eines Vertrauenswertes sind daher alle bisher nicht erreichten Schwellwerte entsprechend zu senken. Die aktuellen Vertrauenswerte geben hierbei, wie im Falle der Vertrauenswert-Anpassung, nur die minimal erforderliche Korrektur vor, während die tatsächliche Korrektur stärker ausfallen kann.

Da der Benutzer die Schwellwerte nicht direkt vorgeben kann, steht im Gegensatz zu dem in Abschnitt 4.3 beschriebenen Ansatz die Zielgröße, hier also der gewünschte Schwellwert, nicht direkt für die Propagierung zur Verfügung. Der Schwellwert zu einer Transition kann jedoch aus den Fuzzy Beliefs abgeleitet werden, die diesen gemäß Benutzereingabe über- beziehungsweise unterschreiten sollen. Daher werden statt der nicht bekannten Schwellwerte die Fuzzy Beliefs der vom Benutzer akzeptierten oder abgelehnten Musterinstanzen als Änderungsanforderungen im FPN propagiert. Die Schwellwerte können damit dann so festgelegt werden, daß sie möglichst viele der propagierten Beliefs korrekt einordnen.
4.6 Adaption der Schwellwerte

Da, wie soeben erläutert, die beiden betrachteten Fälle analog zum Senken und Steigern der Vertrauenswerte betrachtet werden können und dieselben Überlegungen zu Unsicherheit und Widersprüchlichkeit gelten, wie in Abschnitt 4.3 beschrieben, können die in Abschnitt 4.3 beschriebenen Propagationsvorschriften unmittelbar übernommen werden. Bei der Propagation werden jedoch Änderungsanforderungen $cr'(v,w)$ zur Ablehnung und Anforderungen $cr^+(v,w^*)$ zur Akzeptierung an den Stellen $s$ des FPN für die statistische Auswertung getrennt voneinander in Form von Gewichten $w^-_{s,v}=w^-$ und $w^+_{s,v}=w^+$ gesammelt.

Da nur die aus der Benutzerbewertung resultierende Akzeptierung oder Ablehnung möglicher Musterinstanzen in die Änderungsanforderungen eingeht, nicht jedoch die ursprüngliche Situation, können die beiden bisher nicht betrachteten Fälle der Bestätigung einer Positiv- beziehungsweise Negativbewertung (Abbildung 4.10c und d) analog zur Akzeptierung respektive Ablehnung der Analyseergebnisse behandelt werden.

Weil zudem die neuen Schwellwerte einzig auf Basis der Ablehnung oder Akzeptierung der Fuzzy Beliefs bestimmt werden, sind die bisherigen Schwellwerte im FPN für die Propagierung nicht relevant und es kann wiederum die vereinfachte Bias-Darstellung aus Abschnitt 4.3 verwendet werden. Die Änderungsanforderungen zu jedem Schwellwert können dann bei der statistischen Auswertung nach Abschnitt 4.4 wiederum an den Bias-Stellen $cf$ der zugehörigen Transitionen gesammelt werden. Durch die getrennte Betrachtung von Ablehnungs- und Akzeptierungsanforderungen ergeben sich nach der Sammlung entsprechend zwei Datenmengen:

$$
\begin{align*}
  w^+_{Ti,v} &= \sum_{cf \in \text{inst}(i)} w^+_{cf,v} \\
  w^-_{Ti,v} &= \sum_{cf \in \text{inst}(i)} w^-_{cf,v}
\end{align*}
$$

(4.49)

Analog zur Berechnung der neuen Vertrauenswerte kann auf Basis dieser Daten ein neuer Schwellwert $th_{\text{neu}}$ berechnet werden, der den Fehler gegenüber den gesammelten Änderungsanforderungen minimiert. Hierbei ist jedoch zu beachten, daß sich durch die Schwellwertbildung die zu verwendende Fehlerfunktion ändert. Für jeden Fuzzy-Wert $v$, der unter dem neuen Schwellwert liegen soll, entsteht für $v \geq th_{\text{neu}}$ ein Fehler in Höhe von $v$, da der Schwellwert den berechneten Fuzzy-Wert $v$ nicht wie gefordert auf 0 absenkt. Entsprechendes gilt für einen Fuzzy-Wert $v < th_{\text{neu}}$, der eigentlich über dem Schwellwert liegen soll. Die zu minimierende Fehlerfunktion für den Schwellwert $th$ einer Implikation $i$ ergibt sich somit zu

$$
e(i, th) = \sum_{v \geq th} w^+_{Ti,v} \cdot v + \sum_{v < th} w^-_{Ti,v} \cdot v
$$

(4.50)

und $th_{\text{neu}}$ ist so zu wählen, daß gilt

$$
e(i, th_{\text{neu}}(i)) = \text{Min}\{\Delta(i, th)|th \in [0, 100]\}
$$

(4.51)
th_{neu} kann damit durch einen einfachen linearen Durchlauf der 101 möglichen Werte aus \([0,100]\) bestimmt werden.

4.7 Verallgemeinerung

Bisher ist bei der Entwicklung des hier vorgestellten statistischen Adaptionverfahrens lediglich die Anwendung auf den in Kapitel 2 geschilderten Inferenzprozess und die daran beteiligten Pattern Dependency Nets und Fuzzy Petrinetze betrachtet worden. Der dabei anhand der Adaption der Vertrauens- und Schwellwerte entworfene Algorithmus soll nun hinsichtlich seiner Anwendbarkeit auf andere Netzstrukturen untersucht und zu diesem Zweck verallgemeinert werden.


Desweiteren ist zu berücksichtigen, daß der hier beschriebene statistische Ansatz eine große Menge an Lerndaten für jede anzupassende Größe benötigt, um eine repräsentative statistische Basis sicherzustellen. Aufgrund der in Abschnitt 4.4 beschriebenen Zusammenführung der Lerndaten ist es dabei unerheblich, ob diese Daten aus vielen unterschiedlichen Lernmustern stammen oder aus Mehrfachvorkommen der anzupassenden Größen in wenigen Lernmustern (vgl. jedoch Synergieeffekte in Abschnitt 5.2).

Ein weiteres wichtiges Kriterium für die Anwendung dieses Ansatzes ist ein nicht zu komplexer funktionaler Zusammenhang zwischen Vor- und Nachbereichsstellen in den betrachteten Netzen. Für die Rückpropagierung von Änderungsanforderungen von einem Knoten \(k\) des Netzes an alle Knoten, die den Wert von \(k\) beeinflussen, ist es erforderlich, bestimmen zu können, welche Möglichkeiten zur Anpassung dieser Knoten existieren, um den gewünschten Wert für den Knoten \(k\) zu erhalten. Im Falle der FPNs war dies durch die Minimierungsfunktion besonders
4.7 Verallgemeinerung

einfach. In anderen Fällen kann es jedoch je nach zugrundeliegender Berechnungsfunktion nicht oder nur schwer möglich sein.

Im Folgenden wird angenommen, daß diese Bedingungen erfüllt sind. Dann sei für einen Knoten \( k \) aus dem Vorbereich eines Knotens \( l \) der erforderliche Wert \( v \), um an \( l \) den Wert \( u \) zu induzieren, gegeben durch die Rückpropagierungsfunktion \( r \) mit

\[
r(l, k, u) = v
\]

(4.52)

Ferner sei eine Gewichtsfunktion \( c \) gegeben, die für eine Änderungsanforderung \( cr=(u,w) \), die von \( l \) an \( k \) propagiert wird, ein Maß für die Sicherheit ist, daß \( k \) den Wert \( v=r(l,k,u) \) annehmen muß, um den Wert \( u \) in \( l \) zu realisieren. Eine geeignete Gewichtsfunktion kann in der Regel mit den in Abschnitt 4.3 angestellten Überlegungen und einigen empirischen Untersuchungen aus den betrachteten Netzstrukturen und funktionalen Zusammenhängen abgeleitet werden.

Bei der Propagierung einer Änderungsanforderung \( cr=(u|w) \) von einem Knoten \( l \) an einen Knoten \( k \) folgt damit für die resultierende Änderungsanforderung \( cr' \) an \( k \) (unter Vernachlässigung anderer propagierter Änderungsanforderungen)

\[
\begin{align*}
\text{cr}' &= (r(l, k, u))|c(l, k, u) \cdot w)
\end{align*}
\]

(4.53)

Für gegebene Funktionen \( r \) und \( c \) ändert sich damit die Propagierungsvorschrift (4.14) im allgemeinen Fall zu

\[
\begin{align*}
w_{k, u, t} &= w_{k, u, t-1} + \sum_{k' \in \text{post}(k)} c(k', k, v) \cdot \Delta w_{k', v, t} \\
\text{mit } v &= r(k', k, u)
\end{align*}
\]

(4.54)

Die so propagierten Änderungsanforderungen können dann wie in Abschnitt 4.4 beschrieben ausgewertet werden, um den neuen Wert der zu Knoten \( k \) gehörigen Größe zu bestimmen.
KAPITEL 5

Evaluation


5.1 Vorgehen


In dem in Abbildung 3.1 dargestellten Schema eines Lernenden-Systemes ersetzt der Generator somit die Eingabekomponente und das System. Zudem bietet es sich an, zur Reproduktion der Evaluationsergebnisse und für weitere Analysen die vom Generator erzeugten Lernbeispiele zu speichern. Insgesamt ergibt sich somit das in Abbildung 5.1 dargestellte Schema.

Ein solches Generatorkonzept wird im Folgenden für die Erzeugung von Fuzzy Petrinetzen vorgegebener Größe und zugehörige Korrekturdaten zu einem gegebenen PDN vorgestellt. Um die Struktur durch tatsächliche Analysen entstandener FPNs möglichst gut nachzuahmen, bietet es
5.1 Vorgehen

sich an, für die Erzeugung ebenfalls den in Kapitel 2 vorgestellten Inferenzalgorithmus zu verwenden. Da hierzu kein Quelltext verfügbar ist, auf den die Musterspezifikationsregeln ange-wandt werden können, wird ihre Anwendung durch einen Zufallsprozess ersetzt, der über Erfolg oder Mißerfolg einer Regelanwendung entscheidet. Damit die so erzeugten Netze möglichst realistisch sind, können die für den Zufallsprozess erforderlichen stochasticischen Kennwerte anhand realer Beispiele ermittelt werden. Dies sind im Einzelnen:

- Für jedes Axiom im PDN der erwartete prozentuale Anteil von Axiomsstellen an der Gesamtknotenzahl des FPNs, sowie die Varianz dieses Wertes. Hiermit kann die Anzahl der zu erzeugenden Axiomsstellen in Abhängigkeit von der gewünschten Netzgröße als normal-verteilte Zufallsvariable modelliert werden.

- Für die Bottom-Up-Inferenz die bedingte Wahrscheinlichkeit $p_{up}(P|Q)$, daß eine Instanz eines Prädikates $Q$ eine Instanz des von $Q$ getriggerten Prädikates $P$ in ihrem Nachbereich hat. Für eine in $[0,1]$ gleichverteilte Zufallsvariable $x$ gilt $P$ dann als erfolgreich erkannt, wenn $x \leq p_{up}(P|Q)$ gilt\(^1\).


- Für die Bindung bereits angelegter Patterninstanzen $s$ eines Prädikates $P$ als Prämisse weiterer Patterninstanzen die bedingte Wahrscheinlichkeit $p_{out}(x>d|P)$, daß der Ausgangsgrad der Stelle $s$ größer als der aktuelle Grad $d=|post(s)|$ ist.

Diese Werte können durch einfaches Abzählen aus einer Reihe von Beispielanalysen realer Softwaresysteme ermittelt werden. Für $p_{up}$ gilt etwa

$$p_{up}(P|Q) = \frac{|P_Q|}{|inst(Q)|}, \quad (5.1)$$

mit der Menge $P_Q$ von Instanzen von $P$ mit einer Instanz von $Q$ im Vorbereich:

$$P_Q = \{ s \mid s \in inst(P) \land pres_S(s) \cap inst(Q) \neq \emptyset \} \quad (5.2)$$

Die Ermittlung der übrigen Werte erfolgt analog.

Für die im Folgenden durchgeführten Versuche sind zur Bestimmung dieser Kennwerte die Mustererkennungsergebnisse mehrerer realer Systeme ausgewertet worden. Bei den betrachteten Systemen handelt es sich um die GUI-Bibliotheken AWT und Swing des Java Runtime Environment (ca. 70.000 und 150.000 LOC), sowie Teile des UML-Metamodells der Entwicklungsumgebung FUJABA (ca. 55.000 LOC). Bei dem für diese Analysen und die weitere Gene-

Neben der Erzeugung der FPNs ist der Generator auch für die Bereitstellung von simulierten Benutzereingaben verantwortlich. Hierzu werden für das betrachtete PDN neue Zielwerte für

\(^1\) Die zufällige Erkennung einer Musterinstanz ist also als Bernoulli-Experiment mit Erwartungswert $p_{up}$ modelliert.
die Vertrauens- und Schwellwerte vorgegeben, entweder durch den Benutzer oder zufällig durch einen um den aktuellen Wert normalverteilten Zufallsprozess. Wertet man mit den neuen Vertrauens- und Schwellwerten die erzeugten FPNs aus, können die errechneten Fuzzy Beliefs als Sollwerte für Korrekturen an den FPNs mit den ursprünglichen Vertrauens- und Schwellwerten verwendet werden.

Um bei den im Folgenden betrachteten Versuchsdaten repräsentative Ergebnisse sicherzustellen, sind alle Versuche je fünfzig mal\(^1\) mit unterschiedlichen Lernmustern und unterschiedlichen Zielwerten im PDN durchgeführt worden. Die aufgeführten Ergebnisse sind als arithmetische Mittel der Einzelergebnisse zu verstehen. Als Fehlermaß wird der aus Kapitel 3 bekannte mittlere quadratische Fehler (Mean Square Error, MSE) verwendet. Um die Fehlerwerte besser in Relation zu den Fuzzy Belief Werten setzen zu können, sind die aufgeführten Fehlerwerte die Quadratwurzeln der ursprünglichen, quadrierten Fehlerterme.

Messwerte zum Laufzeitbedarf sind mit Hilfe einer Profiler-Software ermittelt worden, so daß nur die tatsächlich von dem betrachteten System (der Infrerenzmaschine beziehungsweise der Lernmaschine) verwendete Laufzeit in die Evaluation eingehen und die Ergebnisse nicht durch Hintergrundprozesse des Betriebssystems beeinflußt werden.

Als Testsystem kommt ein PC mit AMD AthlonXP 1900+ Prozessor mit 1,6GHz Taktzyklus und 1,5GB RAM zum Einsatz. Das verwendete Betriebssystem ist Microsoft Windows 2000.

5.2 Vergleich mit Fuzzy Neural Nets

Im Folgenden werden nun die mit dem in dieser Arbeit entwickelten statistischen Ansatz erzielten Lernergebnisse mit den Ergebnissen des etablierten Backpropagation-Ansatzes mit Fuzzy Neural Nets verglichen. Von Interesse hierbei sind insbesondere die für das Lernen benötigte Zeit und der erzielte Lernerfolg.

In einem ersten Versuch werden hierzu beiden Ansätzen FPNs unterschiedlicher Größe als Lernmuster übergeben, um das Verhalten beider Ansätze für verschiedene Mengen an zu erlernenden Korrekturwerten zu evaluieren. Für beide Ansätze kann die benötigte Rechenzeit in zwei Phasen unterteilt werden.

Die erste Phase ist die Aufbereitung der Lerndaten bei ihrer Erfassung, also im Anschluß an die Fuzzy-Bewertung und die Begutachtung durch den Benutzer. Bei dem Backpropagation-Ansatz wird hierbei das erzeugte FPN in Form mehrerer Lernmuster auf das FNN abgebildet. Beim statistischen Ansatz erfolgt in dieser Phase die Propagierung der Änderungsanforderungen. Da diese Phase online, also im Rahmen der Inferenz erfolgt, ist hier entscheidend, daß der verwendete Lernansatz die Dauer der Inferenz nicht signifikant erhöht. Tabelle 5.1 zeigt die Evaluationsergebnisse dieser Phase.

---

\(^1\) Bei stark zeitaufwendigen Versuchen zwanzig mal
5.2 Vergleich mit Fuzzy Neural Nets

Die erste Spalte der Tabelle zeigt die jeweilige Größenordnung der für das Training verwendeten FPNs. Die nächste Spalte \( t_{\text{inf}} \) gibt einen ungefähren Erfahrungswert für die typische Dauer des Inferenzprozesses für Analysen der gegebenen Größenordnung an. Die dritte Spalte \( t_{\text{fpn}} \) zeigt die durchschnittliche Auswertungsdauer eines FPNs der angegebenen Größe. Die von den beiden Ansätzen in dieser Phase pro FPN benötigte Zeit ist in den Spalten Vier \( t_{\text{stat}} \) und Sechs \( t_{\text{FNN}} \) aufgeführt. Zusätzlich enthält Spalte Fünf \(#\text{In}\) die durchschnittliche Anzahl der für das FNN aus jedem FPN erzeugten Lernmuster.

<table>
<thead>
<tr>
<th>FPN</th>
<th>Inferenz</th>
<th>Statistisch</th>
<th>FNN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( t_{\text{inf}} ) [s]</td>
<td>( t_{\text{fpn}} ) [s]</td>
<td>( t_{\text{stat}} ) [s]</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>0,37</td>
<td>0,74</td>
</tr>
<tr>
<td>100</td>
<td>120</td>
<td>1,91</td>
<td>4,21</td>
</tr>
<tr>
<td>1.000</td>
<td>900</td>
<td>20,4</td>
<td>52,7</td>
</tr>
<tr>
<td>10.000</td>
<td>2400</td>
<td>198</td>
<td>589</td>
</tr>
<tr>
<td>100.000</td>
<td>_1</td>
<td>1974</td>
<td>6416</td>
</tr>
</tbody>
</table>

Tabelle 5.1 Aufbereitung der Lernmuster

Es ist zu erkennen, daß bei beiden Lernverfahren die Aufbereitungsphase länger dauert, als die Auswertung des FPNs. Jedoch wächst die Laufzeit mit zunehmender Netzgröße im Fall der statistischen Auswertung der FPNs deutlich langsamer, als bei den Fuzzy Neural Nets. Für große FPNs mit etwa 10.000 Stellen, was einem analysierten System von mehreren 100.000 Zeilen Quelltext entspricht, liegt die Laufzeit im statistischen Fall damit immer noch etwa in derselben Größenordnung, wie die Auswertungszeit des FPN. Für die Fuzzy Neural Nets hingegen wird durch die Aufbereitung die Inferenzdauer mehr als verdoppelt. Außerdem ist zu erkennen, daß die Anzahl der vom FNN zu verarbeitenden Lernmuster mit steigender Netzgröße immer stärker anwächst.

Die zweite Phase des Lernprozesses beginnt für beide Ansätze sobald ausreichend Lernmuster gesammelt worden sind und der Benutzer die Anpassung der Vertrauens- und Schwellwerte auf Basis dieser Daten anstößt. Bei dem FNN-Ansatz findet nun der Backpropagation-Prozess mit den zuvor erzeugten Lernmustern statt. Für den statistischen Ansatz ist in dieser Phase hingegen lediglich die in Abschnitt 4.4 geschilderte statistische Auswertung im zugehörigen PDN erforderlich. Wie Tabelle 5.2 zeigt, ist daher die Laufzeit \( t_{\text{stat}} \) für das statistische Verfahren in dieser Phase unabhängig von der Anzahl und Größe der Lernmuster und im Vergleich zu den anderen Zeiten vernachlässigbar klein. Im Gegensatz dazu steigt die Laufzeit \( t_{\text{FNN}} \) des Backpropagation-Verfahrens im Neuronalen Netz für größere FPNs so stark an, daß bei FPN-Größen von 10.000 Stellen bereits die praktischen Grenzen der Versuchsduerdhführung erreicht worden sind, so daß bei diesem Versuch auf die übliche Mittelung mehrerer Auswertungen verzichtet worden ist.

Der Lernerfolg der beiden Ansätze kann an der Verringerung des durchschnittlichen Fehlers abgelesen werden. Der Fehler \( e_{\text{inf}} \) vor Anwendung der Lernansätze ist in Spalte Vier von Tabelle 5.2 aufgezeigt.
5.2 Vergleich mit Fuzzy Neural Nets

5.2 aufgeführt. Er schwankt für alle betrachteten Beispiele um den Wert 25, weil für die Bestimmung der Sollwerte im PDN ein normalverteilter Zufallsprozeß mit Standardabweichung 25 um den alten Vertrauenswert verwendet worden ist. Das Fuzzy Neural Net leistet in allen Fällen eine Reduzierung des Fehlers auf Werte im Bereich von 3,0. Der Lernerfolg ist hier bereits für wenige Lerndaten sehr gut und stabilisiert sich schnell mit zunehmender Datenmenge.


Betrachtet man die Abnahme des mittleren Fehlers mit zunehmender Stellenzahl der FPNs für den statistischen Ansatz, so stellt sich die Frage, ob die Abnahme einzig durch die Erhöhung der Gesamtmenge der Stellen hervorgerufen wird, oder ob auch bei insgesamt gleicher Stellenzahl große Netze besser für diesen Ansatz geeignet sind. Dieser Frage soll in einem zweiten Versuch nachgegangen werden. Dazu wird die Zahl der verwendeten FPNs für jede Netzgröße so gewählt, daß die Gesamtstellezahl in allen Fällen in etwa gleich ist. Die Ergebnisse dieses Versuches sind in Tabelle 5.3 aufgeführt.

```
<table>
<thead>
<tr>
<th>FPN</th>
<th>FPNs</th>
<th>$t_{inf}[s]$</th>
<th>$e_{inf}$</th>
<th>$t_{stat}[s]$</th>
<th>$e_{stat}$</th>
<th>$t_{FNN}[s]$</th>
<th>$e_{FNN}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>10</td>
<td>27,3</td>
<td>3,06</td>
<td>17,31</td>
<td>15</td>
<td>3,61</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>120</td>
<td>23,9</td>
<td>2,98</td>
<td>9,84</td>
<td>193</td>
<td>2,92</td>
</tr>
<tr>
<td>1.000</td>
<td>10</td>
<td>900</td>
<td>24,1</td>
<td>3,11</td>
<td>7,13</td>
<td>3751</td>
<td>2,74</td>
</tr>
<tr>
<td>10.000</td>
<td>5</td>
<td>2400</td>
<td>26,1</td>
<td>3,09</td>
<td>6,75</td>
<td>65113^1</td>
<td>2,81</td>
</tr>
<tr>
<td>100.000</td>
<td>5</td>
<td>--</td>
<td>24,7</td>
<td>3,14</td>
<td>6,81</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>
```

1: Nur ein Durchlauf, keine Mittelung mehrerer Versuche aufgrund zu hoher Laufzeit
2: Aufgrund zu erwartender Laufzeit nicht durchgeführt

Tabelle 5.2 Anpassung der Vertrauens- und Schwellwerte

5.2 Vergleich mit Fuzzy Neural Nets

Evaluation 75
5.2 Vergleich mit Fuzzy Neural Nets


<table>
<thead>
<tr>
<th>FPN</th>
<th>FPNs</th>
<th>e_{stat}</th>
<th>t_{FNN}[s]</th>
<th>t_{inf}[s]</th>
<th>e_{FNN}</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>17,31</td>
<td>2,8</td>
<td>10</td>
<td>7,81</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>9,84</td>
<td>27,6</td>
<td>120</td>
<td>7,25</td>
</tr>
<tr>
<td>1.000</td>
<td>10</td>
<td>7,13</td>
<td>556</td>
<td>900</td>
<td>13,6</td>
</tr>
<tr>
<td>10.000</td>
<td>5</td>
<td>6,75</td>
<td>11098</td>
<td>2400</td>
<td>17,6</td>
</tr>
</tbody>
</table>

_Tabelle 5.4 Fehler des FNN bei vorzeitigem Abbruch_

In Spalte Drei und Vier von Tabelle 5.4 ist zu erkennen, daß die benötigte Berechnungszeit bis zum Erreichen von e_{stat} zwar deutlich niedriger ist, als die in Tabelle 5.2 aufgeführte Zeit bis zur Stabilisierung des Netzes. Sie liegt jedoch für große Netze immernoch deutlich über der für den Inferenzprozess benötigten Zeit. Und auch für die mittelgroßen Netze mit 100 und 1000 Stellen ist sie noch deutlich höher, als die Zeit, die für die statistische Auswertung bei gleichem Ergebnis benötigt wird.

Spalte Fünf und Sechs zeigen, daß auch bei Limitierung der Trainingszeit auf die Inferenzdauer t_{inf} nur für kleine Netze unter 1000 Stellen akzeptable Ergebnisse erreicht werden. Für größere Netze wächst durch die Zeitbeschränkung der Fehler dagegen stark an und überschreitet den bei dem statistischen Ansatz verbleibenden Fehler deutlich.
5.3 Fazit

Die bei der Evaluation gewonnenen Ergebnisse zeigen, daß die Entwurfskriterien aus Kapitel 4 erfolgreich umgesetzt worden sind. Der statistische Ansatz skaliert auch für sehr große Systeme mit resultierenden Fuzzy Petrinetzen im Umfang von über 10.000 Stellen, was einem Quelltextumfang von mehreren 100.000 Zeilen entspricht. Der durch die Lernmaschine erzeugte Mehraufwand während der Musteranalyse liegt mit dem circa dreifachen der Auswertungszeit der FPNs beziehungsweise bis zu einem Viertel der Gesamtinferenzdauer in einem vertretbaren Bereich und die für das Training mit den gesammelten Lernmustern benötigte Zeit ist vernachlässigbar klein. Damit ist der Ansatz laufzeiteffizient genug, um etwa auch ein interaktives Online-Training zu ermöglichen, bei dem für jedes neue Lernmuster das PDN angepaßt und dem Benutzer die resultierende Veränderung der Fuzzy-Werten präsentiert werden kann. Die durch die statistische Adaption erreichte Fehlerreduktion ist wie gesehen ebenfalls hinreichend gut, um die auf Basis der korrigierten Vertrauens- und Schwellwerte berechneten Fuzzy-Werte als zuverlässige Richtgrößen für die Musterauswertung durch den Reengineer verwenden zu können. Hier ist gegebenenfalls auf andere Methoden auszuweichen, falls hauptsächlich kleine Systeme oder Systemteile analysiert werden.

Im Gegensatz zur statistischen Adaption liefert der konventionelle Regressionsansatz mit Fuzzy Neural Nets und Backpropagation zwar eine durchgehend bessere Fehlerreduktion. Jedoch ist die hierfür benötigte Berechnungszeit so hoch, daß dieser Ansatz für die praktische Verwendung in dem hier betrachteten Einsatzfeld nicht geeignet ist. Bereits die Bearbeitung eines neuen Lernmusters überschreitet für große Systeme die Inferenzdauer deutlich und auch die Trainingsdauer ist, mit bis zu 18 Stunden für Netze über 10.000 Stellen, unvertretbar hoch. Lediglich bei kleinen Netzen bis zu 1000 Stellen kann dieser Ansatz sinnvoll eingesetzt werden. Da gerade in diesem Bereich die Schwächen der statistischen Adaption liegen, kann gegebenenfalls eine Kombinations-Strategie der beiden Ansätze in Betracht gezogen werden.
5.3 Fazit
Zusammenfassung und Ausblick

6.1 Zusammenfassung

In dieser Arbeit ist ein statistisches Adaptionsverfahren vorgestellt worden, dessen Ziel die Optimierung der unscharfen Gütebewertung von Entwurfmustern ist, die zuvor durch ein Mustererkennungssystem gefunden worden sind. Die durch die Gütebewertung berechneten Fuzzy-Werte helfen dabei, sicher erkannte Muster von solchen zu unterscheiden, die einer weiteren Überprüfung durch den Reengineer bedürfen, und sind somit ein wichtiger Faktor für eine effektive und zuverlässige Mustererkennung. Im Vergleich zu Abbildung 1.1 aus der Einführung ist mit dem entwickelten Adaptionsverfahren somit die in Abbildung 6.1 grün hervorgehobene Rückkopplung zwischen Benutzerkorrekturen und den Vertrauens- und Schwellwerten der Musterspezifikation realisiert worden.

Abbildung 6.1: Das Pattern-Klassendiagramm
6.1 Zusammenfassung


Zur Berechnung der optimierten Bewertungsparameter ist ein einfaches statistisches Verfahren vorgestellt worden, bei dem die gesammelten Änderungsdaten mit Hilfe statistischer Standardverfahren zunächst bereinigt werden können, bevor durch Minimierung des lokalen Fehlers die neuen Vertrauens- und Schwellwerte bestimmt werden. Im Falle der Vertrauenswerte erfolgt die Fehlerminimierung durch gewichtetes Mitteln der gesammelten Daten, im Falle der Schwellwerte durch lineare Suche des optimalen Wertes.

Für den zum Vergleich herangezogenen klassischen Regressionsansatz mit neuronalen Netzen und Backpropagation hat sich im Gegensatz dazu gezeigt, daß zwar eine bessere Fehlerreduktion erreicht wird, das Verfahren jedoch aufgrund der mit der Größe der analysierten Systeme stark ansteigenden Lerndauer für den praktischen Einsatz mit großen Netzen ungeeignet ist.

### 6.2 Ausblick

#### 6.2.1 Statistische Adaption


Das vorgestellte Adaptoverfahren kann in zwei Teile untergliedert werden, die Aufbereitung der Korrekturdaten im FPN und die statistische Ermittlung der neuen Werte auf Basis der gesammelten Lerndaten. Für den ersten Teil ist hier eine einfache, gewichtete Propagationsheuristik verwendet worden. In Umgebungen mit weniger starken Anforderungen hinsichtlich der Skalierbarkeit kann dieses Verfahren unter Umständen durch ein optimaleres, gegebenenfalls auch iteratives Verfahren ersetzt werden, das für ein gegebenes Lernmuster die optimalen Parameter bestimmt. Die Bestimmung des optimalen Wertes über alle Lernmuster kann dann wiederum durch statistische Auswertung der Einzelwerte der Lernmuster erfolgen. So kann
unter Umständen der Lernerfolg durch eine etwas laufzeitintensivere Aufbereitung in Kombination mit der einfachen und effizienten Zusammenführung der aufbereiteten Daten gesteigert werden. Durch einen solchen Kombinationsansatz kann unter Umständen die Laufzeit des Lernprozesses gegenüber einer rein iterativen Lösung auf Kosten der Fehlerreduktion stark reduziert werden, da im Allgemeinen die iterative Optimierung jedes einzelnen Lernmusters deutlich einfacher ist, als die Iteration über alle Lernmuster.

6.2.2 Optimierungsmöglichkeiten im Inferenzprozess

Neben der hier betrachteten Optimierung der Parameter der Fuzzy-Bewertung existieren noch mindestens zwei weitere Bereiche im vorgestellten Inferenzprozess, in denen eine computergestützte Optimierung möglich erscheint.


[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley, Reading, MA, 1995.


ANHANG B

Abbildungsverzeichnis

ABBILDUNG 1.1: ÜBERBLICK ÜBER DEN MUSTERERKENNUNGSPROZESS ................................................................. 3
ABBILDUNG 2.1: ASG-REPRÄSENTATION VON QUELLCODE .............................................................................. 6
ABBILDUNG 2.2: DAS COMPOSITE-PATTERN ........................................................................................................... 7
ABBILDUNG 2.3: EINE VARIANTE DES COMPOSITE-PATTERNS ............................................................................. 8
ABBILDUNG 2.4: IMPLEMENTIERUNG VON ZU-N-ASSOZIATIONEN ............................................................ 9
ABBILDUNG 2.5: FORMALE SPEZIFIKATION DES COMPOSITE-PATTERNS .................................................. 9
ABBILDUNG 2.6: DAS PATTERN-KLASSEN DIAGRAMM ......................................................................................... 11
ABBILDUNG 2.7: AUSSCHNITT EINES PATTERN DEPENDENCY NETS ............................................................ 14
ABBILDUNG 2.8: BEISPIEAUSFÜHRUNG DES ERKENNUNGSPROZESSES .................................................. 19
ABBILDUNG 2.9: AUSSCHNITT EINES FUZZY PETRINETZES ............................................................................. 22
ABBILDUNG 2.10: FUZZY PETRINETZ NACH DER AUSWERTUNG .............................................................. 25
ABBILDUNG 2.11: INFERENZPROZESS STATECHART .......................................................................................... 25
ABBILDUNG 3.1: LERNEN EINES REGRESSIONS PROBLEMS .............................................................................. 31
ABBILDUNG 3.2: DIE LERNMASCHINE IM DETAIL ............................................................................................. 32
ABBILDUNG 3.3: EIN NEURON .......................................................................................................................... 36
ABBILDUNG 3.4: IMPLIKATION IM FNN ............................................................................................................. 37
ABBILDUNG 3.5: UMSETZUNG VON VERERBUNG IM FNN ...................................................................................... 38
ABBILDUNG 3.6: FUZZY PETRINETZ NACH DER AUSWERTUNG ................................................................. 41
ABBILDUNG 3.7: BACKPROPAGATION ALS PSEUDO-CODE .................................................................................. 42
ABBILDUNG 4.1: REALISIERUNGSMÖGLICHKEITEN ......................................................................................... 47
ABBILDUNG 4.2: VERTRAUENSWERTE ALS BIAS-STELLEN ................................................................................ 49
ABBILDUNG 4.3: REALISIERUNGSMÖGLICHKEITEN .......................................................................................... 50
ABBILDUNG 4.4: GEWICHTETE PROPAGATION DER VERTRAUENSWERTE ...................................................... 52
ABBILDUNG 4.5: PROPAGIERUNG DER ÄNDERUNGSANFORDERUNGEN ....................................................... 53
ABBILDUNG 4.6: PROPAGIERUNG MIT ENDORSEMENTS ..................................................................................... 55
ABBILDUNG 4.7: REALISIERUNGSMÖGLICHKEITEN .......................................................................................... 56
ABBILDUNG 4.8: BESTÄTIGEN VON FUZZY-WERTEN .......................................................................................... 58
ABBILDUNG 4.9: PROPAGIERUNG MIT ZUSÄTZLICHEN STELLEN ....................................................................... 65
ABBILDUNG 4.10: BENUTZEREINGABEN FÜR SCHWELLWERTE ........................................................................... 67
ABBILDUNG 5.1: GENERIEREN VON LERNMUSTERN ......................................................................................... 71
ABBILDUNG 6.1: DAS PATTERN-KLASSEN DIAGRAMM ....................................................................................... 79