Werkzeugspezifikation mit Schemakorrespondenzen

*Stefan Gruner**

Lehrstuhl für Informatik III (Softwaretechnik)
der Rhein.-Westf. Technischen Hochschule,
Ahornstraße 55 / D - 52074 Aachen
stefan@i3.informatik.rwth-aachen.de

*gefördert durch die Deutsche Forschungsgemeinschaft im Graduiertenkolleg "Informatik und Technik" der RWTH

Zusammenfassung: *Integrationswerkzeuge sind wichtige Hilfsmittel zur Sicherung der wechselseitigen Konsistenz voneinander abhängiger Dokumente. Dieser Beitrag behandelt eine neue Methode zur Spezifikation von Integrationswerkzeugen. Ein Beispiel aus der Softwaretechnik erläutert das Prinzip der bisherigen graphgrammatischen Werkzeugspezifikation und motiviert deren Verbesserung durch die neue Schemakorrespondenzmethode, welche künftig auch zur Lösung neuerer Integrationsprobleme aus der chemischen Verfahrenstechnik verwendet werden soll.*

Schlüsselwörter: Integrationswerkzeug, Integrationsregel, Paargraphgrammatik, Graphenschema, Korrespondenz

1 Einführung

Komplexitätsreduzierende Arbeitsteilung ist nicht allein in der industriellen Produktion, sondern auch im Entwurf technischer Systeme gängige Praxis. Daß hierbei in den einzelnen Arbeitsbereichen üblicherweise viele Entwurfsschritte bis zur allgemeinen Zufriedenheit rückgängig gemacht und von neuem getan werden, wobei alle Arbeitsbereiche voneinander abhängen und jede Änderung an einem einzelnen Teil an vielen anderen Teilen angemessen beantwortet werden muß, führt oft zu Verwirrung und mangelhaften Ergebnissen.

In solchen Situationen sind *BCT-integratoren* hilfreich: sie verfolgen Abhängigkeiten zwischen den einzelnen Arbeitsbereichen *(Browsing)*, zeigen inkonsistente Stellen an *(Checking)*, setzen notwendige Korrekturen u.U. sogar automatisch durch *(Transforming)*, bewahren so die Konsistenz des Entwurfes und bewältigen einen Großteil der Kommunikation zwischen den Arbeitsbereichen, mit der die Komplexitätsreduktion innerhalb dieser erkauft werden muß.

Das dem Werkzeug zu inkorporierende Integrationswissen stellt eine komplexe Relation der die Zwischenergebnisse der einzelnen Arbeitsbereiche speichernden Dokumente dar. Je feingranularer die einzelnen Dokumente in strukturelle Einheiten, die Inkremente, zerlegbar sind, desto präziser und reichhaltiger gelingt die Darstellung der Abhängigkeiten (Korrespondenzen) zwischen den jeweiligen Bereichen, desto besser wird auch die menschliche Arbeit durch das Integrationswerkzeug unterstützt.

Die Aufzählung der bisher bekannten Integrationswerkzeuge würde den Rahmen dieses Beitrages sprengen; ausführlichere Erörterungen finden sich in [1] [2]. Die meisten Ansätze erweisen sich hiernach als problemspezifisch und nur bedingt verallgemeinerbar.

Deklarative, (idealerweise sogar indeterministische) Spezifikation auf semantisch hohem Niveau wird nur selten unterstützt. So nutzt z.B. syntaxunabhängige Integration die Binnenstruktur der zu integrierenden Dokumente nicht aus, was bei nachrichtenverteilenden Integratoren [3] zu erheblichem Aufwand in den semantisch niedrigen Protokollschichten führen kann. Hingegen ist der Ansatz der Hypertextsysteme [4] aufgrund der Beliebigkeit der Verbindungen zu schwach, um strukturelle Regelmäßigkeiten zu forcieren.

Compilertechnische Integrationsmethoden werden z.B. in [5] beschrieben; die entsprechenden Werkzeuge repräsentieren die zu integrierenden Dokumente in einem Syntaxbaum und propagieren die zur Integration erforderliche Information durch dessen Zweige. Das Konzept ist deterministisch und total und deshalb für solche Entwicklungsaufgaben, bei denen der Benutzer zwischen verschiedenen Integrations- oder Transformationsalternativen zu wählen hat, nur mäßig geeignet; obendrein erweist sich totale Integration (per *"batch"*) dort als ineffizient, wo große Dokumente häufig, aber immer nur geringfügig verändert werden.

Mit der Entwicklung von Integrationswerkzeugen nach dem Prinzip *BCT* ist unser Lehrstuhl seit langem befaßt. Verlangt werden feingranulare Werkzeuge von hohem semantischen Niveau und inkrementeller Funktionalität, wozu sich der paargraphgrammatische Ansatz als geeignet erwiesen hat: mächtigere Datenstrukturen als Bäume sind allgemeine Graphen, deren Integrierbarkeit mit Hilfe von Paargrammatiken [6] längst bekannt ist. Die paargrammatische Methode [7] [8] ermöglicht partielle und indeterministische (benutzergesteuerte) Werkzeuge, deren praktische Bewährung in [1] umfassend beschrieben ist. Dort wird auch bereits angedeutet, daß die erforderlichen "dynamischen" Pro-

Zwei Axiome sind also zueinander nur dann ambivalent (mehrdeutig), wenn sie einen Schemaknoten gemeinsam besitzen und die beiden anderen Knoten in Verallgemeinerungsrelation zueinander stehen. Die Kollapsregeln werden mithin nicht wirksam, wenn die verschiedenen Quell- bzw. Zielknoten Geschwister sind oder auf sonstige Weise "nebeneinander" stehen. Auf diese Weise bleibt die freie Auswahl von Transformationsalternativen erhalten, die zur Lösung der meisten praxisrelevanten Integrationsprobleme erforderlich ist.

Ein stets terminierender *Abschlußalgorithmus* stellt durch sukzessive Anwendung dieser Metaregeln aus einer ursprünglich möglicherweise widersprüchlichen Konstellation von Schemakorrespondenzaxiomen eine widerspruchsfreie Abschlußkonstellation eindeutig her. Da die Korrespondenzspezifikation in Abb.8 in sich konsistent ist, also weder Widersprüche noch Ambivalenzen enthält, zeigt Abb.9 die Wirkung des Abschlußalgorithmus an einem anderen, abstrakten Beispiel: aus einem Widerspruch erzeugt die Verbandregel zunächst mehrere Ambivalenzen, welche dann von den Kollapsregeln so radikal zerstört werden, daß sich der ursprüngliche Widerspruch ebenfalls mit auflöst.

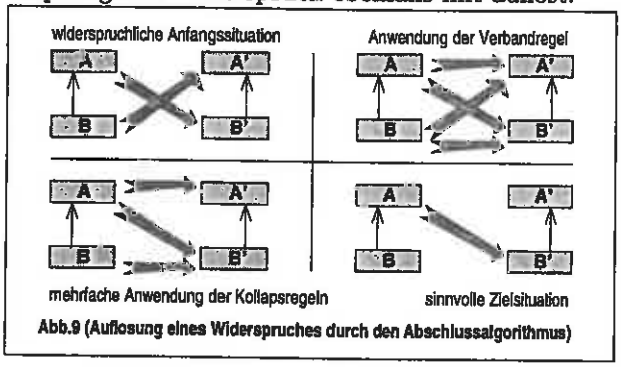


Abb.9 (Auflosung eines Widerspruches durch den Abschlussalgorithmus)

Man beachte, wie die *asymmetrischen* Kollapsregeln die Auflösung des ursprünglich symmetrischen Widerspruches in eine bestimmte Richtung lenken: "manuell" wäre der Widerspruch in Abb.9 nämlich auch durch Entfernung des Axioms A⟹B' lösbar; die automatische Entfernung von Axiom B⟹A' erzwingt aber größtmögliche Präzision der Korrespondenzspezifikation durch größtmögliche Spezialisierung im Zielschema.

Nach Termination des Abschlußalgorithmus sind die verbliebenen Korrespondenzaxiome zwischen zwei Graphenschemata zueinander konsistent, sodaß nun die gemeinsame Semantik mehrerer Axiome festlegbar ist.

Als gemeinsame Semantik *zweier* Axiome gilt ihre *Disjunktion*, falls sie einen gemeinsamen Quell- oder Zielknoten besitzen, andernfalls ihre *Konjunktion*, jeweils bezüglich der Extensionen der entsprechenden Schemaknoten. Disjungierte, also alternativ geltende Axiome werden zu *Pseudoaxiomen* gebündelt und diese wie einzelne Axiome behandelt, die wiederum eine gemeinsame Semantik mit anderen Axiomen und Pseudoaxiomen besitzen. So wird nicht nur die oben besprochene indeterministische Auswahl ermöglicht, sondern auch die Befolgung mehrerer Transformationsbedingungen, die zugleich gelten sollen, erzwungen. Die drei Axiome aus Abb.8 bilden zusammen zwei überlappungsfreie Pseudoaxiome. Eine Spezifikation mit drei sich überlappenden Pseudoaxiomen, die zugleich befolgt werden müssen, zeigt Abb.10:

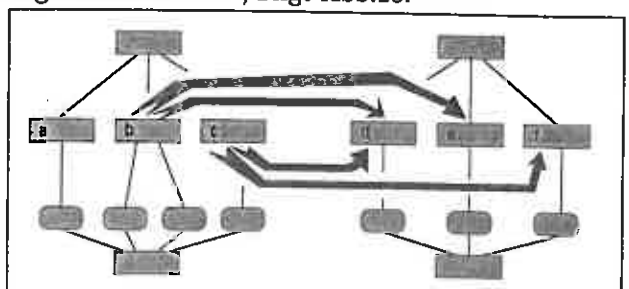


Abb.10 (vier Axiome, die zusammen drei Pseudoaxiome - an b, c und d - bilden)

Eine Integrationsspezifikation gilt schließlich als *korrekt*, wenn die dynamischen Transformationsregeln, welche aus den Graphproduktionen zusammengesetzt werden, nicht im Widerspruch zu den statischen Schemakorrespondenzaxiomen stehen, (was bei dem Beispiel in Abschnitt 2 trivialerweise erfüllt ist). Zum Vergleich von Schemakorrespondenzen und Integrationsregeln dient eine Funktion, die Korrespondenzen und Integrationsregeln auf spezielle Mengen abbildet.

4 Ausblick

Um die neue Methode praktikabel zu machen, soll zukünftig ein *Editorwerkzeug* die Arbeit mit Schemakorrespondenzen in einer bereits bestehenden Entwicklungsumgebung ermöglichen. Die Methode selbst soll anhand eines Integrationsproblems aus der chemischen Verfahrensmodellierung validiert und weiterentwickelt werden. Hierbei gilt es Komponenten eines Anlagenmodells mit Differentialgleichungen konsistent zu halten, welche die chemischen Reaktionen in der Anlage und ihren Komponenten beschreiben [10].

Literatur

[1] M.Nagl, *Building tightly integrated software development environments: the IPSEN approach*, LNCS 1170, Springer-Verlag, Berlin 1996

[2] S.Gruner, *Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge*, Aachener Informatikberichte 97-5, RWTH Aachen, 1997

[3] S.Reiss, *Interacting with the FIELD environment*, Softw. Pract. a. Exp. 20, 89-115, 1990

[4] J.Conklin, *Hypertext: an introduction and survey*, IEEE Computer (September 1987), 17-41

[5] A.Habermann/D.Notkin, *Gandalf: software development environments*, Transactions on Softw. Eng. 12(12), 1117-1127, 1986

[6] T.Pratt, *Pair Grammars, graph languages and string-to-graph translations*, Journ. of Comp. a. Sys. Sc. 5, pp 560-595, 1971

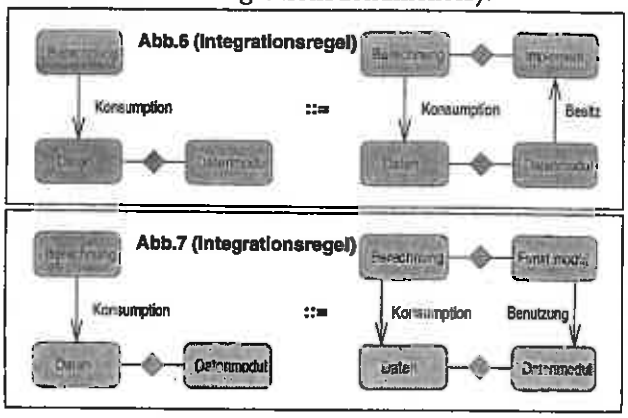
[7] A.Schürr, *Operationales Spezifizieren mit programmierten Graphersetzungssystemen*, Deutscher Universitätsverlag, Wiesbaden 1991

[8] M.Lefering, *Integrationswerkzeuge in einer Softwareentwicklungsumgebung*, Verlag Shaker, Aachen 1995

[9] M.Nagl, *Softwaretechnik: Methodisches Programmieren im Großen*, Springer-Verlag, Berlin 1990

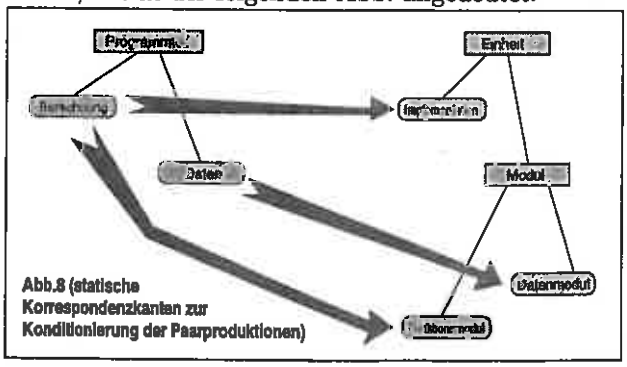
[10] R.Bogusch et al., *Computer aided process modeling with Modkit*, Proc. Chemputers Eur.III, Frankfurt 1996

kument einrichten, desweiteren identische Transformationen, welche lediglich das Vorkommen gewisser Situationen in der Konstellation der Dokumente prüfen, schließlich Transformationen, die aus dem Quelldokument das Zieldokument samt Integrationsdokument erzeugen. Abb.6,7 zeigen Produktionen zur Erzeugung von Knoten im Zieldokument für jeweils eine Berechnung im Quelldokument (einschließlich der Korrespondenzknoten des Integrationsdokumentes):



Die Regeln besitzen dieselbe linke Seite, sind in der entsprechenden Situation also beide anwendbar[1].

Die Graphenschemata konditionieren nur die Gestalt der zugehörigen Dokumentgraphen, enthalten aber selbst noch keine Integrationsinformation; diese liegt erst in den Definitionen der Paargrammatik. Hier setzt die oben angekündigte neue Spezifikationsmethode an: Es ist nämlich möglich, die in Abb.6,7 durch die Korrespondenzknoten dargestellten Integrationsbeziehungen bereits in den Graphenschemata zu konditionieren, wie in der folgenden Abb. angedeutet:



Die Pfeile stellen statische *Integrationsbedingungen* zwischen den entsprechenden Knoten der beiden Schemata dar, im Gegensatz zu den dynamischen Integrationsregeln, die für die Dokumentgraphen gelten. Zweck und Bedeutung dieser Integrationsbedingungen sind das Thema des folgenden Abschnittes.

[1] Hierin gründet der in der Einführung beschriebene Nichtdeterminismus, der dem Benutzer des späteren Werkzeuges an wesentlichen Entscheidungsstellen die Auswahl zwischen möglichen Alternativen überläßt.

3 Schemakorrespondenzen

Spezifikationen adäquater Integrationswerkzeuge für *"real world applications"* werden im Prinzip so entwickelt wie im vorigen Abschnitt beschrieben, sind aber sowohl wesentlich komplizierter im Detail als auch wesentlich größer nach Graphenschemata und Anzahl der Integrationsregeln. Entsprechend schwierig ist die Aufgabe, solche Spezifikationen korrekt und konsistent zu formulieren. Die Schemakorrespondenzmethode erleichtert die erforderlichen Konsistenzanalysen in jedem Fall und unterstützt günstigenfalls sogar die automatische Generierung von Integrationsregeln aus den Paarungsdefinitionen der beteiligten Graphgrammatiken. Die narrative Darstellung der Thematik in diesem Beitrag wird durch ausführlichere und formalere Erläuterungen in [2] ergänzt.

Integrationsbedingungen, wie in Abb.8 dargestellt, heißen *Schemakorrespondenzaxiome*; "Axiome" deshalb, weil sie vom Spezifikator eines Integrationswerkzeuges gesetzt werden und sich alle weiteren Spezifikationsschritte danach richten müssen.

Ein Korrespondenzaxiom hat die Bedeutung, daß eine Instanz aus dem Quellknoten des Pfeiles in eine Instanz des Zielknotens des Pfeiles überführt werden soll. Die gemeinsame Bedeutung mehrerer Korrespondenzaxiome ist schwieriger zu definieren, da alle Konstellationen von Schemakorrespondenzaxiomen bestimmten Bedingungen genügen müssen, um einander nicht zu widersprechen. Zwei Korrespondenzaxiome sind zueinander widersprüchlich, wenn ihre jeweiligen Quellknoten in anderer Ordnung bezüglich der Verallgemeinerungsrelation stehen als die jeweiligen Zielknoten.

So entstehen *Metaregeln* für Schemakorrespondenzen, die deren Widerspruchsfreiheit gewährleisten, aber nicht mit den Integrationsregeln im grammatischen Spezifikationsteil verwechselt werden dürfen. Eine *Verbandregel* ist für die Suche von Widersprüchen, zwei *Kollapsregeln* sind für deren Auflösung zuständig.

Da Graphenschemata *Verbände*[2] darstellen müssen, ist folgende Korrespondenzaxiomenersetzung, nämlich die Verbandregel, in sich widerspruchsfrei:

Falls A, B bzw. A', B' Quell- bzw. Zielknoten zweier Korrespondenzaxiome sind, füge Axiome C ⟹ C' oder D ⟹ D' hinzu, wo C (bzw. C') kleinste gemeinsame Oberklasse von A und B (bzw. A' und B'), analog D (bzw. D') größte gemeinsame Unterklasse.

Da die Anwendungsbedingung der Verbandregel selbst keine widersprüchliche Situation darstellt, wird die Verbandregel nur dort angewandt, wo sie die Anwendbarkeit einer der folgenden *ambivalenzlösenden* Axiomersetzungen, der Kollapsregeln, vorbereitet:

Falls eine Knotenklasse A Quelle zweier Axiome ist, so lösche dasjenige mit der allgemeineren Zielklasse. Ist eine Knotenklasse B' Ziel zweier Axiome, so lösche dasjenige mit der spezielleren Quellklasse.

[2] Abb.8 verzichtet auf die hier irrelevanten kleinsten gemeinsamen Unterklassen "⊥".

duktionenkorrespondenzen durch "statische" Schemakorrespondenzen vorspezifizierbar sind.

Der hier vorgestellte Beitrag, ausführlicher kommentiert in [2], beschreibt erste Schritte der Entwicklung dieses statischen Korrespondenzmodells zur Verbesserung der bisherigen Methode.

2 Integrationsmethode

Die Methode der paargraphgrammatischen Spezifikation von Integrationswerkzeugen hat sich in der Praxis bewährt und bildet die Grundlage der oben erwähnten verbessernden Erweiterung. Dieser Abschnitt vermittelt deshalb schrittweise an einem durchaus praxisrelevanten, aus didaktischen Gründen aber stark vereinfachten Beispiel erst die nötige Vorstellung des paargrammatischen Ansatzes, damit dann unten die Schemakorrespondenzmethode als der eigentliche neue Beitrag dieses Aufsatzes eingeführt werden kann.

Beispiel: *Aus einem großen, nicht dokumentierten Programm der veralteten Programmiersprache Cobol soll eine abstrakte Architekturbeschreibung, welche das Fundament einer Reimplementation des alten Programmes in einer zeitgemäßen Programmiersprache bildet, abgeleitet werden.*

a) Zunächst wird ein *Graphenschema* für Cobolprogramme gebildet, welches besagt, daß diese wesentlich aus Berechnungen und Daten, welche durch Zeilennummern eindeutig bestimmte Programmteile darstellen, bestehen:

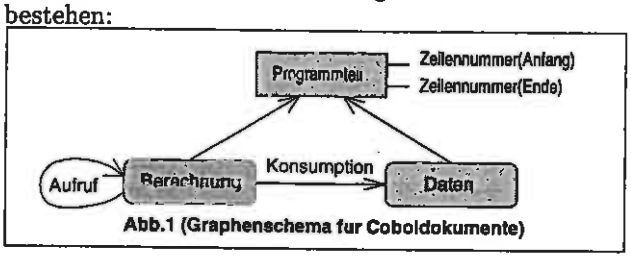
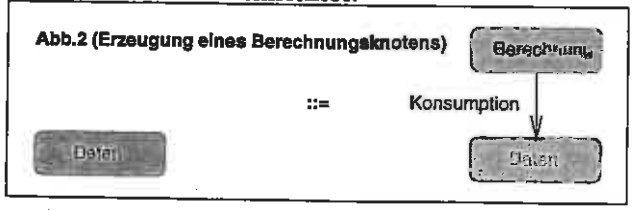


Abb.1 (Graphenschema fur Coboldokumente)

In diesem Formalismus beschreiben Rundecke die Grundtypen, Rechtecke deren sortierende Klassen, beschriftete Pfeile verschiedentliche Beziehungen zwischen Klassen oder Typen, schließlich unbeschriftete Pfeile die besondere Verallgemeinerungsrelation.

b) Ist das Schema definiert, wird es durch *Produktionen* zur *Graphgrammatik*, die alle durch das Graphenschema charakterisierten Dokumentgraphen erzeugen kann, vervollständigt. Abb.2 zeigt z.B. eine Produktion, die zu einem schon vorhandenen Programmteil vom Typ Daten eine Berechnung erzeugt und die vom Graphenschema vorgeschriebene Konsumptionsrelation zwischen beiden einrichtet:



c) Analog müssen nun die Charakteristika der Architektursprache [9] als Graphenschema dargestellt werden. Deren Dokumente bestehen aus Einheiten, die einen eindeutigen Namen tragen und entweder Module, oder deren Implementationen darstellen. Module dürfen einander benutzen und werden in Datenmodule und Funktionsmodule unterschieden. Jener Beschreibung entspricht dieses Graphenschema:

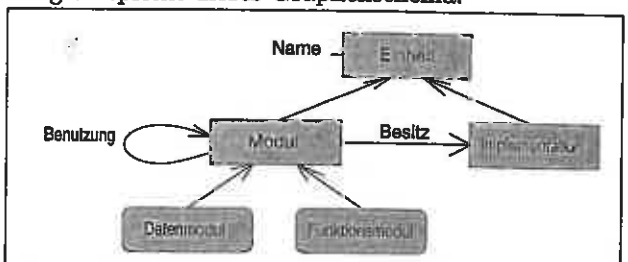
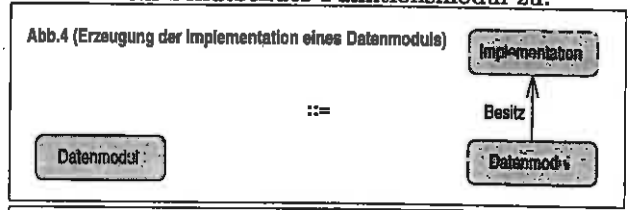
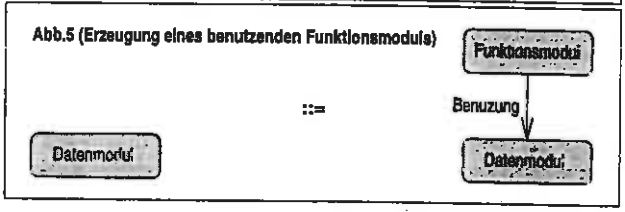


Abb.3 (Graphenschema fur Dokumente der Architektursprache MIL)

d) Folgende Produktionen der zugehörigen Graphgrammatik fügen einem Datenmodul eine Implementation bzw. ein benutzendes Funktionsmodul zu:





e) Im nächsten Entwurfsschritt werden die Graphgrammatiken der Architektur- und Coboldokumentgraphen zu einer *Paargrammatik* verschmolzen, indem man jeweils einzelne Produktionen der Quell- und Zieldokumentgrammatik als zusammengehörig definiert: wird das Quelldokument mittels einer Quellproduktion p verändert, muß auch das Zieldokument, um die Konsistenz der Gesamtkonfiguration zu wahren, einer Zielproduktion p' unterworfen werden, welche per Korrespondenzdefinitionem mit p verbunden ist. Paargrammatiken sind prinzipiell symmetrisch, legen also nicht fest, welche ihrer beiden Seiten Quell- und welche Zielseite sein soll. Dies wird erst später bei der Bestimmung der Werkzeugfunktionalität unterschieden.

In diesem Beispiel können Berechnungen sowohl expizit einem Funktionsmodul entsprechen, als auch in der Implementation eines Datenmoduls verborgen werden. Daher muß Produktion 2 sowohl mit Produktion 4 als auch mit Produktion 5 verknüpft werden, wodurch zwei alternative Produktionen der entsprechenden Paargrammatik entstehen.

f) Aus diesen Paarproduktionen werden im letzten Schritt der Spezifikation die *Integrationsregeln* des entstehenden Integrationswerkzeuges abgeleitet. Dazu gibt es im allgemeinen verschiedene Möglichkeiten. Sinnvoll sind Regeln, welche zwischen schon vorhandenen Quell- und Zieldokumenten ein Integrationsdo-