Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

— mit 15 Abbildungen —

Stefan Gruner

Graduiertenkolleg für Informatik und Technik
und Lehrstuhl für Informatik III (Softwaretechnik)
der Rheinisch-Westfälischen Technischen Hochschule,
Ahornstraße 55 / (D)-52056 Aachen
stefan@i3.informatik.rwth-aachen.de

Zusammenfassung. Dieser Arbeitsbericht ist ein Beitrag zum interdisziplinären Sonderforschungsbereich Nr. 476 zur informatischen Unterstützung arbeitsbereichsübergreifender Entwicklungsprozesse in der Verfahrenstechnik. Die Notwendigkeit, verschiedene, voneinander abhängige Dokumente einer Entwicklungskonfiguration, z.B. in der (informatischen) Software- oder der (ingenieurwissenschaftlichen) Verfahrenstechnik, miteinander konsistent zu halten, bezeichnen wir als Integrationsproblem und finden in der Literatur bereits einige Lösungsansätze dazu, welche in dieser Schrift kurz dargestellt und weiterentwickelt werden.

I.

In einer alten, noch mit der Schreibmaschine getippten Arbeit [1] finden wir ein originelles Zitat, welches uns auf aus heutiger Sicht amüsante, nichtsdestoweniger lehrreiche Weise an unsere Thematik führt:

> "Als neues Peripheriegerät hat sich das Bildschirmgerät durchgesetzt. Eine Verwendung als Eingabegerät liegt nahe, womit sich die Frage nach höheren, bequem zu handhabenden Programmiersystemen stellt. Es hat inzwischen eine Reihe von Ansätzen gegeben, den Bildschirm wegen seiner hohen Geschwindigkeit bei der Ausgabe größerer Datenmengen in interaktiven Systemen einzusetzen."

Nicht nur haben die vergangenen achtundzwanzig Jahre, seit das einleitende Zitat erstmalig veröffentlicht wurde, dieses ganz und gar bestätigt, sondern es sind, darüber hinaus, unsere Bildschirmgeräte den an sie gestellten Anforderungen mittlerweile kaum noch gewachsen, wodurch einige Probleme, welche in dem vorliegenden Beitrag zu besprechen sind, aufgeworfen werden. Der mit der Einführung des Bildschirmes einhergehende relative Bedeutungsschwund des Computers als Symbol- und Rechenmaschine zugunsten seiner zunehmenden Verwendung als Bild- und Darstellungsmaschine — der Mensch ist ja ein "Augentier"— ist mittlerweile so weit fortgeschritten, daß jeder Bildschirm zu klein wäre, um ein praxisrelevantes Problemmodell als ganzes graphisch abzubilden. Die Benutzer einer solchen Maschine fühlen sich dadurch genötigt, die Gesamtmodelle der jeweils zu behandelnden Problematiken in sogenannte Partialmodelle zu teilen und in kleinen Ausschnitten darstellen sowie bearbeiten zu lassen. Das oben in der Zusammenfassung genannte Integrationsproblem besteht dann (in unserem Kontext) in dem augenscheinlichen Verlust der konzeptionellen Verbindungen zwischen den einzelnen Darstellungseinheiten, insbesondere nach Einführung der Ford'schen Arbeitsteilung auch in den Bereichen geistiger Tätigkeit, wie z.B. Entwurf, Modellierung, Programmierung. [1]

Zur Verdeutlichung eines Integrationsproblems diene das folgende Beispiel, welches einen sehr kleinen Ausschnitt einer mehrgeteilten verfahrenstechnischen Modellierung einer Blasensäule zur Herstellung eines chemischen Stoffes nach der zusammenfassend in [2], ausführlicher in [3] beschriebenen Datenmodellierungstechnik vorführt. Die **Abb.2** stellt einen Ausschnitt des strukturellen Partialmodells der Blasensäule dar, wie vom Lehrstuhl für Prozeßtechnik der RWTH im Rahmen der Kooperation in einem Sonderforschungsbereich [4] für uns skizziert. Wir sehen ein Bauteil cell-1, bestehend aus flüssigen bzw. blasenartigen Komponenten liquid-1 und vapor-1, welche einander an einem Doppelfilm two-film-1 berühren und verschiedene, nach FILM oder VALVE (Ventil) unterschiedene Schnittstellen zu ihrer jeweiligen Außenwelt bereitstellen, wobei die äußeren Schnittstellen des zusammengesetzten Objektes als referentielle Kopien der freien Schnittstellen seiner Bestandteile modelliert werden. Weitere Einzelheiten spielen an dieser Stelle keine Rolle, da wir uns nur mit der Form, nicht aber

[1] Es mag dem ein oder anderen Leser vielleicht merkwürdig erscheinen, daß der Verlust der Wahrnehmung von Beziehungen zwischen Dingen oder Personen in der Sprache der Philosophie und Soziologie im allgemeinen Entfremdung genannt wird. So betrachtet wäre unsere Integrationsproblematik eine technische Variante dieses Phänomens — dies aber nur ganz am Rande bemerkt.

mit dem (chemietechnischen) Inhalt solcher Strukturen befassen. Eine Skizze der Blasensäule selbst ist in **Abb.1** gezeigt, damit die Herkunft der in Abb.2 verwendeten Begriffe verständlich ist.

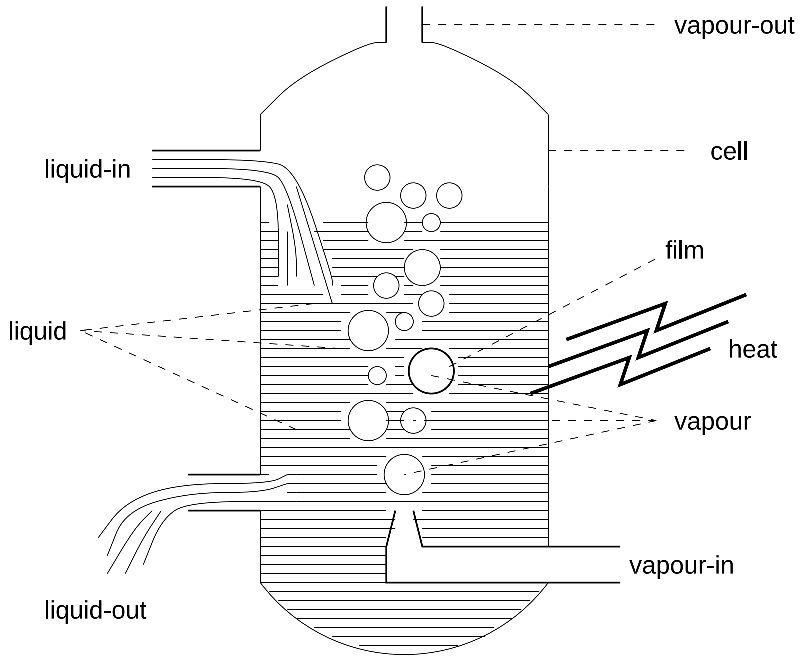

Abb.1

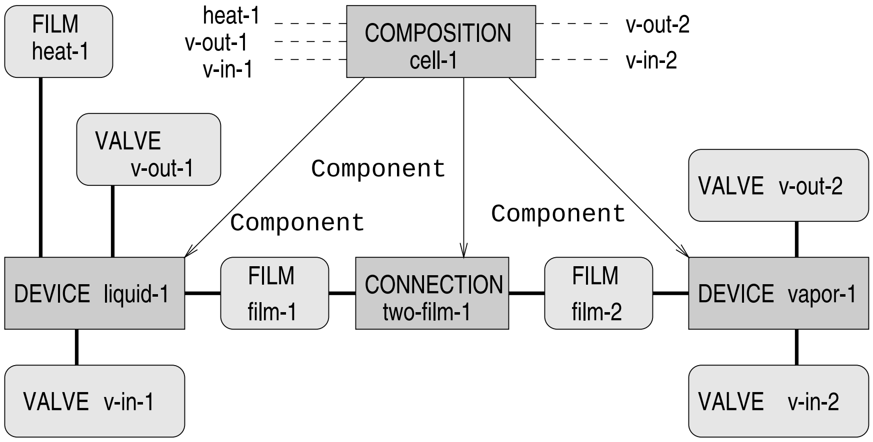

Abb.2

Ein Ausschnitt eines anderen Partialmodells der Blasensäule ist in der **Abb.3** zu sehen, welche den materiellen Gehalt einiger struktureller Komponenten erfaßt.[2] Vier singuläre Phasensysteme, nämlich phase-1, phase-2, phase-3 und phase-4, enthalten jeweils die gleiche Mischung liquid-products-1, deren stoffliche Anteile SO32$^-$, NS3O93$^-$, H2O, HADS, NO2$^-$ und HSO3$^-$ zwei verschiedenen, in phase-3 stattfindenden Reaktionen reaction-1 und reaction-2 unterworfen sind.

[2] Die diesem Partialmodell offenbar stillschweigend vorausgesetzte Vermutung einer materiellen Beschaffenheit unserer Wirklichkeit teilen wir im wesentlichen übereinstimmend mit der in [5] dazu vorgetragenen Argumentation.

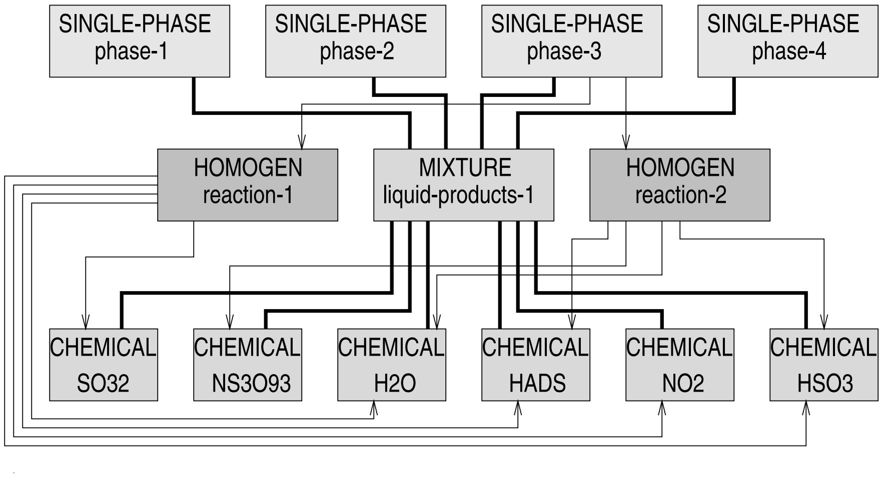

Abb.3

Beide Partialmodelle für Struktur und Material der Blasensäule zeigen keine Information über mögliche Zusammenhänge mit dem jeweils anderen Partialmodell, obgleich solche Zusammenhänge selbstverständlich bestehen, da ein Inhalt ohne Beinhaltendes gar nicht denkbar und eine Struktur ohne Inhalt nicht sonderlich brauchbar ist. In **Abb.4** wird die Grenze zwischen den beiden Partialmodellen überwunden und durch die dicken Pfeile angezeigt, wie Abb.2 und Abb.3 korrespondieren: das Reaktionssystem phase-3 entspricht der Flüssigkeitskomponente liquid-1, an deren Schnittstellen genau das gleiche Stoffgemisch anzutreffen ist, wie in der Komponente selbst (mit Ausnahme der Schnittstelle heat-1 für den hier nicht stofflich modellierten Wärmeaustausch. Die meisten anderen Komponenten aus Abb.2 und Abb.3 sind in Abb.4 ausgeblendet, um den Blick des Betrachters nicht von den an dieser Stelle wesentlichen Korrespondenzen zwischen den beiden Partialmodellen abzulenken.)

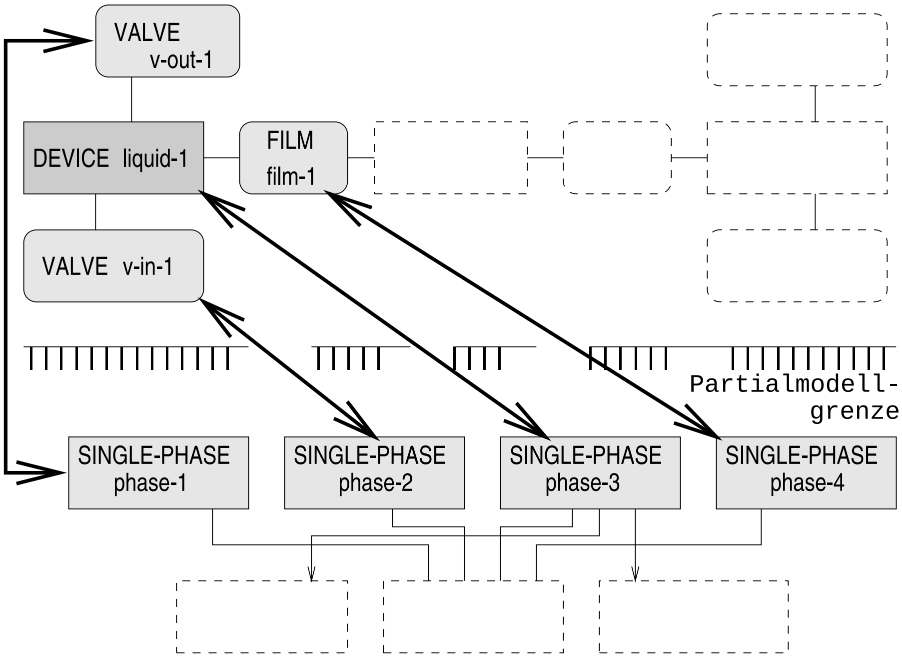

Abb.4

Wir haben bereits in [6][7] dargelegt, daß die Änderungskontrollierung als Schlüsselproblem im arbeitsteiligen Entwurf aufzufassen ist, da jede Änderung in einem Partialmodell nicht allein dortselbst wirksam wird, sondern vor allem die anderen, korrespondierenden Partialmodelle betrifft, welche dann entsprechend nachbearbeitet werden müssen, damit die Konsistenz der Gesamtkonfiguration aller beteiligten Partialmodelle bewahrt bleibt. Würde zum Beispiel in Abb.4 das Bauteil liquid-1 durch irgend eine andere Komponente aus dem Vorrat prozeßtechnischer Modellierungsmittel ersetzt, so wäre sicher auch phase-3 hinfällig, vielleicht sogar noch ihre benachbarten Komponenten. Die Übermittlung von Änderungsinformation aus einem Partialmodell in ein anderes, korrespondierendes Partialmodell sowie die dortige korrekte Verarbeitung dieser Information ist so wichtig und, bei hinreichend großen Modellen, so unanschaulich, daß Werkzeugprogramme[3], im folgenden Integrationswerkzeuge oder Integratoren genannt, den Menschen in dieser Aufgabe unterstützen sollen. Obwohl —oder: weil (?)— unsere Integrationsprobleme, da sie nicht alle drei der allgemein bekannten Definitionskriterien gutgestellter Probleme[4] erfüllen, der Klasse der schlechtgestellten Probleme angehören, finden wir in der Literatur bereits eine Fülle methodischer Ansätze zur Entwicklung von Integrationswerkzeugen, welche im folgenden Abschnitt besprochen werden.

II.

Da wir in der Literatur bereits ausführliche Vergleichungen bisheriger Ansätze zur Integration von Datenmodellen (bzw. Entwicklungsdokumenten im allgemeinen) vorfinden [9][10], fassen wir uns an dieser Stelle kurz, um in dem nachfolgenden Absatz III lediglich diejenigen Methoden ausführlicher zu besprechen, auf denen der später in diesem Beitrag vorgestellte erweiterte Ansatz beruht.

Eine weitverbreitete Integrationsmethode besteht in der "manuellen", also unspezifizierten Implementierung automatischer Formatkonvertierer [11], die ähnlich wie einfache Programmübersetzer (Compiler) funktionieren. Nachteil dieser Werkzeuge ist ihre nichtinkrementelle Arbeitsweise, die bei jeder Transformation eines schon bestehenden Quelldokumentes das möglicherweise ebenfalls schon bestehende Zieldokument nicht nur lokal und soweit als nötig verändert, sondern völlig zerstört und neu konstruiert, sodaß bereits getroffene Entscheidungen und Absichten der Entwickler der bearbeiteten Modelldokumente unter gewissen Umständen durch das Integrationswerkzeug eher behindert als gefördert werden. Hingegen sind automatische Konvertierer gar nicht anwendbar, wenn von der Menge aller Quelldokumente zur Menge aller Zieldokumente des Integrationswerkzeuges keine Injektion, welche durch einen Konvertierer verkörperbar wäre, bekannt ist. (Übrigens ist auch der vorliegende Aufsatz mit Hilfe von Formatkonvertierern entstanden: das Manuskript wurde aus LaTeX nach dvi und von da nach Postscript transformiert.) In der Industrie setzt sich derzeit, zusammen mit der Modellierungssprache EXPRESS, ein Standard namens STEP [12] zum einheitlichen Dokumentenaustausch durch, welcher zwar die Darstellung interdokumentärer Integritätsbedingungen vorsieht, jedoch nicht solcherart, daß daraus schon die Funktionalität eines entsprechenden Integrationswerkzeuges ableitbar wäre. Da unser Ziel jedoch nicht allein im Bau sol-

[3] "Denkzeuge" in der Sprechweise von [8]

[4] unbedingte Lösbarkeit, eindeutige Lösbarkeit, sowie stabile Lösbarkeit rücksichtlich des Verhältnisses von Eingabe- zu Ausgabeschwankungen

cher Werkzeuge ad hoc, sondern in deren methodischer Spezifikation besteht, kommen in EXPRESS formulierte Partialmodelle nicht als Ausgangspunkt einer entsprechenden Methodik in Frage. Hypertextsysteme, wie in [13] vorgestellt, erlauben zwar die Definition fein detailierter Korrespondenzen von Inkrement zu Inkrement —darunter verstehen wir die kleinsten sinnvollen Einheiten eines Dokuments, den Lexemen und Graphemen aus der Sprachentheorie vergleichbar— der an einer Gesamtkonfiguration beteiligten Partialmodelle, mehr aber auch nicht, d.h.: die Verbindungskanten eines solchen Systems sind ganz beliebig und repräsentieren keine strukturellen Gesetzmäßigkeiten über notwendige und hinreichende Konsistenzbedingungen der Konfiguration. Besser zur formalen Spezifikation von Integrationswerkzeugen sind Ansätze mit attributierten Syntaxbäumen und Grammatiken geeignet, die wir bereits recht früh in [14][15][16][17][18] sowie seit neuerer Zeit in [19] finden. In attributierten Syntaxbäumen sind Attributwertänderungen durch deren Zweige propagierbar. Dies gestattet die Spezifikation und Generierung konsistenzprüfender Analysewerkzeuge, falls alle beteiligten Dokumente oder Partialmodelle als Teilbäume eines gemeinsamen Syntaxbaumes darstellbar sind, wobei spezielle Tunnelattribute auch die Distribution solcher Strukturen ermöglichen. Obwohl die den oben erwähnten automatischen Konvertierern ähnliche, unidirektionale und unflexible Funktionsweise solcherart generierter Werkzeuge problematisch bleibt, erachten wir das Prinzip dieser Ansätze für richtungweisend, insbesondere dann, wenn es mit den Ideen zur Schemamigration und -integration aus dem Bereich der Datenbanken kombiniert wird [20], da dort mit ähnlichen konzeptuellen Schwierigkeiten wie bei dem in Absatz I erwähnten Integrationsproblem gekämpft wird.

III.

Der eigentliche Ausgangspunkt unserer Überlegungen wird von den in diesem Absatz besprochenen Ansätzen zur Entwicklung von Integratoren bestimmt, welche alle mindestens eines der folgenden drei Charakteristika aufweisen: (i) attributierte Graphen als einheitliche Datenstruktur (vgl. die attributierten Syntaxbäume aus Absatz II), (ii) Klassifikation der zu integrierenden Modellinkremente nach Ober- und Unterbegriffen in einem Schema (ähnlich den allgemein bekannten ER-diagrammen konzeptueller Datenbankschemata), sowie (iii) grammatische Paarbildungen zur Definition einer operationalen Semantik des entsprechenden Integrationswerkzeuges.

In seiner Dissertation [21] beschreibt *B. Westfechtel* ein paargrammatisch spezifiziertes Werkzeug zur Integration von Dokumenten des Softwaredesigns und (in der Programmiersprache Modula-2 notierten) Modulschnittstellen in einer Softwareentwicklungsumgebung, welchem implizit ein allgemeines, folgendermaßen formuliertes Prinzip zugrundeliegt.

Definition III.1 (paargrammatisches Korrespondenzprinzip)
1) *Seien P, P' Grammatiken mit Produktionen $p_1, \dots p_n$, bzw. $p'_1, \dots p'_m$ mit $n, m > 0$. Es bezeichnen darin γ bzw. γ' die jeweiligen Startsymbole; außerdem notiert $S \xrightarrow{x} s$ die Ableitung eines Satzes s aus einer Satzform S mittels einer endlichen Produktionenfolge x. Ferner sei $C = \{(p_i, p'_j) \mid p_i \in P, p'_j \in P'\}$ eine beliebige Relation zwischen P und P', und wir notieren $p_i C p'_j$, falls $(p_i, p'_j) \in C$. Seien schließlich $\mathcal{L}(P)$ und $\mathcal{L}(P')$ die von P und P' erzeugten Sprachen. Dann heißen zwei Sätze $l \in \mathcal{L}(P)$ und $l' \in \mathcal{L}(P')$*

konsistent in C, genau dann wenn gilt:

$\exists x : \gamma \xrightarrow{x} l$ *und* $\exists y : \gamma' \xrightarrow{y} l'$ *mit*

$x = p_{(1)}; p_{(2)}; \ldots p_{(k)}$, *wobei* $p_{(i)} \in P$ $(i = 1, \ldots k)$ *und*

$y = p'_{(1)}; p'_{(2)}; \ldots p'_{(k)}$, *wobei* $p'_{(i)} \in P'$ $(i = 1, \ldots k)$, *sodaß gilt:*

$p_{(1)} C p'_{(1)},\ p_{(2)} C p'_{(2)},\ \ldots\ p_{(k)} C p'_{(k)}$

und wir notieren dies mit $l =_C l'$.

2) *Für zwei Sprachen* $\mathcal{L}(P)$, $\mathcal{L}(P')$ *definieren wir bezüglich* C *drei Relationen wie folgt:*

$\mathcal{L}(P) <_C \mathcal{L}(P')$ *wenn gilt:* $\forall l \in \mathcal{L}(P)\ \exists! l' \in \mathcal{L}(P') : l =_C l'$ *(schwache Quellkonsistenz).*

$\mathcal{L}(P) >_C \mathcal{L}(P')$ *wenn gilt:* $\forall l' \in \mathcal{L}(P')\ \exists! l \in \mathcal{L}(P) : l =_C l'$ *(schwache Zielkonsistenz).*

$\mathcal{L}(P) \equiv_C \mathcal{L}(P')$ *wenn gilt:* $\mathcal{L}(P) >_C \mathcal{L}(P')$ *und* $\mathcal{L}(P) <_C \mathcal{L}(P')$ *(starke Konsistenz)* •

Nota bene: Die Formulierung dieses Prinzips beinhaltet noch keine Vorschrift darüber, an welcher von eventuell mehreren möglichen Stellen in den jeweiligen Satzformen zwei korrespondierende Produktionen anzuwenden sind! Erst durch eine solche Vorschrift wird die Definition des paargrammatischen Korrespondenzprinzips zur eigentlichen Definition der paargrammatischen Korrespondenz selbst. Auf diese Frage des Applikationskontextes kommen wir an späterer Stelle in diesem Beitrag noch präzisierend und mit Beispielen zu sprechen. Man beachte in der Definition außerdem, daß im allgemeinen p_i und $p_{(i)}$ beziehungsweise p'_j und $p'_{(j)}$ nicht die selbe Produktion aus P beziehungsweise P' bezeichnen. Der zweite Teil dieser Definition ist folgendermaßen motiviert. Gilt für zwei Sprachen keine Korrespondenzrelation, möglicherweise aber für einzelne Sätze dieser Sprachen so wie im ersten Teil definiert, dann übt ein entsprechendes Integrationswerkzeug die Funktion eines Analysators aus, welcher dem Benutzer entdeckte Inkonsistenzen in der Gesamtkonfiguration mitteilt, die dieser dann selbst zu korrigieren hat. Bilden die Satzpaare einer Konsistenzrelation über zwei Sprachen den Funktionsgraphen einer bijektiven Funktion, so gibt es auch einen automatischen Vorwärts- und Rückwärtstransformator zwischen den beiden Sprachen. Bei bloßer Injektivität ist automatische Transformation nur in einer Richtung möglich. Ist eine solche Korrespondenz zwischen Sprachen nicht einmal injektiv, sind nur halbautomatische Transformationswerkzeuge möglich, die an indeterministischen Auswahlstellen den Werkzeugbenutzer um ein "Orakel" fragen müssen. Auf diese Weise können wir die in der bereits zitierten Literatur vorgefundene Unterscheidung zwischen halbautomatischen und vollautomatischen Integratoren auf ein theoretisches Fundament stellen.

In der Dissertation von *Th. Janning* [22] finden wir die Beschreibung eines Werkzeuges zur Integration von in der Sprache der strukturierten Analyse notierten Anforderungsdefinitionen an ein Softwaresystem und einer (von mehreren möglichen) korrespondierenden Darstellung von dessen modularer Architektur, welchem die im folgenden erläuterte Spezifikationsmethode zugrundeliegt. Zunächst werden alle kleinsten sinnvollen Einheiten der Quellsprache —dort: die strukturierte Analysesprache— aufgezählt und nach gemeinsamen Eigenschaften erforscht, welche dann durch Oberbegriffe klassifizierend dargestellt werden. Ein solches Modell nennen wir ein Schema der dadurch beschriebenen Sprache. So wie die Analyse der Quellsprache in einem Quellschema dieser Sprache resultiert, wird nun auch ein Zielschema über der zu integrierenden Zielsprache errichtet. Ganz analog werden dann diese beiden Schemata wiederum nach semantischen Ähnlichkeiten durchsucht, welche wiederum durch Oberbegriffe klassifizierend dargestellt werden. Dieses Konstrukt nennt Janning das Metamodell der Integration. Ein Inkrement der Quellsprache und ein Inkrement der Zielsprache —wobei "Quelle" und "Ziel" u.U. vertauschbar sind, weshalb wir eigentlich nur von "linker" bzw. "rechter Seite" sprechen sollten— sind nach dieser Spezifikationsmethode genau dann integrierbar, wenn ein Pfad vom Quellinkrement durch das zugehörige Quellschema, weiter durch das "darüber" befindliche Metamodell, dann durch das Zielschema und schließlich bis zu diesem Zielinkrement gefunden werden kann, wobei diese statische Integrationsspezifikation jedoch nichts darüber aussagt, auf welche Weise ein entsprechendes Werkzeug diese Funktionalität implementieren soll, (im Gegensatz zu der dynamischen Integrationsspezifikation durch Grammatiken). In **Abb.5** wird das Prinzip der statischen Integrationsspezifikation mit einem Integrationspfad durch Schemata und Metamodell anhand einer abstrakten Skizze veranschaulicht. Nach diesem Ansatz muß im allgemeinen nicht jedes Inkrement überhaupt integrierbar sein, wie es auch nicht erforderlich ist, jedem integrierbaren Inkrement genau ein Inkrement in der jeweils anderen "Säule" des "Tores" zuzuordnen. Solche Indeterminismen löst später der Benutzer in Interaktion mit dem entsprechenden Integrator. Auch sollte intuitiv klar sein, daß ein Integrationspfad nur an einer einzigen Stelle im Metamodell seine aufsteigende Laufrichtung in eine absteigende (bezüglich der Ordnung von Ober- und Unterbegriffen) ändern, aber nicht im Zickzack durch den "Torbogen" schlingern darf — eine entsprechende Formalisierung dieses Modellierungsansatzes finden wir in der Arbeit Jannings jedoch nicht. In der Abb.5 ist der Wendepunkt des Integrationspfades im Metamodell, ein gemeinsamer Oberbegriff der beiden zu integrierenden Inkremente, schwarz markiert.

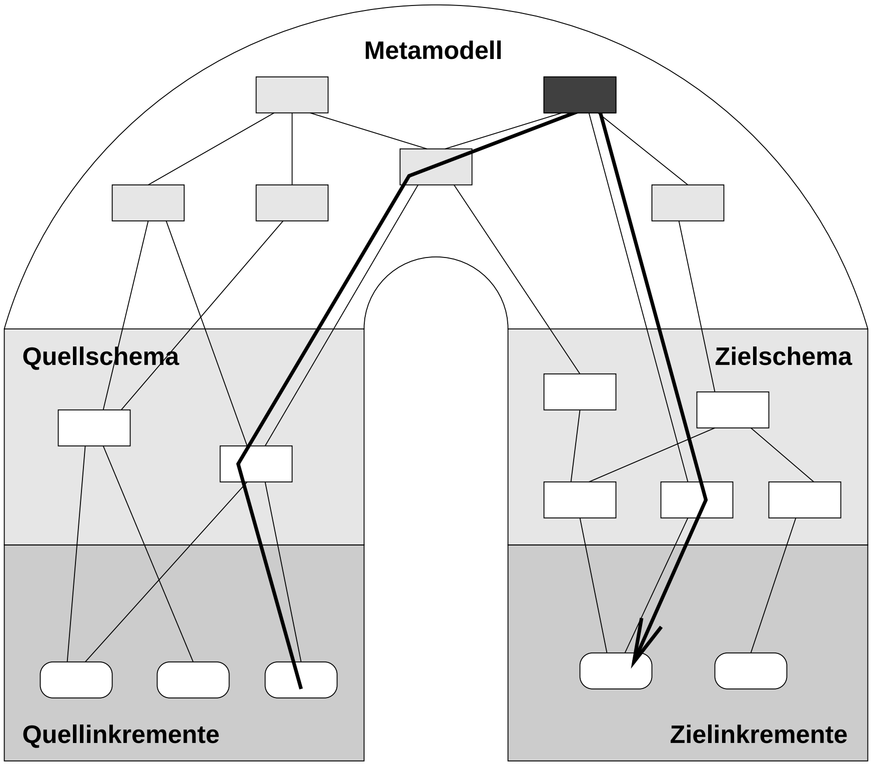

Abb.5

Ein anderes Werkzeug für den gleichen Zweck, nämlich der Integration von Anforderungs- und Architekturdokumenten, wird in der Dissertation von *M. Lefering* [23] wie zuvor bei Westfechtel durch die Paarung von Grammatiken spezifiziert, nun allerdings nicht mehr mit Textgrammatiken, sondern mit kontextsensitiven Graphgrammatiken, so wie dies bereits zu Anfang der Siebzigerjahre vorgeschlagen wurde [24]. Zur Definition seiner graphgrammatischen Paarungen verwendet Lefering die durch Jannings Metamodell beschriebenen Korrespondenzen, allerdings nur implizit, d.h.: wir finden dort noch keinen den Zusammenhang zwischen den beiden Spezifikationsmethoden beschreibenden Formalismus. Da wir auf Graphgrammatiken in unserem Beitrag noch ausführlich zu sprechen kommen werden, ist eine längere Kommentierung der Arbeit Leferings an dieser Stelle nicht erforderlich; wichtig ist hier lediglich zu erkennen, wie sich die Fäden der einzelnen Ansätze zur Spezifikation von Integrationswerkzeugen allmählich zu verflechten beginnen.

Außerdem lesen wir neuerdings von interessanten, an Leferings Arbeit anschließenden Ansätzen, die Paarung von Graphgrammatiken mit Hilfe einer tabellarisch formulierten Paarungsdefinition C automatisch in funktionsfähige Integratorprototypen zu übersetzen [25] und erachten dies als eine wichtige Parallelentwicklung zu unseren eher theoretischen Versuchen einer Formalisierung des Zusammenhanges zwischen statischer und dynamischer Integrationsspezifikation im Sinne der Methoden von Janning und Lefering.

Sowohl sämtliche in diesem Absatz unseres Berichtes aufgezählten Spezifikationsansätze, als auch die Weiterentwicklung dieser Methoden, stützen sich auf die Forschungsergebnisse von *A. Schürr*, der im Kontext unseres Arbeitsgebietes u.a. dreierlei gezeigt hat, nämlich: daß (i) Graphersetzungssysteme zu brauchbaren Softwareentwicklungsumgebungen erweiterbar sind, aus denen "rapid prototypes" generiert werden können, daß (ii) eine solche graphische Softwareentwicklungsumgebung die graphgrammatische Spezifikation von Integrationswerkzeugen wirkungsvoll unterstützt, und daß (iii) spezielle, geschichtete Graphgrammatiken —auf die wir im folgenden noch kurz zu sprechen kommen werden— nicht der Unentscheidbarkeit des allgemeinen Graphanalyseproblems (graph parsing) anheimfallen [26][27][28][29][30]. Ob eine Integrationsspezifikation durch geschichtete Graphgrammatiken darstellbar ist oder nicht, hat auf den nach dieser Spezifikation entstehenden Integrator folgende Auswirkung. Im ersten Fall ist eine lose gekoppelte Arbeitsweise möglich, was bedeutet, daß ein Integrationsschritt nach beliebig vielen Bearbeitungsschritten auf dem Modellgraphen durchgeführt werden kann, ohne daß die Historie dieser Arbeitschritte im Integrator gespeichert werden muß. Im zweiten Fall ist es zwingend erforderlich, einen Integrationsschritt durchzuführen, sobald ein Arbeitsschritt auf dem Modellgraphen durchgeführt wurde —wir nennen dies eine eng gekoppelte Arbeitsweise—, da wegen der Unentscheidbarkeit der allgemeinen Graphanalyse keine terminierende Berechnung einer erzeugenden Sequenz von Graphproduktionen mehr garantiert werden kann, anhand derer ja die gegenseitige Konsistenz zweier Modellgraphen entschieden werden muß, (vgl. Def.III.1). Aus dieser mißlichen Lage könnte man sich bei der Implementation des Integrators dann allenfalls durch die Speicherung der Historie aller Arbeitsschritte freikaufen. Obwohl es also wichtig ist, zu wissen, ob die Spezifikation eines Integrationswerkzeuges aus geschichteten Graphgrammatiken besteht oder nicht, sind unsere im folgenden Absatz vorgestellten Überlegungen von dieser Unterscheidung unabhängig, da wir zunächst nicht mit der Implementation eines bestimmten Integrators, sondern erst mit einer allgemeinen Spezifikationsmethode befaßt sind. (Bei der späteren maschinellen Unterstützung unserer Methode durch ein noch zu entwickelndes Softwarewerkzeug wird die Analyse der Schichteigenschaft von Graphgrammatiken allerdings kaum zu vermeiden sein.)

IV.

Das im vorausgegangenen Absatz definierte paargrammatische Korrespondenzprinzip (Def.III.1) macht in seiner Allgemeinheit keine Aussage über die Gestalt der Korrespondenzrelation C, sodaß wir, da ja auch unser Ansatz auf diesem Prinzip beruht, nun erläutern müssen, welche besonderen Eigenschaften einer solchen Relation C zukommen, die eine Vereinigung der oben besprochenen Konzepte charakterisiert. Dazu legen wir zunächst unsere Vorstellung von Graphenschemata, Graphersetzungsregeln, Graphgrammatiken usw. dar, wobei wir versuchen, uns von dem in [31] vereinbarten Begriffssystem der dt. "Graphtransformationsgemeinde" nicht allzuweit zu entfernen.

Die Abb.5 (Absatz III) deutet bereits an, daß ein Begriff in einem Schema nicht nur genau einem, sondern auch mehr als einem Oberbegriff untergeordnet werden darf, was im Bereich der objektorientierten Programmiersprachen als Mehrfachvererbung bekannt ist. Vorteil der Mehrfachvererbung gegenüber der Einfachvererbung, bei der jeder Begriff höchstens einem Oberbegriff untergeordnet werden darf, ist eine größere semantische Ausdrucksfähigkeit des Systems, Nachteil der Mehrfachvererbung im allgemeinsten Fall ist gegenüber der sicheren, da vererbungskonfliktfreien Einfachvererbung eine zu große syntaktische Beliebigkeit, was auch rasch zu semantischer Verwirrung führen kann. Deshalb gestatten wir Mehrfachvererbung ausschließlich innerhalb der syntaktisch strengen Form des (mathematischen) Verbandes [32], sodaß je zwei Begriffen innerhalb eines solchen Schemas genau ein kleinster gemeinsamer Oberbegriff sowie genau ein größter gemeinsamer Unterbegriff in diesem Schema zukommt. So erzielen wir gute semantische Mächtigkeit bei einfacher syntaktischer Struktur, und halten uns hierbei für implizit durch die Autoren von [33] bestätigt, welche unter ihren Schematransformationen auch eine zur Erzeugung verbandähnlicher Strukturen bereithalten.

Definition IV.1 (Graphenschema)

1) *Sei* $\Sigma = \{\perp, \top\} \uplus S'$ *mit* $S' \neq \emptyset$ *und* $\perp, \top \notin S'$ *eine Menge von Symbolen. Dann heißt ein Quadrupel* $S = (\Sigma, \preceq_S, q, z)$ *Graphenschema über* Σ, *falls folgende Bedingungen erfüllt sind:*

Die Ordnungsrelation $\preceq_S \subseteq \Sigma \times \Sigma$ *definiert einen mathematischen Verband über den Symbolen aus* Σ *so, daß bezüglich der kanonischen transitiven Erweiterung* $\preceq^*{}_S$ *der Verbandsordnung* $\preceq_S$ *gilt:* $\forall s_i \in S' : (\perp \preceq^*{}_S s_i) \wedge (s_i \preceq^*{}_S \top)$.

Ein Symbol $s_i \in S'$ *heißt finale Klasse genau dann wenn gilt:* $\nexists (y \neq \perp) : y \preceq s_i$.

Die Menge aller finalen Klassen in S *bezeichnen wir mit* $fin(S)$. *Das Symbol* $\perp$ *nennen wir die leere Klasse, das Symbol* $\top$ *die Allklasse des Graphenschemas. Alle anderen Symbole heißen abstrakte Klassen oder Oberklassen im Graphenschema.*

Ferner gelte $fin(S) := \mathcal{V} \uplus \mathcal{A}$ *mit* $\mathcal{V} \neq \emptyset$, *wobei* $\mathcal{V}$ *eine Menge von Knotenmarken ("vertex") und* $\mathcal{A}$ *eine Menge von Kantenmarken ("arcus") bildet.*

Außerdem gelte $\forall\ v \in \mathcal{V},\ a \in \mathcal{A},\ e \in \Sigma : (v \preceq^*{}_S e \wedge a \preceq^*{}_S e) \Longrightarrow e = \top$.

Eine Abbildung $\mathcal{Q} : \mathcal{A} \to \mathcal{V} \cup \mathcal{O}$ *ordnet den finalen Kantensymbolen Quellknoten- und eine Abbildung* $\mathcal{Z} : \mathcal{A} \to \mathcal{V} \cup \mathcal{O}$ *den finalen Kantensymbolen Zielknotenklassen zu, wobei* $\mathcal{O} := \{s | s \neq \top, t \preceq^*{}_S s, t \in \mathcal{V}\}$ *Oberklassen der finalen Knotenklassen, ohne* $\top$.

2) *Ein Graphenschema heißt redundant, wenn es mindestens zwei verschiedene Klassensymbole* e_i, e_j *enthält für die gilt:*

$T_i \subseteq fin(S)$, $T_j \subseteq fin(S)$, *mit* $T_i = \{t \mid t \preceq^*{}_S e_i\}$, $T_j = \{t \mid t \preceq^*{}_S e_j\}$ *und* $T_i = T_j$,

3) *Ferner ist es gestattet, wenn auch nicht erforderlich, die Menge $fin(S)$ beliebig in Terminalsymbole und Nichtterminalsymbole oder in Schichten im Sinne der geschichteten Graphgrammatiken zu sortieren. Wird eine solche Unterscheidung nicht eigens definiert, so bilden alle Symbole $s_i \in fin(S)$ eine einzige Schicht* •

Die Definition ist länglich, aber nicht schwierig zu verstehen; hier folgen einige Anmerkungen. Zu **1)** Wir werden gleich sehen, wie die Elemente aus $fin(S)$ zur Markierung von Graphenknoten und Graphenkanten verwendet werden. Wir werden ebenfalls sehen, wie in dem speziellen Fall von Graphersetzungsregeln sogar Marken oberhalb von $fin(S)$ verwendet werden dürfen. Zu **2)** Redundanz in Modellen und Spezifikationen ist manchmal nützlich, im Übermaß jedoch nicht. Das im Schlußabschnitt VII dieses Beitrages erwähnte Spezifikationswerkzeug sollte aufgrund dieser Definition in der Lage sein, auftretende Redundanzen in Graphenschemata warnend anzuzeigen. Zu **3)** Wir werden nachher an einem Beispiel den Unterschied terminaler und nichtterminaler Symbole anschaulich machen. Dem kundigen Leser ist überdies gewiß aufgefallen, daß unsere Graphenschemata auf die üblichen numerischen und textuellen Klassenattribute verzichten; dies geschieht o.B.d.A deshalb, weil wir uns hier mit Strukturen befassen, während das Wesentliche solcher Attributen in ihren Werten, d.h. außerhalb unserer Strukturen ruht.

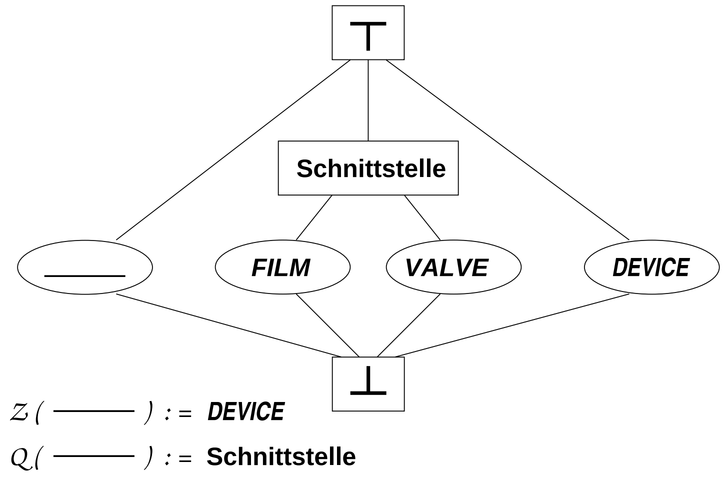

Abb.6

Die **Abb.6** gibt ein Beispiel eines ganz simplen, nicht redundanten Graphenschemas, dessen Begriffe der Sprache der chemischen Prozeßmodellierung entlehnt sind; (vgl. Abb.2). Die finalen Klassen eines Graphenschemas, hier rundlich eingezeichnet, repräsentieren "Dinge", genauer gesagt: die zu integrierenden Inkremente graphisch formulierter Partialmodelle, während die Oberklassen des Graphenschemas die selbst nicht integrierbaren Oberbegriffe repräsentieren, durch welche die Inkremente geordnet und verallgemeinernd beschrieben werden. Die Verbandsordnung selbst wird in Abb.6 durch Linien zwischen den Klassen dargestellt. Eine sinnvolle Interpretation dieser Abbildung ist durch die Proposition: "*Jedes* VALVE *und jeder* FILM *ist eine* Schnittstelle, *und nichts* ($\perp$) *ist zugleich* VALVE, FILM, DEVICE *und* —" beschreibbar. In diesem sehr kleinen Beispiel gilt $\mathcal{A}=\{—\}$; Quelle und Ziel von "—" sind die Klassen Schnittstelle und DEVICE.

Definition IV.2 (markierter, gerichteter, schematreuer Graph)
1) *Ein Hexupel $G = (V, A, M, q, z, m)$ heißt markierter, gerichteter Graph, wenn folgende Bedingungen erfüllt sind:*
V, mit $|V| < \infty$, ist eine Menge von Knoten.
A, mit $|A| < \infty$, ist eine Menge von Kanten.
Totale Funktionen vom Typ $q : A \to V$ und $z : A \to V$ ordnen den Kanten Quell- und Zielknoten zu.
M ist eine Menge von Marken.
Eine Funktion $m : V \uplus A \to M$ ordnet allen Knoten- und Kanten eine kennzeichnende Marke zu.
2) *Ein markierter, gerichteter Graph $G = (V, A, M, q, z, m)$ heißt schematreu zu einem gegebenen Graphenschema S, wenn gilt:*
$M = fin(S)$, $m(V) \subseteq \mathcal{V}$, $m(A) \subseteq \mathcal{A}$,
$\forall\ q(a) = v_q,\ z(a) = v_z$:
$\mathcal{Q}(m(a)) = V_q,\ \mathcal{Z}(m(a)) = V_z,\ m(v_q) \preceq^*{}_S V_q,\ m(v_z) \preceq^*{}_S V_z$ •

Definition IV.3 (Teilgraph, Differenzgraph, disjunkte Vereinigung)
1) *Seien $G = (V, A, M, q, z, m)$ und $G' = (V', A', M', q', z', m')$ zwei Graphen. G' heißt Teilgraph von G, genau dann wenn gilt:*
$V' \subseteq V$, $A' \subseteq A$, $M' \subseteq M$, $q' = q|_{A'}$, $z' = z|_{A'}$, $m' = m|_{A' \uplus V'}$,
und wir notieren $G' \subseteq G$.
2) *Sei G ein Graph und G' ein Teilgraph davon.*
Dann ist $G'' = G - G' := (V'', A'', M'', q'', z'', m'')$ der Differenzgraph von G und G', wobei
$V'' = V - V'$, $A'' = A - X$, *mit* $X := \{a \in A \mid q(a) \in V' \vee z(a) \in V'\}$,
$M'' \subseteq M$, $q'' = q|_{A''}$, $z'' = z|_{A''}$, $m'' = m|_{V'' \uplus A''}$
3) *Seien G, G' zwei Graphen mit $V \cap V' = \emptyset$ und $A \cap A' = \emptyset$.*
Dann ist der bipartite Graph $G'' = G \uplus G' := (V'', A'', M'', q'', z'', m'')$ die Vereinigung von G und G', wobei
$V'' = V \uplus V'$, $A'' = A \uplus A'$, $M'' = M \cup M'$,
$m''(x) = m(x)$ falls $x \in V \uplus A$, andernfalls m"(x)=m'(x),
$q''(a)$ und $z''(a)$ analog zu $m''(x)$ •

Nach diesen Vorbereitungen sind wir nun in der Lage, die Regeln einer Graphgrammatik, oder allgemeiner: eines Graphtransformationssystems zu beschreiben, wie wir sie in der Methode zur Spezifikation graphbasierter Integrationswerkzeuge benötigen.

Definition IV.4 (schematreue Graphersetzungsregel)
Ein Tripel $r = (L, R, K)$ heißt schematreue Graphersetzungsregel bezüglich eines gegebenen Graphenschemas S und eines gegebenen Graphen G, wenn folgende Bedingungen erfüllt sind:
Die linke Seite L und rechte Seite R sind jeweils Graphen ($\to$ Def.IV.2.1)
Der Regelkontext K ist ein (möglicherweise leerer) Graph mit $L \supseteq K \subseteq R$.
Jedes Anwendungsresultat $G' \in \mathcal{G} := r(G)$ ($\to$ Def.IV.6) *ist ein schematreuer Graph bezüglich S, falls r auf G anwendbar und G schematreu bezüglich S* •

Wir werden später sehen, daß die Symbole der linken Regelseite mit beliebigen Klassen eines Graphenschemas markiert (typisiert) werden dürfen, während Symbole der rechten Regelseite, welche Neuerzeugungen in einem "Wirtsgraphen" verursachen, stets mit finalen Klassen zu markieren (typisieren) sind. Diese repräsentieren ja unmittelbar "Dinge", d.h. Inkremente eines Partialmodells, während jene nur zur abstrakteren Beschreibung dieser Inkremente dienen können. Die Graphen K, L aus r müssen also nicht selbst schematreu sein ($\rightarrow$ Def.IV.2.2), damit r eine schematreue Graphersetzungsregel ist. Das Resultat von $r(G)$ ist im Allgemeinen eine Menge $\mathcal{G}$ von Graphen, da r nicht notwendigerweise an genau einer Stelle in G anwendbar ist. Die Wirkung von Graphersetzungsregeln auf gegebene Graphen ist der Gegenstand der nun folgenden Definitionen.

Definition IV.5 (schematreuer Graphhomomorphismus)
Sei S ein Graphenschema, G mit V und A, sowie G' mit V' und A' zwei Graphen ($\rightarrow$ Def.IV.2.1)*, m und m' Markierungen, welche die Knoten und Kanten von G bzw. G' auf beliebige Klassen aus S abbilden.*
Dann heißt eine totale Abbildung $h : (V \uplus A) \rightarrow (V' \uplus A')$
schematreuer Graphhomomorphismus bezüglich S,
wenn folgende Bedingungen erfüllt sind:

$h(a) = a' \Longrightarrow h(q(a)) = q'(a') \;\wedge\; h(z(a)) = z'(a')$.

$h(v) = v' \Longrightarrow m'(v') \preceq^*{}_S\, m(v)$.

$h(a) = a' \Longrightarrow m'(a') \preceq^*{}_S\, m(a)$,

und wir notieren $h : G \rightarrow G'$ •

Analog zu oben verlangen wir in der Definition schematreuer Graphhomomorphismen wiederum nicht, daß die beteiligten Graphen selbst die Schematreuebedingung (Def.IV.2.2) erfüllen.

Definition IV.6 (homomorphe Anwendung einer Graphersetzungsregel)
Sei S ein Graphenschema, G ein dazu schematreuer Graph und r eine Graphersetzungsregel.

1) *r ist anwendbar auf G, wenn folgende Bedingungen erfüllt sind.*

L ist die linke Seite von r,

$G' \subseteq G$ ist ein Teilgraph von G und

$\exists h : L \rightarrow G'$ schematreu.

2) *O.B.d.A setzen wir (ggf. durch Umbenennungen) voraus, daß G und $(R - K)$ namenskonfliktfrei. Da h auf $(R - K)$ nicht definiert, führen wir folgende Erweiterung ein: $\hat{h} := h|_{V(K) \uplus A(K)}$ und $\hat{h} := id|_{(V(R)-V(K)) \uplus (A(R)-A(K))}$. Die Anwendung einer Graphersetzungsregel $r = (L, R, K)$ auf einen "Wirtsgraphen" $G = (V, A, M, q, z, m)$ mit $h(L) = H_0 \subseteq G$ und $R = (V', A', M', q', z', m')$ verläuft dann schrittweise wie folgt.*

$\longrightarrow H_1 := G - (h(L) - h(K))$; *(Löschung)*

$\longrightarrow H_2 := H_1 \uplus (R - K)$; *(Erneuerung)*

$\longrightarrow H_3 := (V'', A'', M'', q'', z'', m'')$ *mit*

$A'' := A(H_2) \cup \{\hat{h}(a) | a \in A'\}$,

$q''(\hat{h}(a)) := \hat{h}(v_q)$ *wo* $q'(a) = v_q$,

$z''(\hat{h}(a)) := \hat{h}(v_z)$ *wo* $z'(a) = v_z$,

$V'' := V(H_2)$, $M'' := M(H_2) \cup M'$,

$m'' := m(H_2)$ "$\uplus$" m'; (Kanten- und Markierungsvervollständigung), und wir notieren: $H_3 \in r(G)$ •

Warum sagen wir: "$G-(h(L)-h(K))$", aber nicht: "$G-h(L-K)$"? Da wir keinen Isomorphismus von L in einen Teilgraphen von G verlangen, könnte es vorkommen, daß $\hat{h}$ zwei Knoten aus K und $L-K$ auf den selben Knoten in G projiziert. Im Falle $G-h(L-K)$ würde dann ein Bild aus dem Kontext K gelöscht, obwohl der Kontext ja gerade durch seine Invarianz ausgezeichnet sein soll. Im Falle $G-(h(L)-h(K))$ bleiben hingegen alle Abbilder von K erhalten. Beide Formulierungen sind jedoch äquivalent, falls $\hat{h}$ einen Isomorphismus darstellt. Da die Definition schematreuer Graphersetzungsregeln zwar "wohl", jedoch nicht konstruktiv formuliert ist, stellt sich nun aber die Frage, woran man schematreue Graphersetzungsregeln erkennen kann.

Lemma IV.7 (Gestalt schematreuer Graphersetzungsregeln)
Eine Graphersetzungsregel $r = (L, K, R)$ ist schematreu bezüglich eines Graphenschema S, wenn folgende Eigenschaften gelten:
Für alle Knoten $v \in (R-K) : m(v) \in \mathcal{V}$,
für alle Kanten $a \in R, a \notin K : m(a) \in \mathcal{A}$, wobei zugleich
$\forall q(a) = v_q, z(a) = v_z \in R$, $m(q(a)) = V_q, m(q(z)) = V_z$:
$V_q \preceq^*{}_S (\mathcal{Q})(m(a))$ und $V_z \preceq^*{}_S (\mathcal{Z})(m(a))$ •
Beweis: Durch $L-K$ verursachte Löschungen in einem "Wirtsgraphen" G sind irrelevant, da die Definition eines schematreuen Graphen keine Bedingungen an die Anzahl der in ihm enthaltenen Knoten oder Kanten stellt. Falls $(R-K) \neq \emptyset$, so erfüllen alle neuen Symbole in $(R-K)$ die Markierungsbedingungen der oben definierten Schematreue. Wegen "$\forall q(a) = v_q, z(a) = v_z \in R$" gilt dies insbesondere auch für diejenigen Kanten, welche die neuerzeugten Knoten an K binden, oder welche neu zwischen alte Knoten aus K gelegt werden. Da der oben definierte Homomorphismus $\hat{h}$ die strukturellen Eigenschaften von $(R-K)$ identisch in den resultierenden Graphen $H \in r(G)$ überträgt, ist H ein schematreuer Graph, falls r auf G anwendbar und G ebenfalls schematreu. Mit $m(v) = V'$ und $fin(S) \ni m'(\hat{h}(v)) \preceq^*{}_S V'$ für alle $v \in K$ folgt die Behauptung.$\surd$

Nota bene: falls L einer Graphersetzungsregel r den Erfordernissen der Schematreue widerspricht, so finden wir keinen Homomorphismus von L in einen gegebenen schematreuen Graphen G; dann ist r trivialerweise "schematreu", weil niemals anwendbar, sodaß durch r also auch kein Graph $H \in r(G)$ erzeugt werden kann, welcher nicht schematreu wäre.

Definition IV.8 (schematreue Graphgrammatik)
Sei S ein Graphenschema. Eine endliche Menge $GG := \{G_0, r_1, \ldots r_n\}$ heißt schematreue Graphgrammatik bezüglich S, wenn gilt: das Startaxiom G_0 ist ein schematreuer Graph bezüglich S, und alle $r_i \in GG$ sind schematreue Regeln bezüglich S •

Alle Graphersetzungsregeln einer schematreuen Graphgrammatik müssen also dem gleichen Graphenschema gehorchen, und es muß auch (mindestens) eine axiomatische Regel geben, welche aus einem leeren Graphen ϵ irgend etwas zu erzeugen vermag. In praxisrelevanten Spezifikationen sind unsere Graphgrammatiken und ihre Graphersetzzungsregeln recht komplex, doch ihr Prinzip ist kein anderes als das von den Textgrammatiken aus Chomskys Hierarchie wohlbekannte, sodaß wir nun alle begrifflichen Voraussetzungen zur Entwicklung der kombinierten, d.h. schematischen und grammatischen Korrespondenzmethode hinreichend dargestellt haben.

V.

Schon bei Lefering werden Graphersetzungsregeln nicht nur namentlich, d.h. als ganze, gepaart, sondern auch im Detail, d.h.: die Korrespondenzrelation C erfaßt auch einzelne Symbole aus den linken und rechten Seiten L_i und R_i der Regeln r_i der an der Integration beteiligten Graphgrammatiken, wodurch die Korrespondenz zweier Partialmodellgraphen, $G_1 =_C G_2$, die ja die "Sätze" unserer Graphgrammatiken bilden, ebenfalls nicht nur eine namentliche, sondern auch eine feiner detailierte ist, wie das folgende Beispiel verdeutlicht. (Der Ausdruck $G_1 =_C G_2$ wird später in diesem Abschnitt definiert.) Gegeben seien zwei korrespondierende Partialmodellgraphen $G_1 =_C G_2$, dazu zwei jeweils darauf anwendbare Graphersetzungsregeln r_1 und r_2, die in C korrespondieren, d.h.: $r_1 C r_2$. Für unser Beispiel wählen wir die beiden r_i vergrößernd, d.h.: $L_i \subset R_i$. Mit $G_1 =_C G_2$ und $r_1 C r_2$ gilt dann definitionsgemäß auch $G'_1 =_C G'_2$, wobei $G'_1 \in r_1(G_1)$ und $G'_2 \in r_2(G_2)$ zwei Ergebnisse der jeweiligen Graphersetzungen bezeichnen. Da aber nun $G_1 =_C G_2$ und $G'_1 =_C G'_2$, so wäre es höchst verwunderlich, wenn es nicht auch zwischen ihren Differenzen $(G'_1 - G_1)$ und $(G'_2 - G_2)$ irgendeine Art von Beziehung zu finden gäbe. In der Tat ist eine solche Teilgraphenkorrespondenz von der fein detailierten Korrespondenzdefinition der erweiternden Regelanteile $(R_1 - L_1)$ und $(R_2 - L_2)$ bestimmt. Dieses primitive Beispiel führt uns also im folgenden zu der Frage, auf welche Weise zwei (oder mehrere) Graphersetzungsregeln miteinander korrespondieren dürfen bzw. auf welche Weise sie nicht miteinander korrespondieren dürfen, und es sollte auch nicht mehr erstaunen, daß Graphenschemata zur Beantwortung dieser Frage eine wichtige Rolle spielen werden. Wir werden nun darlegen, wie die "dynamische" Integrationsspezifikation, also die Koppelung von graphgrammatischen Ersetzungsregeln, durch eine "statische" Integrationsspezifikation, nämlich die Koppelung der entsprechenden Graphenschemata, geleitet werden kann, wodurch später die Unterstützung der Spezifikationsarbeit durch ein Metawerkzeug, den Schemakorrespondenzeditor, ermöglicht wird. Die Idee der Schemakorrespondenzen haben wir in rudimentärer Form bereits in [34][35] vorgestellt, seither aber verändert und weiterentwickelt, sodaß auf ihre erneute Darstellung an dieser Stelle nicht verzichtet werden kann. Bevor die diesbezüglichen formalen Definitionen gegeben werden, wird diese Weiterentwicklung erst in einer informellen Darstellung rasch skizziert.

Die damals postulierte Unidirektionalität der Schemakorrespondenzen paßt nicht gut zu der Isotropie in dem hier verfolgten Ansatz von Pratt, der keine bestimmte Richtung einer Korrespondenzrelation bevorzugt. Der mit dem darum geübten Verzicht auf eine bevorzugte Richtung erhandelte Informationsverlust verlangt nun allerdings nach Kompensation durch einen neuen Typ von Korrespondenzaxiom, nämlich dem negativen, wie in **Abb.7** an einem kleinen Beispiel dargestellt. Nach der früheren, gerichteten Notation —siehe **a)**— war klar, daß der Kessel via Behälter und Flüssigkeit unter anderem mit dem Wasser korrespondiert, während die Tasse durch ihre eigene, speziellere Korrespondenz ausschließlich zum Wasser gehört. Wird nun auf die Vorzugsrichtung verzichtet —siehe **b)**—, so wäre die spezielle Verbindung zwischen Wasser und Tasse falsch, da sich das Wasser ja ebensogut in einem Kessel befinden dürfen sollte, während das Öl entgegen der ursprünglichen Intention via Flüssigkeit und Behälter plötzlich zur Tasse gelänge. Das nach Art der nichtmonotonen Logiken die Verbindung zwischen Öl und Tasse verbietende, speziellere, negative Korrespondenzaxiom —in **c)** mit der durchbrochenen Kante veranschaulicht— löst unser Dilemma. (Auf die Darstellung der unbeteiligten Schemaklassen $\perp, \top$ wird hier, wie auch zumeist im folgenden, verzichtet.)

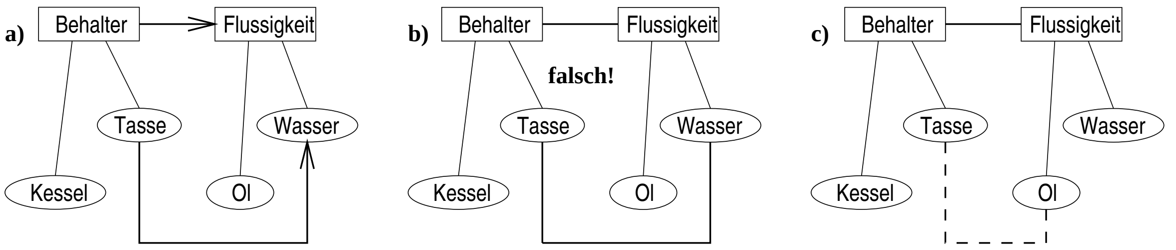

Abb.7

Die zweite Neuerung berücksichtigt die durch unser prozeßtechnisches Anwendungsgebiet gegebenen Situationen, in denen nicht mehr nur zwei, sondern auch drei oder mehr Partialmodelle integriert werden müssen, wobei die Darstellung einer solchen mehrgliedrigen Konstellation durch mehrere zweigliedrige Korrespondenzrelationen zu schlechten, weil sehr komplexen, fehleranfälligen Gesamtkonstellationen führen kann, wenn eine zweigliedrige Korrespondenz ohne Berücksichtigung der jeweils anderen zweigliedrigen Korrespondenzen definiert wird, wie in **Abb.8** zu sehen.

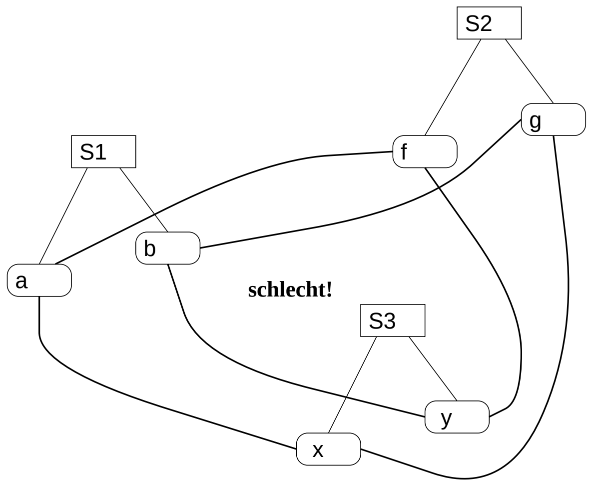

Abb.8

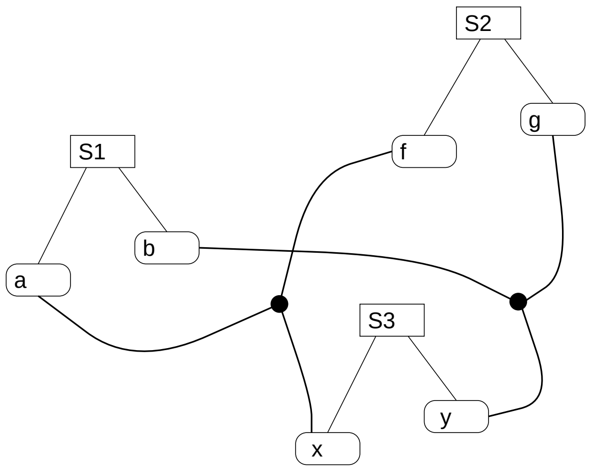

Abb.9

Wir wissen auch nicht recht die vielen topologischen Möglichkeiten einzuschätzen, eine Menge von Schemata durch jeweilige Paarungen zu einer Korrespondenzenkonstellation zusammenzuschalten — linear, ringförmig, radförmig,...? In unserem Ansatz wollen wir daher versuchen, mehrgliedrige Korrespondenzen durch Hyperkanten, also sternförmigen Gebilden mit mehr als zwei Enden, darzustellen, sodaß jedes Korrespondenzaxiom alle beteiligten Schemata berührt, wie in **Abb.9** skizziert. Unsinnige Spezifikationen werden zwar, wie man sich leicht vorstellen kann, auch durch unsere methodische Beschränkung auf Hyperkanten nicht ausgeschlossen, aber die Anzahl der möglicher Hyperkanten wächst nur recht beschränkt in der Anzahl der beteiligten Graphenschemata, während uns die Modellierung mehrgliedriger Korrespondenzen durch Paarungen schon bald mit einer unabsehbaren Anzahl von Möglichkeiten verwirren würde — man bedenke stets, daß allzu komplizierte Spezifikationen sehr wahrscheinlich Fehler enthalten müßten, welche die danach gebauten Integrationswerkzeuge unweigerlich unbrauchbar machen würden.

Definition V.1 (Schemakorrespondenz der Größe n)
Seien $S_1, \ldots S_n$ Graphenschemata mit ihren jeweiligen Symbolmengen Σ_i und Verbandsordnungen $\preceq_i$, $(i = 1, \ldots n)$.
Dann heißt eine (endliche) Menge $SK := \pi \uplus \nu$
von "positiven Axiomen" $\pi :\subseteq \Sigma_1 \times \ldots \times \Sigma_n$
und "negativen Axiomen" $\nu :\subseteq \Sigma_1 \times \ldots \times \Sigma_n$
Schemakorrespondenz über $S_1, \ldots S_n$, falls gilt:
$\forall (e_1, \ldots e_n) \in SK : e_i \neq \top_i,\ i = 1 \ldots n,$
$\forall (e_1, \ldots e_n) \in SK : e_i \neq \perp_i,\ i = 1 \ldots n,$
$\nexists (\ldots, e_i, \ldots e_j, \ldots) \in SK,\ \nexists v \in \mathcal{V}_i,\ \nexists a \in \mathcal{A}_j : v \preceq^*{}_i\ e_i \wedge a \preceq^*{}_j\ e_j,$
$\nexists (\ldots, e_i, \ldots e_j, \ldots) \in SK,\ \nexists a \in \mathcal{A}_i,\ \nexists v \in \mathcal{V}_j : a \preceq^*{}_i\ e_i \wedge v \preceq^*{}_j\ e_j$ •

Es können also jeweils nur Knoten- mit Knoten- und Kanten- mit Kantenklassen korrespondieren. Die anschauliche Bedeutung eines Korrespondenzaxiom ist die Benennung einer strukturellen, existenziellen Abhängigkeit (→ vgl. Def.V.6.1.$*$) von Inkrementen über Partialmodellgrenzen hinweg; (vgl. Abb.4). Zunächst betrachten wir —abgesehen von seiner Negation— nur einen syntaktischen Axiomentypus; denkbar wären allerdings auch mehrere verschiedene Axiomentypen mit der intuitiven Bedeutung verschiedener existenzieller Abhängigkeiten (über Partialmodellgrenzen) auf die eine oder andere Weise. Zum Konsistenzvergleich von schematischen und grammatischen Korrespondenzdefinitionen bleibt es uns allerdings nicht erspart, die Semantik einer Schemakorrespondenz auch formal zu definieren, (womit unsere früher gegebene Semantikdefinition [34] ebenfalls überwunden ist).

Definition V.2 (Semantik der Schemakorrespondenz)
1) *Sei SK eine Schemakorrespondenz der Größe n über den Graphenschemata $S_1, \ldots S_n$.*
Sei $A^+ = (e_1, \ldots, e_n)$ ein positives Korrespondenzaxiom.
Dann lautet seine Semantik
$[[A^+]] := \{(t_1, \ldots, t_n) \mid t_i \in fin(S_i),\ t_i \preceq^*{}_{Si}\ e_i,\ i = 1 \ldots n\}.$
Ebenso gilt für ein negatives Korrespondenzaxiom $A^- = (e_1, \ldots, e_n)$, daß
$[[A^-]] := \{(t_1, \ldots, t_n) \mid t_i \in fin(S_i),\ t_i \preceq^*{}_{Si}\ e_i,\ i = 1 \ldots n\}.$
2) *Seien nun $A^+{}_1, \ldots, A^+{}_k, A^-{}_1, \ldots A^-{}_m$ alle Korrespondenzaxiome aus SK.*
Dann lautet dessen Semantik $[[SK]] := (\bigcup_{i=1\ldots k} [[A^+_i]]) - (\bigcup_{j=1\ldots m} [[A^-_j]])$.
3) *Seien SK_i, SK_j zwei syntaktisch verschiedene Schemakorrespondenzen über gegebenen Schemata $S_1, \ldots S_n$. Dann heißen SK_i und SK_j äquivalent, falls $[[SK_i]] = [[SK_j]]$, und wir notieren $SK_i \equiv SK_j$* •

Es ist intuitiv klar, daß auf diese Weise alle kombinatorisch möglichen Korrespondenzen jeweils zwischen Knoten- oder Kantensymbolen spezifizierbar sind. Das zugrundeliegende Prinzip kennen wir aus jeder Vorlesung über nichtmonotone Logiken: "Vögel können generell fliegen, außer den Straußen, den Kiwis und den Pinguinen". In **Abb.10** sind drei zueinander äquivalente Schemakorrespondenzen der Größe 2 zu sehen, darin, wie oben vereinbart, die positiven Axiome als starke und die negativen Axiome als unterbrochene Linien zwischen den beiden Graphenschemata dargestellt werden.

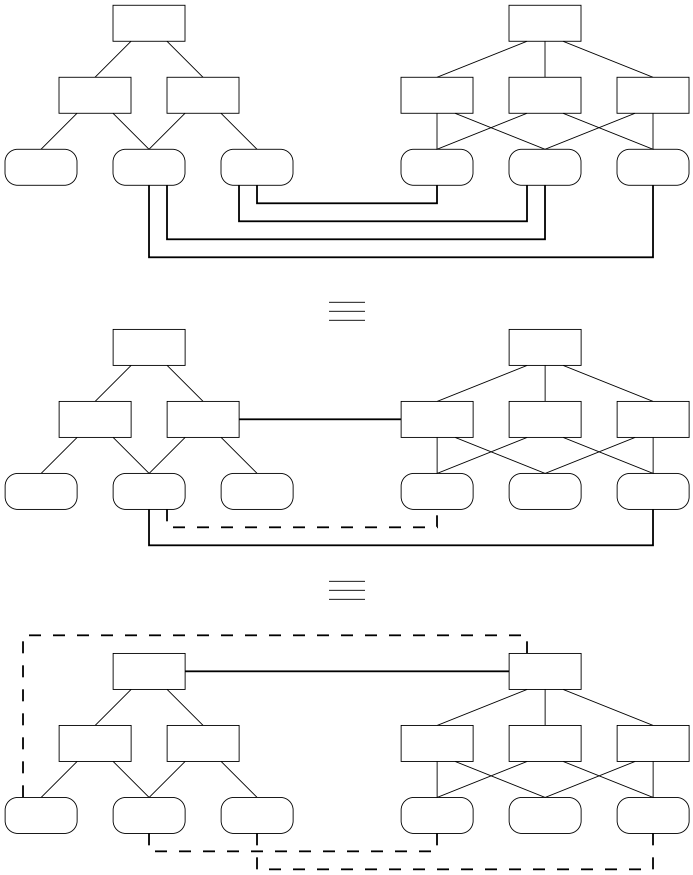

Abb.10

Um die Semantik einer Schemakorrespondenz zu erhalten, vereinigen wir also nach der Definition alle durch positive Axiome erlaubten Korrespondenzen und ziehen davon die Vereinigung aller durch negative Axiome ausgesprochenen Verbote ab, (soweit es überhaupt etwas abzuziehen gibt). In der Praxis benutzen wir zur Konstruktion von Schemakorrespondenzen selbstverständlich nicht allein die Verknüpfung finaler Klassen der beteiligten Schemata, sondern nutzen die semantische Äquivalenz verschiedener Korrespondenzen aus, um mit wenigen positiven und negativen Korrespondenzaxiomen auf Oberklassen elegante und übersichtliche Schemakorrespondenzen gleicher Bedeutung zu erhalten. Nachdem wir uns eine genaue Vorstellung der charakteristischen Eigenschaften aller Schemakorrespondenzen verschafft haben, werden wir nun verstehen, welche Bedeutung wir ihnen für die graphgrammatischen Korrespondenzen verleihen können.

Definition V.3 (schematreue Graphenkorrespondenz)
Sei SK eine Schemakorrespondenz über Graphenschemata $S_1, \ldots S_n$ und $G_1 = (V_1, A_1, \ldots), \ldots G_n = (V_n, A_n, \ldots)$ jeweils dazu schematreue Graphen.
Dann heißt $GK := VK \uplus AK$ schematreue Graphenkorrespondenz, wenn gilt:
$VK :\subseteq V_1 \times \ldots \times V_n$,

$AK :\subseteq A_1 \times \ldots \times A_n$ *und*
$\forall (v_1, \ldots v_n) \in VK : (m_1(v_1), \ldots m_n(v_n)) \in [[SK]]$, *sowie*
$\forall (a_1, \ldots a_n) \in AK : (m_1(a_1), \ldots m_n(a_n)) \in [[SK]]$,
und wir schreiben $G_1 =_{GK} G_2 =_{GK} \cdots =_{GK} G_n$ $\bullet$

Analog zu Abschnitt IV, da wir die Wirkung von Graphersetzungsregeln auf Graphen erläutert haben, benötigen wir nun ein Mittel zur Operation auf Graphenkorrespondenzen.

Definition V.4 (schematreue Graphersetzungskorrespondenz)
1) *Sei* SK *eine Schemakorrespondenz über Graphenschemata* $S_1, \ldots S_n$. *Seien* $G_1, \ldots G_n$ *dazu schematreue Graphen, sowie* GK *ihre schematreue Graphenkorrespondenz. Seien schließlich* $r_1 = (L_1, K_1, R_1), \ldots r_n = (L_n, K_n, R_n)$ *auf ihre jeweiligen Graphen anwendbare, schematreue Graphersetzungsregeln.*
Dann heißt ein $(n+3)$*-Tupel* $\mathcal{R} := (r_1, \ldots r_n, LK, KK, RK)$ *schematreue Graphersetzungskorrespondenz, wenn gilt:*
$LK :\subseteq (V(L_1 - K_1) \times \ldots \times V(L_n - K_n)) \uplus ((A(L_1) - A(K_1)) \times \ldots \times (A(L_n) - A(K_n)))$,
$KK :\subseteq (V(K_1) \times \ldots \times V(K_n)) \uplus (A(K_1) \times \ldots \times A(K_n))$,
$RK :\subseteq (V(R_1 - K_1) \times \ldots \times V(R_n - K_n)) \uplus ((A(R_1) - A(K_1)) \times \ldots \times (A(R_n) - A(K_n)))$,
sodaß jedes Anwendungsergebnis ($\rightarrow$ Def.V.6) $GK' \in \mathcal{R}(GK)$ *eine schematreue Graphenkorrespondenz zu* SK *ist, wo auch* GK *eine schematreue Graphenkorrespondenz.*
2) *Eine Kopplung* $(\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_n)$ *von Graphgrammatiken heißt schemakorrespondenztreue Integrationsspezifikation bezüglich* SK, *falls alle darin enthaltenen Graphersetzungskorrespondenzen* $\mathcal{R}_j$ $(j = 1 \ldots n)$ *schematreu sind bezüglich* SK, *und das gemeinsame Startaxiom* $\mathcal{R}_0$ *eine schematreue Graphenkorrespondenz bildet.* $\bullet$

Man beachte, daß die Graphersetzungskorrespondenzen durch die disjunkten Definitionen für Löschungen, Kontexte und Neuerzeugungen "zeitinvariant", genauer gesagt: ersetzungsinvariant sind, sodaß es z.B. nicht vorkommen kann, daß zwei ansonsten unveränderte Graphknoten nach Anwendung einer Graphersetzungskorrespondenz plötzlich nicht mehr miteinander korrespondieren, obwohl sie es vor der Anwendung getan haben.

Definition V.5 (schematreuer Korrespondenzhomomorphismus)
Sei SK *eine Schemakorrespondenz über Graphenschemata* $S_1, \ldots S_n$ *und* $GK = VK \uplus AK$, $GK' = VK' \uplus AK'$ *zwei Graphenkorrespondenzen über Graphen* $G_1, \ldots G_n$ *bzw.* $G'_1, \ldots G'_n$. *Seien ferner* $h_i : G_i \rightarrow G'_i$ *schematreue Graphhomomorphismen* ($\rightarrow$ Def.IV.5) *bezüglich* S_i, $i = 1 \ldots n$.
Dann heißt eine Abbildung $\mathcal{H} : (VK \uplus AK) \rightarrow (VK' \uplus AK')$
schematreuer Korrespondenzhomomorphismus bezüglich SK *und* h_i $(i = 1 \ldots n)$, *wenn die folgenden Bedingungen erfüllt sind:*
$\forall (v_1, \ldots v_n) \in VK \ \exists (v'_1, \ldots v'_n) \in VK' : (v'_1, \ldots v'_n) = \mathcal{H}((v_1, \ldots v_n))$.
$\forall (a_1, \ldots a_n) \in AK \ \exists (a'_1, \ldots a'_n) \in AK' : (a'_1, \ldots a'_n) = \mathcal{H}((a_1, \ldots a_n))$.
$\forall \mathcal{H}((x_1, \ldots x_n) \in GK) = (x'_1, \ldots x'_n) \in GK' : x'_i = h_i(x_i)$, $i = 1 \ldots n$ $\bullet$

Definition V.6 (Anwendung einer Graphersetzungskorrespondenz)
Sei SK eine Schemakorrespondenz, GK eine dazu schematreue Graphenkorrespondenz über Graphen $G_1, \ldots G_n$. Sei schließlich $\mathcal{R}= (r_1, \ldots r_n, LK, KK, RK)$ eine Graphersetzungskorrespondenz.
1) *$\mathcal{R}$ ist anwendbar auf GK, wenn folgende Bedingungen erfüllt sind:*
Die einzelnen r_i sind jeweils anwendbar ($\rightarrow$ Def.IV.6) *auf G_i bezüglich schematreuen Graphhomomorphismen $\hat{h}_i : L_i \rightarrow G_i$, $i = 1 \ldots n$.*
Es gibt einen zu SK schematreuen Korrespondenzhomomorphismus ($\rightarrow$ Def.V.5) *$\mathcal{H} : (LK \uplus KK) \rightarrow GK$.*
Wenn x ein Knoten oder eine Kante in G_i mit $x = \hat{h}_i(y)$ wo y ein Knoten oder eine Kante aus $(L - K)$ der Regel r_i, und $gk = (\ldots x, \ldots, x', \ldots) \in GK$ wo x' ein Knoten oder eine Kante aus G_j $(i \neq j)$, dann gilt auch $x' = \hat{h}_j(y')$ wo y' ein Knoten oder eine Kante aus $(L - K)$ der Regel r_j, und es gibt auch ein $lk \in LK$ mit $gk =\mathcal{H}(lk)$. . . . $(*)$
2) *Die Anwendung einer Graphersetzungskorrespondenz auf eine Graphenkorrespondenz verläuft dann schrittweise wie folgt:*
— *$G'_i = r_i(G_i)$ mit $\hat{h}_i$ für $i = 1 \ldots n$.*
— *$GK_1 := GK - \mathcal{H}(LK)$.*
— *$GK' := GK_1 \cup \{(x_1, \ldots x_n) \mid (y_1, \ldots y_n) \in RK, x_i = \hat{h}_i(y_i), i = 1 \ldots n\}$* •

In der Erläuterung zu Def.V.1 wird auf die Interpretation der Schemakorrespondenzen als existenzielle Abhängigkeiten zwischen Partialmodellinkrementen hingewiesen. Dieser Interpretation wird in der Bedingung $(*)$ von Def.V.6.1 Tribut gezollt, welche verlangt, daß ein in der Korrespondenzliste verzeichnetes Inkrement eines Graphen nur dann gelöscht werden darf, wenn zugleich auch all seine Korrespondenten in den anderen Graphen gelöscht werden. Außerdem muß eine solche korrekte Löschung durch $\mathcal{H}$ "bestätigt" werden und darf nicht "heimlich" von den beteiligten Graphersetzungsregeln alleine verursacht sein. Deshalb ist es im zweiten Schritt von Def.V.6.2 auch nicht erforderlich, mehr als $\mathcal{H}(LK)$ aus GK zu entfernen (vgl. im Gegensatz dazu Def.IV.6.2). Ähnliche Koexistenzbedingungen sind auch in der Datenbank- und Schemaintegrationsliteratur geläufig, vgl. z.B. [36].

Da auch die Graphersetzungskorrespondenzen "wohl", jedoch nicht konstruktiv definiert sind, beantworten wir die Frage nach der Erkennbarkeit schematreuer Graphersetzungskorrespondenzen analog zu Lm.IV.7 mit der folgenden Aussage.

Lemma V.7 (Gestalt schematreuer Graphersetzungskorrespondenzen)
Sei SK eine Schemakorrespondenz und $\mathcal{R}:= (r_1, \ldots r_n, LK, KK, RK)$ eine Graphersetzungskorrespondenz. $\mathcal{R}$ ist schematreu, wenn gilt:
$\forall (x_1, \ldots x_n) \in RK : (m_1(x_1), \ldots m_n(x_n)) \in [[SK]]$ •

Beweis: Da in der relevanten Definition der Graphenkorrespondenzen nichts über die Mächtigkeiten $|VK|$, $|AK|$ ausgesagt wird, spielen Löschungen durch $\mathcal{H}(LK)$ keine Rolle. Da für alle neuen Symbole $x_i \in R_i - K_i$ gilt: $m_i(x_i) \in fin(S_i)$ und $\forall (t_1, \ldots t_n) \in [[SK]] : t_i \in fin(S_i)$ $(i = 1 \ldots n)$, folgt sofort, daß durch die Einfügung von $\mathcal{H}(RK)$ die Schematreue einer Graphenkorrespondenz nicht zerstört werden kann, falls $\mathcal{R}$ überhaupt anwendbar. Insbesondere werden keine Korrespondenzen eingefügt, welche durch negative Axiome der entsprechenden Schemakorrespondenz SK verboten werden. Mit der oben definierten zeitlichen Invarianz von Korrespondenzen folgt das Behauptete.$\surd$

Nach diesen länglichen und eher theoretischen Ausführungen folgt im nächsten Absatz zur Illustration ein aus didaktischen Gründen kleines und einfaches graphgrammatisches Spezifikationsbeispiel mit einigen aus der Einführung bekannten Begriffen der chemischen Verfahrenstechnik. Darin werden wir u.a. sehen, wie die in der Definition des Graphenschemas (Def.IV.1.3) gestatteten Nichtterminalsymbole verwendet werden können.

VI.

In **Abb.11** ist **a)** ein Graphenschema für Phase-Systems und Chemical Components zusammen mit **b)** vier Graphersetzungsregeln gegeben, welche korrekte Graphen eines kleinen Materialmodells erzeugen. Die Hilfsklasse Anker gehört nicht eigentlich zum Materialmodell, wird aber benötigt, um zusammenhängende Materialgraphen erzeugen zu können. Die Kanten zwischen den Knoten der Graphersetzungsregeln werden in diesem Beispiel nicht im Graphenschema modelliert, weil wir hier nicht mit Kantenkorrespondenzen, sondern nur mit Knotenkorrespondenzen arbeiten wollen. (Implizit gilt also $\mathcal{A}=\{—\}$.) Die Knoten in den rechtsseitigen und linksseitigen Graphen der Graphersetzungsregeln sind durch Nummern 1, 2 dargestellt. Ihnen ist jeweils ihre Markierung $m(x) \in fin(S)$, $x = 1, 2$, angeheftet. Kommt die gleiche Nummer sowohl in der rechten als auch in der linken Seite einer Regel vor, so gehört dieser Knoten zum Kontext K. Es erübrigt sich zu erwähnen, daß jede Regel p_i einen neuen Namensraum öffnet, d.h. daß es sich z.B. bei 1 aus p_0 und 1 aus p_2 selbstverständlich um verschiedene Knoten handelt, obwohl sie mit der gleichen Klasse aus S markiert sind. Wie man leicht sieht, erlauben die Graphersetzungsregeln das Anhängen neuer Phase-Systems an einen (einzigen) Anker und analog das Anhängen neuer Chemical Components an ein beliebiges Phase-System.

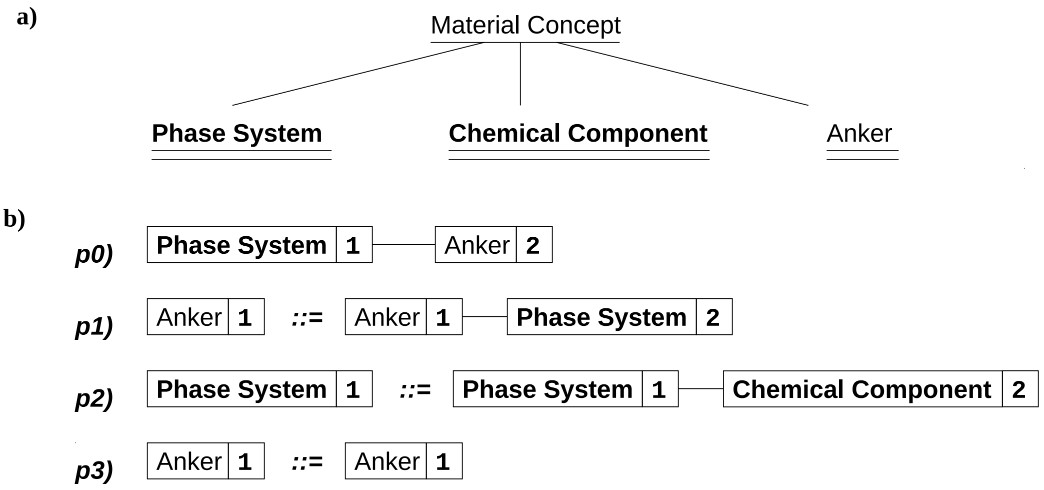

Abb.11

Analog finden wir in **Abb.12 a)** ein simples Graphenschema mit uns bereits aus der Einführung bekannten Begriffen des Strukturmodells, sowie **b)** Graphersetzungsregeln zum Aufbau der entsprechenden Modellgraphen. Eine Hilfsklasse Anker wird diesmal nicht benötigt, da diese Modellgraphen im Gegensatz zu oben in sich selbst verankert sind, eine Hilfklasse Nterm hingegen schon, welche nichtterminale Knoten markiert.

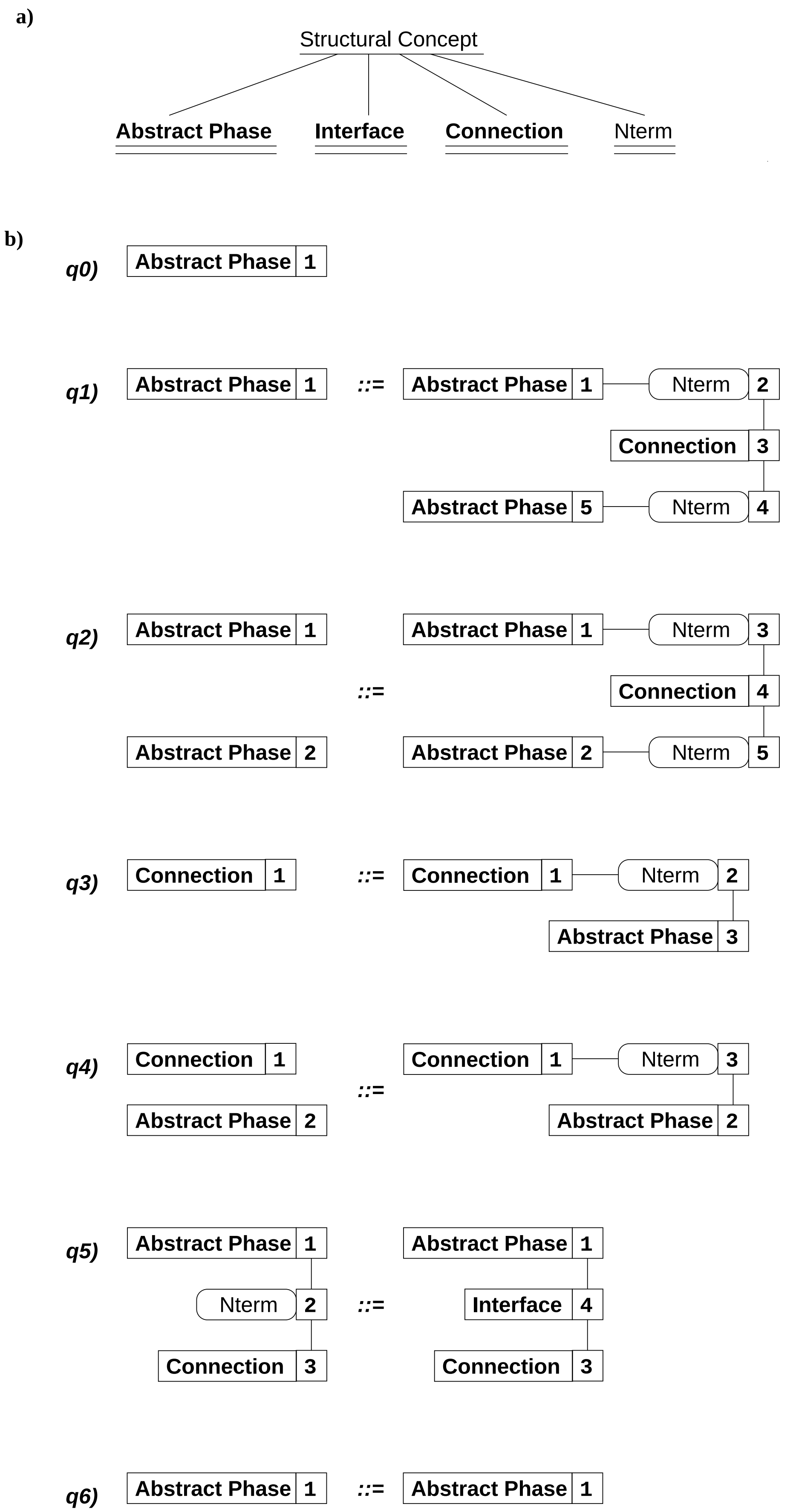
a)
Structural Concept
Abstract Phase
Interface
Connection
Nterm
b)
q0)
Abstract Phase 1
q1)
Abstract Phase 1
::=
Abstract Phase 1
Nterm 2
Connection 3
Abstract Phase 5
Nterm 4
q2)
Abstract Phase 1
Abstract Phase 2
::=
Abstract Phase 1
Nterm 3
Connection 4
Abstract Phase 2
Nterm 5
q3)
Connection 1
::=
Connection 1
Nterm 2
Abstract Phase 3
q4)
Connection 1
Abstract Phase 2
::=
Connection 1
Nterm 3
Abstract Phase 2
q5)
Abstract Phase 1
Nterm 2
Connection 3
::=
Abstract Phase 1
Interface 4
Connection 3
q6)
Abstract Phase 1
::=
Abstract Phase 1

Abb.12

Man beachte auch die unterschiedliche Bedeutung der Verbindungskanten jeweils im Graphenschema und in den Graphersetzungsregeln! Jene repräsentieren die Schemarelation $\preceq$, diese hingegen verbinden die Knoten in den Regeln zu Graphen (und werden in diesem Beispiel, wie auch oben, wiederum nicht mit Schemaklassen markiert). Es ist leicht zu erkennen, daß durch die Graphersetzungsregeln q0) bis q5) alle Graphen erzeugt werden, die aus mindestens einem Knoten namens Abstract Phase bestehen und in denen jede Connection via mindestens zwei Interfaces Verbindungen zwischen Abstract Phases zieht, (wobei wir Graphen, welche noch Nichtterminalsymbole vom Typ Nterm enthalten, als unvollständig zurückweisen). Die Grammatik sieht ein wenig kompliziert aus, und man könnte fragen, warum nicht eine einzige Regel für die Erzeugung beliebig vieler Abstract Phases definiert wird, welche dann durch weitere Regeln miteinander verbunden werden können. Da der Anwender aus dem Verfahrensingenieurwesen aber nur solche Strukturmodelle wünscht, welche zusammenhängende Graphen bilden, müßten wir eine solche primitivere Grammatik noch mit zusätzlichen Terminationsbedingungen in Gestalt weiterer Regeln ausstatten, damit alle vereinzelten Abstract Phases sicher verbunden werden. Deshalb ist unsere Beispielgrammatik so definiert, daß vereinzelte Knoten gar nicht auftreten können, d.h.: jede Satzform unserer Grammatik ist ein zusammenhängender Graph.

Die Modellierungsvorschrift der Verfahrensingenieure besagt nun, daß zwischen dem Symbol Phase System aus Abb.11a) und dem Symbol Abstract Phase aus Abb.12a) eine positive Schemakorrespondenz gesetzt werden muß, ebenso zwischen Phase System und Interface.

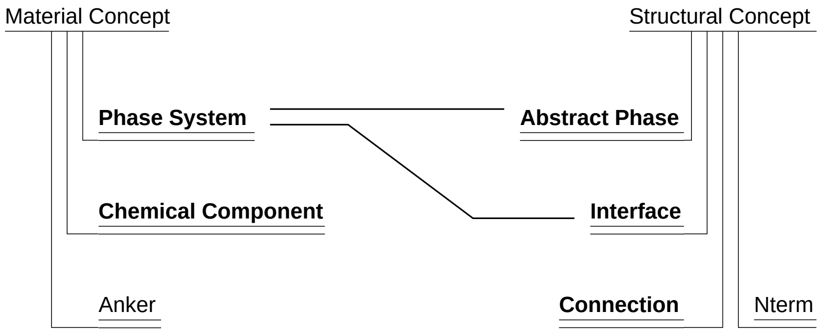

Abb.13

Nach dem bisher Gesagten lautet nun die Semantik dieser Schemakorrespondenz: $[[SK]] = \{$ (Phase System, Abstract Phase) (Phase System, Interface) $\}$ und determiniert die relative Korrektheit der folgenden grammatischen Korrespondenzdefinition C, deren Elemente so strukturiert sind: (Regel, Regel, L-korr., K-korr., R-korr.).

$C := \{$ (p0, q0, (), (), (1,1)) (i)
(p1, q1, (), (), (2,5)) (ii)
(p1, q3, (), (), (2,3)) (iii)
(p1, q5, (), (), (2,4)) (iv)
(p2, q6, (), (1,1), ()) (v)
(p3, q2, (), (1,1), ()) (vi)
(p3, q4, (), (1,2), ()) $\}$ (vii)

Wir beobachten, daß die Relation C in der Richtung $q_i \rightarrow p_j$ funktionalen Charakter aufweist, da C aus genau 7 Tupeln besteht, und jedes q_i, $i = 0 \ldots 6$, genau einmal darin vorkommt. Wir können also aus einem “q-Modell” simultanautomatisch —d.h.: durch sofortige Integration nach jeder Regelanwendung— integrierte “p-Modelle” erzeugen, da selbst die konstante (Dummy-)Regel q_6 via Korrespondenz $(1{,}1)$ die Anwendungsstelle ihrer Partnerin p_2 determiniert. In der anderen Richtung $\mathsf{p}_i \to \mathsf{q}_i$ kann C hingegen keinen funktionalen Charakter aufweisen, da nur vier Regeln auf sieben Tupel verteilt werden und somit mindestens eine Regel mehrfach vorkommen muß. Zum Abschluß dieses kleinen Lehrbeispiels zeigt die **Abb.14** die Erzeugung zweier korrespondierender Modellgraphen durch parallele Applikation in C korrespondierender Graphersetzungsregeln.

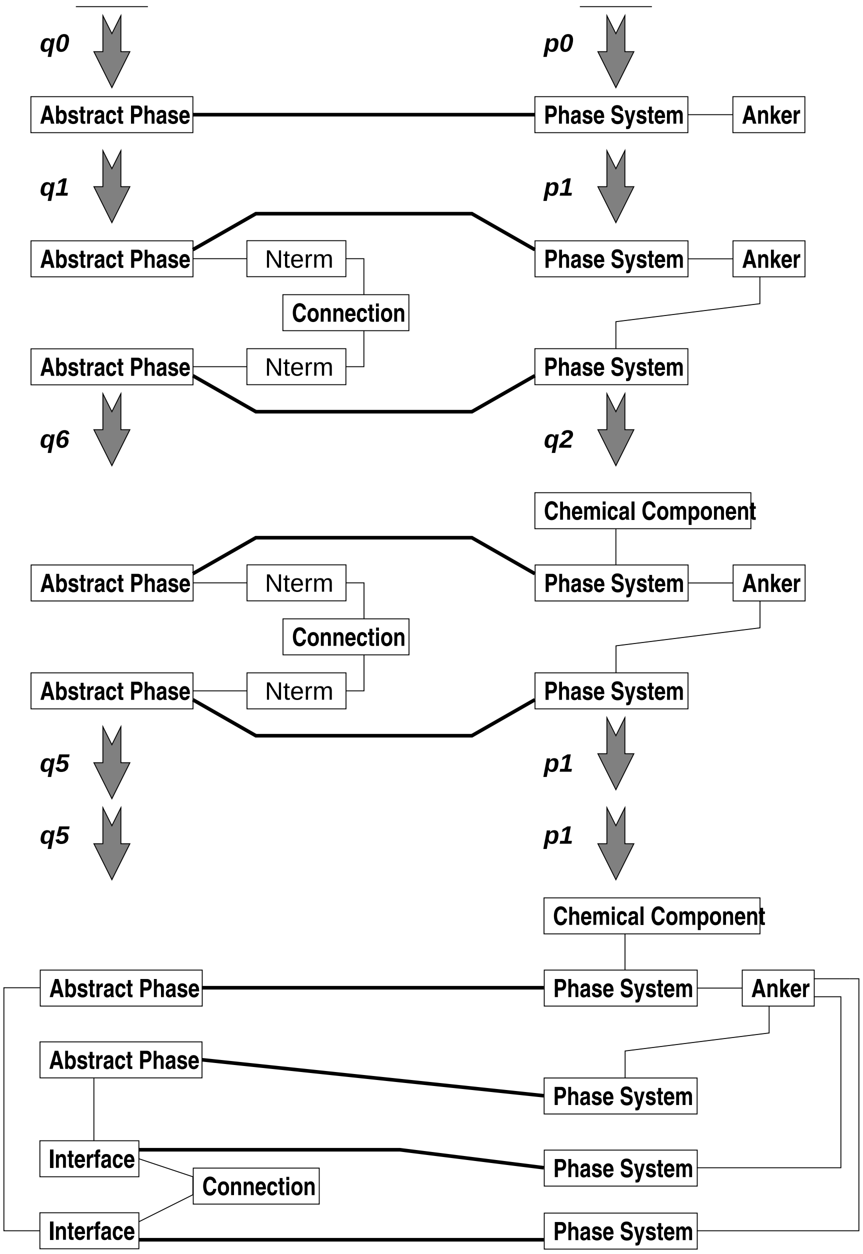

Abb.14

Die durch die grammatischen Korrespondenzen implizierten Knotenkorrespondenzen in den beiden so erzeugten Modellgraphen sind in der Abb.14 durch dicke Linien dargestellt und dürfen nicht mit den Schemakorrespondenzen aus Abb.13 verwechselt werden. Außerdem werden in der Abb.14 zur Übersichtlichkeit alle Graphenknoten nur durch ihre Markierungen repräsentiert.

In Abschnitt III wurde betont, daß nichtsimultane Integration durch Graphenparsing im allgemeinen nur dann möglich ist, wenn die beteiligten Graphgrammatiken zur Klasse der geschichteten Graphgrammatiken gehören. Dies ist in unserem Beispiel nicht der Fall, da sowohl die Graphersetzungsregel p_3 als auch q_6 als nichtvergrößernde Regeln die Schichtungseigenschaft ihrer jeweiligen Grammatik zerstören. Dennoch ist in unserem Beispiel das Doppelgraphenparsing entscheidbar! Ohne formalen Beweis begründen wir dies folgendermaßen. Die Graphenkorrespondenzen können als besondere Kanten aufgefaßt werden, welche sowohl die beteiligten Modellgraphen als auch die korrespondierenden Graphersetzungsregeln zu einem einzigen Supergraphen bzw. Supergraphersetzungsregeln verschmelzen. Dadurch gewinnen wir zusätzliche Information. Ferner beobachten wir, daß die beiden "faulen" Regeln p_3 und q_6 nicht miteinander verschmolzen werden, sondern jeweils mit einer anderen, "fleißigen" Graphersetzungsregel. Die so entstandene Supergrammatik, repräsentiert durch die Relation C, enthält in unserem Beispiel nur noch "fleißige" Superregeln, welche die Schichtungseigenschaft erfüllen. Unter günstigen Umständen ist also das Gesamtparsingproblem einer graphischen Partialmodellkonfiguration noch dann voll entscheidbar, wenn die Parsingprobleme der einzelnen Partialmodelle nur semi-entscheidbar sind. In unserem Beispiel gilt dazu eine ganz einfache Korrespondenzinvariante, nämlich:

#Phase System = #Abstract Phase + #Interface.

Im übrigen kann man oftmals durch geeignet definierte Sichten auf die zu integrierenden Partialmodelle genau diejenigen für die Integration irrelevanten Teile ausblenden, welche ansonsten die Einführung der "faulen" Graphersetzungsregeln in eine Integrationsspezifikation erzwingen würden. Allerdings folgt aus der Entscheidbarkeit der Konsistenz einer gegebenen Graphenkonstellation im allgemeinen nicht die deterministisch automatische Erzeugbarkeit eines korrespondierenden Graphen zu bereits gegebenen.

VII.

Alle in diesem Aufsatz gegebenen formalen Definitionen von Eigenschaften dienen der Unterstützung der Erstellung brauchbarer Integrationsspezifikationen durch ein Korrespondenzeditor genanntes softwaretechnisches Metawerkzeug, dessen Implementation in das softwaretechnische Rahmenwerk der Spezifikationsumgebung PROGRES [37] bereits begonnen hat. Von diesem Metawerkzeug und seinen auf den in den vorausgegangenen Absätzen dieses Beitrages vorgestellten Definitionen beruhenden Korrektheits- und Konsistenzanalysefunktionen werden wir berichten, sobald vorzeigbare Implementationsergebnisse zur Verfügung stehen. Anders als in [9], wo auf Seite 650 festgestellt wird: "We have not been interested in getting scientific results as such.", sind wir aber auch an jenem 'as such' interessiert, da die Brauchbarkeit von Konzepten und Ergebnissen in Zukunft u.U. anders beurteilt werden könnte als aus der unvermeidlich beschränkten Perspektive gegenwärtiger Betrachtung. Darüber hinaus hoffen wir, selbst auch ein kleines Epsilon in dieser Richtung beitragen zu können...

Die Definition des Graphenschemas (Def.IV.1) abstrahiert noch nicht weitestmöglich von unserer bisherigen, durch PROGRES bedingten praktischen Spezifikationsarbeit, in der Oberklassen von Kantenmarkierungen keine Rolle spielen. Tatsächlich weist Def.IV.1 nur den finalen Kantenmarkierungen Quelle und Ziel im Raum der Knoten-markierungen zu, nicht jedoch den Kantenoberklassen. Diese sind bis jetzt noch insofern nutzlos, als sie keinerlei Eigenschaften "nach unten" vererben und in den Graphersetzungsregeln noch nicht zur Kantenmarkierung verwendet werden. Wiese man auch den Kantenoberklassen Quelle und Ziel zu, so wären Kantenoberklassen auch in L und K, nicht jedoch in $(R-K)$, einer Graphersetzungsregel r als Markierungen zulässig und sinnvoll; eine Kantenunterklasse könnte dann so definiert werden, daß sie Quelle und Ziel ihrer Kantenoberklasse erbt und präzisierend einschränkt. Die entsprechende Modifikation von Def.IV.1 sei dem interessierten Leser überlassen.

Noch eine interessante Frage ist uns bereits des öfteren begegnet, nämlich, ob man nicht nur einfache Graphersetzungsregeln, sondern auch zusammengesetzte Graphtransaktionen als ganze nach dem Pratt'schen Korrespondenzprinzip koppeln könne. Graphtransaktionen werden im Wesentlichen durch reguläre Ausdrücke über Graphersetzungsregeln definiert, d.h.: Jede Graphersetzungsregel p ist auch eine Graphtransaktion gt, und sind gt_1 und gt_2 Graphtransaktionen, so auch $gt_1; gt_2$ (Komposition), $gt_1|gt_2$ (Alternative) sowie $\{gt_1\}^*$ (endliche, aber unbeschränkte Wiederholung). Im Rahmen unseres Forschungsprojektes verbleibt allerdings (leider) kaum noch Raum zu diesbezüglichen Untersuchungen. Auch haben wir bis jetzt noch nicht nach der Konfluenz unserer Graphersetzungssysteme sowie verwandten Eigenschaften gefragt.

Dank

A. Schürr korrigierte in mühseliger Arbeit etliche Fehler aus dem Manuskript, half mit noch mehr guten Ideen und Anregungen weiter und hat, aller Erfahrung zum Trotz, die Hoffnung auf die Lernfähigkeit seines mißratenen Schülers noch nicht gänzlich aufgegeben! Eventuell verbliebene Fehler und Mängel in dieser Arbeit sind selbstverständlich allein mir anzulasten.

Literatur

[1] H.J. Schneider, *Chomsky-Systeme für partielle Ordnungen*; Arbeitsberichte des Instituts für mathematische Maschinen und Datenverarbeitung, Band 3, Nummer 3, Friedrich-Alexander-Universität Erlangen und Nürnberg, August 1970.

[2] W. Marquardt, *Towards a Process Modeling Methodology*; in R. Berber (Ed.), Model-based process control, pp.3-41, Nato-ASI series Vol.293, Kluwer Academic Publishers, Dordrecht (NL) 1995.

[3] W. Marquardt et al., *The Chemical Engineering Data Model VEDA, Parts 1 - 6*; Technische Berichte, Lehrstuhl für Prozeßtechnik der RWTH Aachen, 1998.

[4] M. Nagl, B. Westfechtel (Hrsg.), *Integration von Entwicklungssystemen und Ingenieuranwendungen — Substantielle Verbesserung der Entwicklungsprozesse unter Nutzung existierender Systeme*; Springer-Verlag, Berlin 1998.

[5] B. Russel, *The Problems of Philosophy*, (dort insbesondere Kapitel 2); Oxford University Press, 1912.

[6] S. Gruner, M. Nagl, A. Schürr, *Fine-grained and structure-oriented Integration Tools are needed for (Product) Development Processes*; Aachener Informatikberichte 98-2, Fachgruppe Informatik der RWTH Aachen, ISSN 0935-3232, Januar 1998;

erscheint demnächst auch in M. Broy (Ed.), Proc. Nato Workshop on Requirements Targeting Software and Systems Engineering, (Bernried am Starnberger See, den 12. Oktober 1997).

[7] S. Gruner, M. Nagl, F. Sauer, A. Schürr, *Inkrementelle Integrationswerkzeuge für arbeitsteilige Entwicklungsprozesse*; in [4].

[8] G. Vollmer, *Auf der Suche nach der Ordnung — Beiträge zu einem naturalistischen Welt- und Menschenbild*; Wissensch. Verlagsges. S.Hirzel, Stuttgart 1995.

[9] M. Nagl (Ed.), *Building tightly integrated Software Development Environments: The IPSEN Approach*; Serie LNCS, Band 1170, Springer-Verlag, Berlin 1996.

[10] H. Nissen, *Separierung und Resolution multipler Perspektiven in der konzeptuellen Modellierung*; Dissertation an der Mathem.-Natw. Fakultät der RWTH Aachen, Serie DISDBIS, Band 38, Verlag Infix, Sankt Augustin 1997.

[11] D. Schefström, G. v.d. Broek (Eds.), *Tool Integration*; Wiley, Chichester 1993.

[12] ISO CD 10303, *Product Data Representation and Exchange*; NIST Gaithersburg.

[13] J. Conklin, *Hypertext — An Introduction and Survey*; IEEE Comp., pp.17-41, September 1987.

[14] G. Kaiser, S. Kaplan, J. Micallef, *Multi-user distributed language-based Environment*; IEEE Software 4/6, pp.58-67, 1987.

[15] P. Borras et.al., *Centaur — The System*; in P. Henderson, Proc.3rd ACM Software Engineering Symposium on Practical Software Development Environments; ACM Soft. Eng. Notes 13/5, pp.14-24, 1988.

[16] H. Ganzinger, R. Giegerich, *Attribute Coupled Grammars*; Proc. ACM Symp. on Compiler Construction, SIGPLAN Notices 17/6, pp.172-184, 1984.

[17] T. Reps, T. Teitelbaum, *The Synthesizer Generator Reference Manual*; Springer-Verlag, New York 1988.

[18] A. Aho, M Ganapathi, S. Tijang, *Code Generation using Tree Matching and Dynamic Programming*; TOPLAS 11/4, pp.491-516, 1989.

[19] J. Cordy, J. Carmichael, *The TXL Programming Language Syntax and informal Semantics*; Report 93/355, Computing and Information Science at Queen's University, Kingston (CDN) 1993.

[20] M. Jeusfeld, U. Johnen, *An executable Metamodel for Re-engineering of Database Schemas*; Int. Journal of Cooperative Information Systems 4(2/3), pp.237-258, 1995.

[21] B. Westfechtel, *Revisions- und Konsistenzkontrolle in einer integrierten Softwareentwicklungsumgebung*; Dissertation an der Mathem.-Natw. Fakultät der RWTH Aachen, Serie IFB, Band 280, Springer-Verlag, Berlin 1991.

[22] Th. Janning, *Requirements Engineering und Programmieren im Großen — Integration von Sprachen und Werkzeugen*; Dissertation an der Mathem.-Natw. Fakultät der RWTH Aachen, Deutscher Universitätsverlag, Wiesbaden 1993.

[23] M. Lefering, *Integrationswerkzeuge in einer Softwareentwicklungsumgebung*; Dissertation an der Mathem.-Natw. Fakultät der RWTH Aachen, Shaker, Aachen 1995.

[24] T. Pratt, *Pair Grammars, Graph Languages and String-to-Graph Translations*; Journal of Computer and System Sciences 5, pp.560-595, 1971.

[25] J. Jahnke, W. Schäfer, A. Zündorf, *A Design Environment for Migrating relational to object-oriented Data Base Systems*; Proc. Internat. Conference on Software Maintenance 1996, IEEE Computer Society Press, pp.163-170, 1996.

[26] A. Schürr, *Operationales Spezifizieren mit programmierten Graphersetzungssystemen — Formale Definitionen, Anwendungsbeispiele und Werkzeugunterstützung*; Dissertation an der Mathem.-Natw. Fakultät der RWTH Aachen, Deutscher Universitätsverlag, Wiesbaden 1991.

[27] A. Schürr, *Specification of Graph Translators with Triple Graph Grammars*; in Tinhofer (Ed.), Proc.WG'94 International Workshop on Graph-Theoretic Concepts in Computer Science, Serie LNCS, Band 903, pp.151-163, Springer-Verlag, Berlin 1994.

[28] A. Schürr, A. Winter, A. Zündorf, *Graph Grammar Engineering with PROGRES*; in Schäfer, Botella (Eds.), Proc.5th ESEC, Serie LNCS, Band 989, pp.219-234, Springer-Verlag, Berlin 1995.

[29] J. Rekers, A. Schürr, *Defining and Parsing Visual Languages with Layered Graph Grammars*; Journal of Visual Languages and Computing 7/3, 1996.

[30] A. Schürr, *Programmed Graph Replacement Systems*; in G.Rozenberg (Ed.), Handbook of Graph Grammars and Computing by Graph Transformation, vol.1 (foundations), pp.479-546, World Scientific, Singapur 1997.

[31] M. Andries, G. Engels, A. Habel, B. Hoffmann, H.-J. Kreowski, S. Kuske, D. Plump, A. Schürr, G. Taentzer, *Graph Transformation for Specification and Programming*; Bericht Nr.7/96 des Fachbereiches für Mathematik und Informatik, ISSN 0722-8996, Universität Bremen, September 1996.

[32] G. Birkhoff, *Lattice Theory*; American Math. Soc., 1948.

[33] P. Assenova, P. Johannesson, *Improving Quality in conceptual Modeling by the Use of Schema Transformations*; in B. Thalheim (Ed.), Conceptual Modeling ER96, 15th International Conference on Conceptual Modeling, Cottbus, Oktober 1996, Serie LNCS, Band 1157, Springer-Verlag, Berlin 1996.

[34] S. Gruner, *Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge*; Aachener Informatikberichte 97-5, Fachgruppe Informatik der RWTH Aachen, ISSN 0935-3232, Mai 1997. (Dankenswerterweise hat der Student D. Vogelheim auf zumindest einen formalen Fehler in einer der Definitionen dieser Publikation aufmerksam gemacht.)

[35] S. Gruner, *Werkzeugspezifikation mit Schemakorrespondenzen*; Softwaretechnik-Trends 17/3, ISSN 0720-8928, pp.39-42, 1997.

[36] V. Vidal, M. Winslett, *A rigorous Approach to Schema Restructuring*; in M. Papazoglou (Ed.), OOER95: Object-oriented and Entity-Relationship Modelling, Proc. of the 14th internat. Conf., Goldcoast (AUS), Dezember 1995, Serie LNCS, Band 1021, Springer-Verlag, Berlin 1995/96.

[37] Die Software ist im Cyberspace frei erhältlich bei
http://www-i3.informatik.rwth-aachen.de/research/progres/index.html

Appendix

Zur Belohnung aller, die bis hierhin tapfer durch die öde Korrespondenzwüste gewandert sind, gibt es zum Schluß in **Abb.15** eine erfrischende Karikatur, gezeichnet von dem bekannten Maler und Graphiker Heinrich Zille. Außerdem sei noch bemerkt, daß die römisch bezifferten Abschnitte unserer Arbeit absichtlich keine Überschriften tragen, damit der geneigte Leser (die Leserin per Default immer inklusive) dort mit einem Schreibgerät seiner Wahl ganz eigensinnig diejenigen Überschriften, welche ihm wahr und gut und schön erscheinen, eintragen kann, sofern er dies zu tun gedenkt.

Abb.15 *“Wenn de nochmal dein Frihstick vakoofst un sagst, du hast de Sperlinge jefuttert, denn bringt dir Vata uffs Jimnasium, da mußte noch ville mehr lernen; nu loof.”*