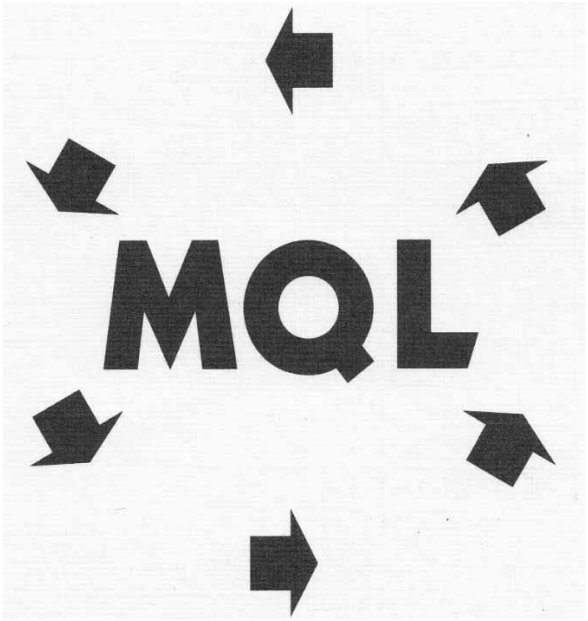
FORTSCHRITTSBERICHT
SIMULATION FBS 9



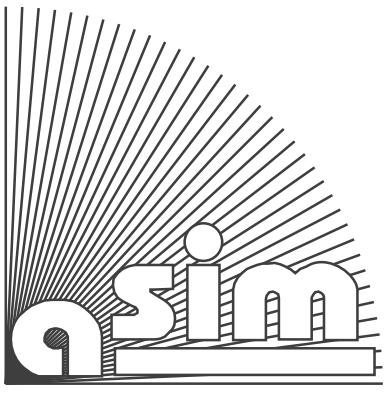
Thomas Preiß

Relationale Datenbanksysteme als Basis für Modellbildung und Simulation von kontinuierlichen Prozessen

ARGESIM

ISBN Ebook 978-3-903347-09-0
ISBN Print 978-3-901608-59-9

DOI: 10.11128 / fbs.09


argesim


ARGESIM

Fortschrittsberichte Simulation

FBS Band 9

Herausgegeben von **ASIM**
Arbeitsgemeinschaft **Sim**ulation, Fachausschuß 4.5 der GI

Thomas Preiß

Relationale Datenbanksysteme als Basis für Modellbildung und Simulation von kontinuierlichen Prozessen

ARGESIM / ASIM – Verlag, Wien, 2000
ISBN Print 978-3-901608-59-9

Ebook Reprint 2020
ISBN Ebook 978-3-903347-09-0
DOI: 10.11128/fbs.09

ASIM Fortschrittsberichte Simulation / ARGESIM Reports

Herausgegeben von **ASIM**, Arbeitsgemeinschaft Simulation, Fachausschuß 4.5 der **GI** und der **ARGESIM**

Betreuer der Reihe:

Prof. Dr. G. Kampe (ASIM)
Fachhochschule Esslingen
Flandernstraße 101, D-73732 Esslingen
Tel: +49-711-397-3741, Fax: --397-3763
Email: kampe@ti.fht-esslingen.de

Prof. Dr. D.P.F. Möller (ASIM)
Inst. F. Informatik, TU Clausthal-Zellerfeld
Erzstraße 1, D-38678 Clausthal-Zellerfeld
Tel: +49-5323-72-2404, Fax: --72-3572,
moeller@vax.in.tu-clausthal.de

Prof. Dr. F. Breitenecker (ARGESIM / ASIM)
Abt. Simulationstechnik, Technische Universität Wien
Wiedner Hauptstraße 8 - 10, A - 1040 Wien
Tel: +43-1-58801-5374, Fax: +43-1-5874211,
Email: Felix.Breitenecker@tuwien.ac.at

FBS Band 9

Titel: Relationale Datenbanksysteme als Basis für Modellbildung und Simulation von kontinuierlichen Prozessen

Autor: Thomas Preiß
Email: thomas.preiss@noel.gv.at

Begutachter des Bandes:

Prof. Dr. F. Breitenecker TU Wien, Prof. Dr. D.P.F. Möller Univ. Hamburg

ARGESIM / ASIM – Verlag, Wien, 2000
ISBN Print 978-3-901608-59-9

Ebook Reprint 2020
ISBN Ebook 978-3-903347-09-0

DOI: 10.11128/fbs.09

DISSERTATION

Relationale Datenbanksysteme als Basis für Modellbildung und Simulation von kontinuierlichen Prozessen

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften

eingereicht an der Technischen Universität Wien
Technisch Naturwissenschaftliche Fakultät
unter Anleitung von
a.o.Univ.Prof. Dipl.-Ing. Dr. Felix Breitenecker
von

Dipl.-Ing. Thomas Preiß
Speichberggasse 28
A-3002 Purkersdorf
Matr.Nr. 8625860
geboren am 30.04.1968 in Wien

Wien, im Februar 2000

Kurzfassung

Im großen Umfeld der Datenverarbeitung werden ausgeklügelte Systeme zur Speicherung und Verwaltung von Daten entwickelt und zu großer Leistungsfähigkeit optimiert. Diese Daten ergeben sich einerseits in der direkten Eingabe und Verwaltung, wie es bei „klassischen“ Datenverarbeitungsanwendungen der Fall ist, andererseits erst nach einer rechner- und zeitintensiven Verarbeitung, wie es bei diskreter und kontinuierlicher Simulation erfolgt. Diese relationalen DatenBank-Management-Systeme (*RDBMS*) können problemlos Verbindungen (Relationen) herstellen, beherrschen direkte Zugriffe und verstehen es, diese Zugriffe mittels Metadaten zu optimieren.

Dieses „Wissen“ des *RDBMS* über die in ihm enthaltenen Informationen ist es auch, das die Verwendung in der Simulation motiviert. Es werden die einzelnen Objekttypen direkt bei der Definition in den dafür vorgesehenen Tabellen abgelegt. Diese Vorgangsweise ermöglicht bei der weiteren Modellierung als auch Experimentdurchführung einen effizienten und raschen Aufbau des Modells bzw. der Modellstruktur.

In mehreren Kapiteln soll nun gezeigt werden, wie relationale Strukturen für kontinuierliche Simulation nutzbar zu machen sind. Die Darstellung erstreckt sich von der Darstellung der grundlegenden Theorien bis zu den Detailanalysen tatsächlicher Implementierungskonzepten.

Kapitel 1 stellt einen Überblick der aktuellen Begriffswelt der Simulation dar. Wir motivieren nach der Darstellung der Einsatzgebiete, der Methodologie und Gliederung von Modellbildung und Simulation die Erweiterung der Modellbildung durch die Einbindung relationaler Schemata.

Die dafür erforderlichen theoretischen Überlegungen werden in Kapitel 2 dargestellt. Da die Anwendung relationaler DatenBank Management Systeme *RDBMS* Ziel unserer Arbeit ist, werden deren Grundlagen angeführt. Wir bauen darauf die Theorie eines Tupelkalküls auf, das zur Darstellung von Modellbildung, Verknüpfung von Modellen und deren numerischer Auswertung, der Simulation an sich, geeignet ist. Dieses Konzept nennen wir „Datenbank basierende Modellverknüpfung“ (*DataBased Model Relation – DBMR*). Die Einbettung dieser Punkte in das neue Tupelkalkül führt auch zu einer Modifikation des bisherigen

Modellbegriffs.

Ebenso finden in diesem Kapitel die für die Anwendungsbeispiele erforderlichen Grundlagen, wie hierarchische Modellbildung und Modellbildung mittels Bondgraphen, Beachtung.

In Kapitel 3 verwenden wir *RDBMS*, die Erstellung von Simulationsergebnissen zu erweitern. Wir stellen nach der Präsentation von Integrations- und Interpolationsmethoden eine Einbettung in unser Relationenkonzept vor. Die dafür erforderlichen Algorithmen werden mittels der Tupeloperationen definiert. Dieses Auswertungsprinzip von Modellen nennen wir „Datenbank basierende Inpterpolation/Datenbank basierende Berechnung“ *DataBased Interpolation/DataBased Calculation – DBI/DBC*).

Die in den vorhergehenden Abschnitten dargestellten Konzepte finden ihren Niederschlag in Kapitel 4. Wir definieren die formale Sprache *Model Query Language*, die die Anwendung des Tupelkalküls ermöglicht. Die in der Herleitung des grundlegenden Relationenschemas nicht berücksichtigte Normalisiertheit wird nachgezogen und stellt daher eine wichtige Basis für die Einbettung in *RDBMS* dar.

Eine Detailanalyse für eine Implementierung stellt Kapitel 5 dar. Wir beschreiben die ausgewählte Menge an Elementen der Sprache *MQL* und das dafür erforderliche Datenmodell. Ebenso ist die Konzeptionierung der Benutzschnittstellen erforderlich.

Einge praktische Anwendungen sind in Kapitel 6 beschrieben. Diese sind derart ausgewählt, dass sämtliche Aspekte des neuen Konzepts beleuchtet werden können. Kapitel 7 fasst die wichtigsten Punkte des Implementierungskonzepts *DABEQS* dar.

Schließlich möchte ich Prof. Felix Breitenecker für die Möglichkeit danken, diese Überlegungen als Dissertation zusammenzufassen.

Abstract

In data processing sophisticated systems are developped, which store and manage large amounts of data. These systems are optimized to perform on a high level of availibility and reliability. Data, which must to be processed, is created by online input, which occurrs in classical database applications; another possibility of creating data is, that this data is a result of large numerical computations, which are especially done by discrete event or continuous system simulation.

DataBase Management Systems (DBMS) are able to create relations between special kinds of sets, and can access data directly. This access is optimized by the usage of meta-data. The fact, that the DBMS recognizes these meta-data, is the motivation to use data base management systems in continuous system modelling. A model of continuous simulation based on the theory of sets is presented and these sets are the base of the relational model of continuous simulation. The different object types are stored in the tables of the DBMS. Using this relational data model leads to more efficient modelling and easier experimentation. The "flexible input/output mechanism", which is comparable to indices on tables of the relational schema, opens a wide range of possibilities of joining models.

This thesis describes different aspects of using relational structures in continuous simulation. The range of description of these aspects starts with the fundamental theories and concepts towards the analysis of real implementations.

In chapter 1 we present an overview on simulation. Several topics of simulation and the structure of modelling and simulation are shown there. We also expand the terms on modelling with the inclusion of relational schemas.

The fundamentals of our theory on the *data based model relation – DBMR* are presented in chapter 2. In the beginning we describe the basics of relational database management systems (*RDBMS*). These fundamentals are necessary to construct our theory of tuple calculus, which describes the modelling, joining of models and calculating numerical results. Due to the definition of the tuple calculus we have to modify a bit the ordinary view of models. We also describe

shortly hierarchical modelling and bond graph modelling.

We use *RDBMS* to expand the calculation of simulation results in chapter 3 and present an overview on algorithms for interpolation and integration and finally we use the operations of the tuple calculus to define additional methods, the *data based interpolation/data based calculation – DBI/DBC* to get numerical results.

In chapter 4 the concept of *DBMR* to define a formal language *model query language – MQL* is used. Due to the property of the relational schema (it is of non first normal form), we have to create a new one, which is normalized. In this form *DBMR* is embedded. A detailed analysis of a implementation is given in chapter 5.

In chapter 6 some examples are descriped and explained. These are selected to present each aspect of the new concept *DBMR*. An appendix, which lists some useful information, is given in chapter 7.

Finally I want to express my thanks to Prof. Felix Breitenecker for the chance to present these concepts in my thesis.

Inhaltsverzeichnis

Kapitel 1

Überblick Simulation

Wie wir schon in der Motivation zu dieser Arbeit festgestellt haben, wird das Modellieren dynamischer Systeme durch die enorme Leistung, die die Computertechnologie zur Verfügung stellt, erleichtert. So hat sich das Werkzeug „Simulation“ in praktisch allen Bereichen etabliert, in denen das Verhalten und die inneren Zusammenhänge eines komplexen Systems studiert werden sollen. In den letzen Jahren ist eine Fülle von Computersprachen entwickelt worden, die die Nachbildung kontinuierlicher sowie diskreter Modelle ermöglichen.
Der Begriff Simulation leitet sich aus dem Lateinischen *simulare* ab und bedeutet soviel wie *nachbilden, nachahmen, etwas vortäuschen.* Für die Welt der Wissenschaft, die sich zusehends immer mehr der Simulation bedient, ist diese Definition als „Täuschung“ doch etwas zu ungenau. Daher gibt es einige Definitionen, die aus diesem Bereich hervorgegangen sind. Wir betrachten eine davon, die VDI Richtlinie 3633:

> *Simulation ist die Nachbildung eines dynamischen Prozesses in einem Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind.*

Eine Grafik (Abbildung 1.1) veranschaulicht dies, und zeigt wie die gewonnen Erkenntnisse direkt auf die Realität einwirken. Wir sehen, dass die Modellbildung kein einmaliger Prozess ist, sondern dass durch die ständige Rückkopplung zwischen dem Modell und Realitätsbeobachtungen eine laufende Überprüfung und allfällige Modifikation des sich im Einsatz befindlichen Modells erfolgt.

(*Validierung des Modells*)

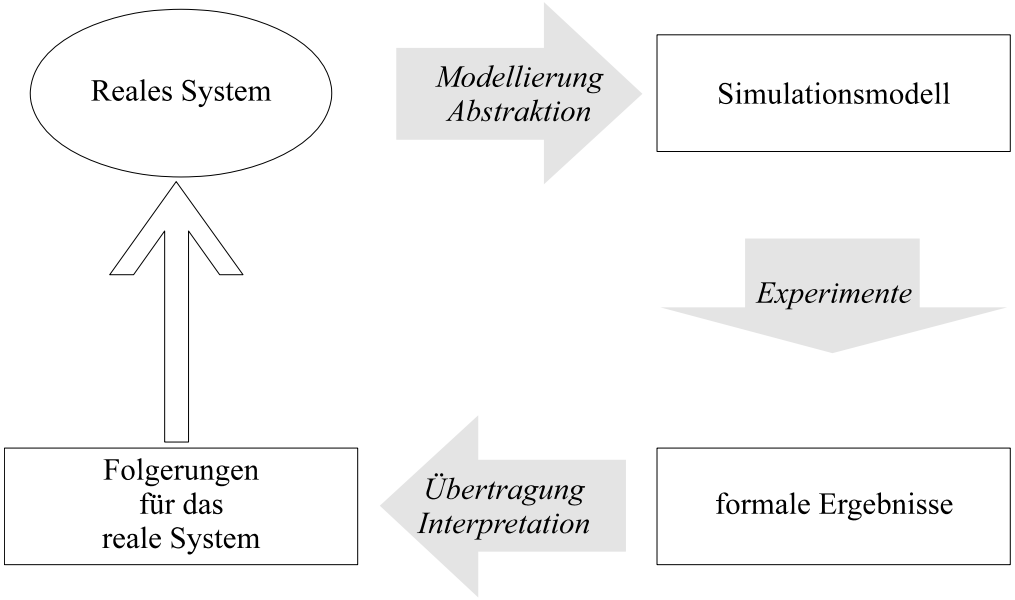


Abbildung 1.1: Der Zusammenhang zwischen Simulation und Realität

1.1 Einsatzgebiete der Simulation

Die Technik bedient sich der Simulation immer mehr als Standardwerkzeug, sodass sie in fast allen Disziplinen zu finden ist:

- Umweltsimulatoren vollziehen Entwicklungen unter der Einwirkung speziell vorgegebener Einflüsse,
- Modelle von Lebensvorgängen erlauben Verbesserungen in der ärztlichen Ausbildung und der Entwicklung von medizinischen Geräten,
- chemische Modelle erlauben die Berechnung der optimalen Reaktionsabläufe,
- graphische Modelle von ökonmischen, ökologischen und sozialen Systemen lassen mitunter deren Zusammenhänge deutlicher hervortreten,
- Fahr- und Flugsimulatoren bilden reale Situationen nach und lassen Steuerungsfunktionen des Menschen in die laufende Simulation einfliessen,

- Fertigungssimulatoren ahmen die Funktion der in einem Fertigungsablauf vorkommenden Prozesse nach,
- Modellbildung mit Hilfe von Datenbanken, wie sie in dieser Arbeit vorgestellt wird, erlaubt zusätzlich die genaue Zuordnung und Verwaltung der in der Simulation entstehenden Datenmengen, seien es Modell- oder Experimentsdaten

1.2 Methodologie der Modellbildung

Wie schon erwähnt, ist die Grundlage der Simulation das Finden eines mathematischen Modells, das das Verhalten des dynamischen Prozesses hinreichend gut beschreibt. Das Aufstellen eines Modells erfordert zweierlei Art Information über den Prozess:

- Kenntnis über und Einsicht in den Prozess
- Experimentelle Messdaten von Ein- und Ausgängen

Kenntnisse über den Prozess erlauben es nun, mit Hilfe der Deduktion ein mathematisches Modell aufzustellen. Messdaten allerdings beinhalten keine Aussage über die innere Struktur des Prozesses, man bedient sich daher induktiver Methoden zur Modellbildung. Eine Begrenzung auf eine der induktiven beziehungsweise deduktiven Methode ist aber kaum möglich, so muss mit einer Kombination beider gearbeitet werden.
Bei Verwendung rein induktiver Vorgangsweisen (z.B. Finden einer „passenden“ Formel, die Eingänge in bestimmte Ausgangswerte überleitet) stellt die Frage, welche die beste Formel, welches Modell aus einer Vielzahl von in Frage kommenden ausgewählt werden soll. Eine genaue Validierung (feststellen, ob das Modell gut und passend ist) ist nur durch Festsetzen zusätzlicher Systemgrenzen möglich, da sonst die Bandbreite einer Abweichung zu groß ist.
Ein weitaus „mathematischeres“ Vorgehen ist die deduktive Methode. Der zu simulierende dynamische Prozess wird in Einzelschritte zerlegt, deren mathematische Prinzipien bekannt sind. Aus diesen lässt sich dann leicht in analytischer Weise die mathematische Modellbeschreibung ableiten. Messdaten werden dann

nur noch zur Modellvalidierung herangezogen. Allerdings müssen auch hier klare Systemgrenzen gezogen und Modellvereinfachungen durchgeführt werden, da sonst das Modell in seinem Umfang zu groß wird und dann Modellbehandlung beziehungweise Simulation mit hohem Aufwand verbunden ist. (Vernachlässigung von Luftwiderstand, Rollreibung etc.).

1.3 Gliederung von dynamischen Modellen

Wenn man im Folgenden eine Unterteilung dynamischer Modelle vornimmt, so basiert diese auf den vorgenannten Überlegungen über die Art der Gewinnung der Modellgleichungen.
Betrachtet man nun ein Modell, so erkennt man, dass bei der Bildung des Modells Induktion und Deduktion verzahnt sind. Bei einem Modell, dessen innere Struktur nicht erkannt werden konnte und so nur das Ausgangsverhalten als Reaktion auf bestimmte Eingangsgrößen und Parameter modelliert werden konnte, spricht man von sogenannter „black box“ Modellbildung. Ist aber die Möglichkeit gegeben, das Modell ganz oder teilweise mittels deduktiven Methoden zu beschreiben, spricht man „white box“ bzw. „gray box“ Modellbildung. Eine Skizze (Abbildung 1.2) verdeutlicht dies und gibt Beispiele für diese Unterteilung. Wollen wir unsere Konzepte in dieses „Schema“ einordnen, erkennen wir, dass die Verbindung von Modellbildung und relationalen Datenbanken in jedem dieser Bereiche angesiedelt werden kann. Wir gehen ja von einem mathematischen Modell aus, über dessen Ursprung wir keinerlei Einschränkung machen werden.

1.4 Werkzeug der Simulation - Simulationssprachen

Um eine Simulation praktische durchführen zu können, ist beinahe immer die Hilfe eines Computers von Nöten, da die mathematischen Modelle eine händische Lösung im Allgemeinen nicht zulassen. Dies ist darauf zurückzuführen, dass die Differentialgleichungssysteme, die bei der Modellierung kontinuierlicher Systeme entstehen, direkt nur in den seltensten Fällen gelöst werden können, bzw. bei diskreten Modellen die Beobachtung und statistische Auswertung ein sehr

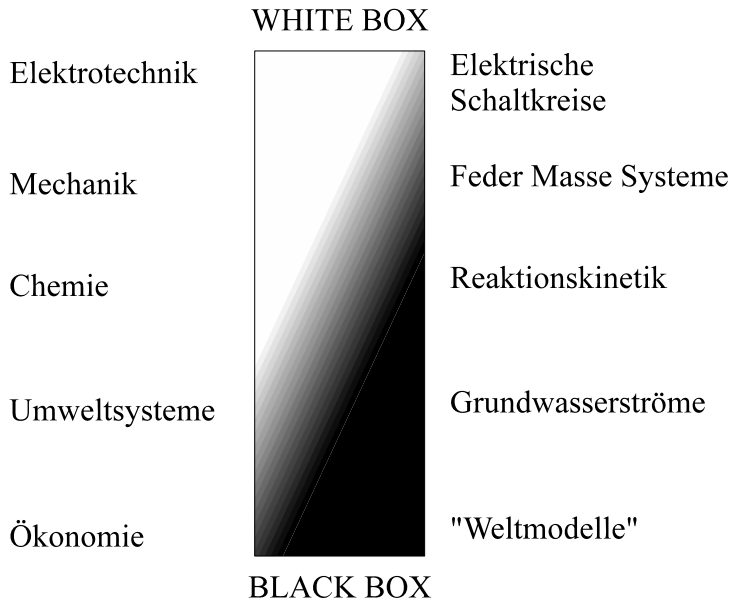


Abbildung 1.2: White Box – Black Box Modellbildung

zeitraubendes Vorhaben ist.

Die Anforderungen der Praxis lassen Simulationssprachen in zwei Kategorien einteilen: Solche für kontinuierliche und solche für diskrete Systeme. Die ersteren sind „maßgeschneidert“ zur Behandlung von Differentialgleichungssystemen, die zweiteren sind zur Untersuchung statistisch verwertbarer Ereignisse (Warteschlangen, Ausfallsraten, Transportvorgänge und dergleichen) geeignet. Als Beispiele für Simulationssprachen für kontinuierliche Systeme wären zu nennen: ACSL, MOSIS, DESIRE, DYMOLA.

Die am häufigsten verwendeten bei Simulation diskreter Systeme sind: SIMAN, GPSS/H, SIMPLE++.

Schließlich stoßen wir auch auf Systeme, die bereits die Verarbeitung beider Typen in sich vereinigt. Wir denken dabei an MATLAB, MAPLE, MATHEMATICA und dergleichen.

1.5 Erweiterung der Modellbildung

Wie wir bereits motiviert haben, wollen wir durch die Einführung einer in einer Datenbank abgebildeten Struktur den Zyklus der Modellbildung und Anwendung der darin enthaltenen Informationen dahingehend erweitern, dass den Me-

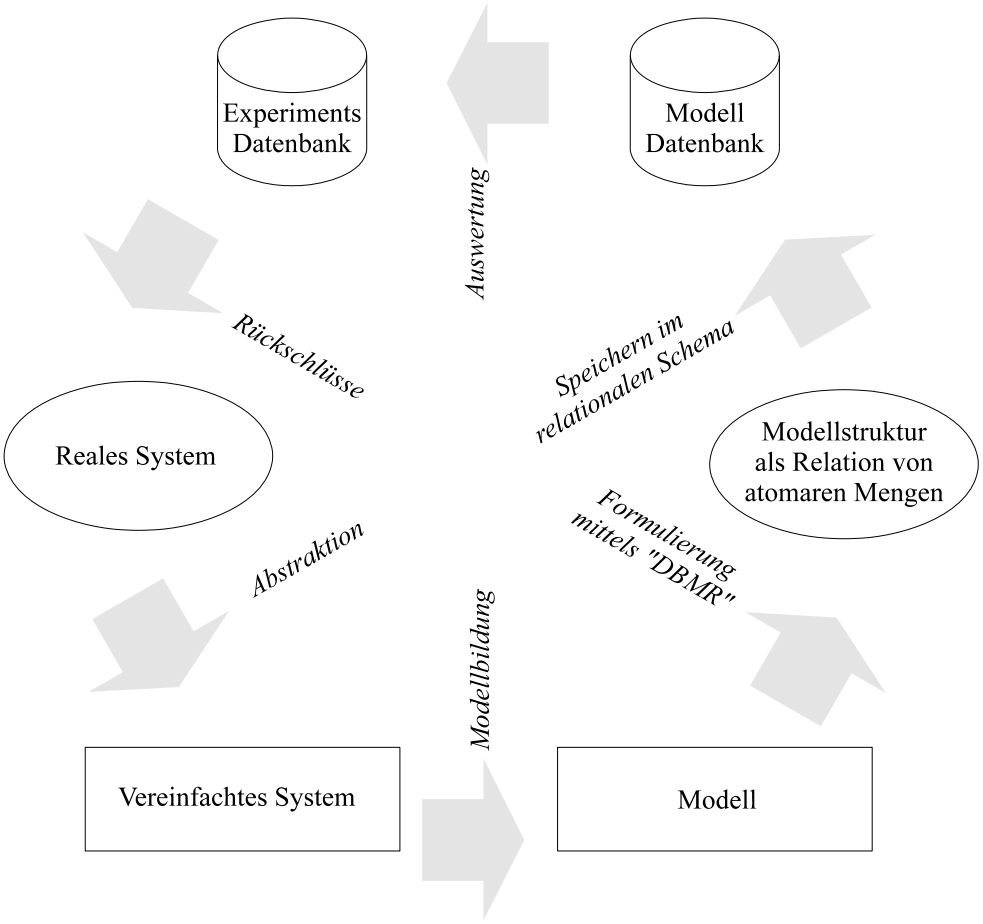


Abbildung 1.3: Der *Life Cycle* von Modellbildung und Simulation

thoden zur Abbildung des Systems in diesem Flussgraph besonders Rechnung getragen wird.

In Abbildung 1.3 ist dieser Zusammenhang, dessen einzelne Punkte in dieser Arbeit natürlich noch genau erläutert werden, dargestellt. Zwar finden wir auch hier den oben beschriebenen „Rückkopplungsmechanismus“, sehen aber, dass in diesem Ablauf der Umformung des mathematischen Modells in die Struktur der Datenbank besonders Rechnung getragen wird.

Kapitel 2

Grundlagen

In diesem Abschnitt wollen wir, aufbauend von den klassischen Konzepten des relationalen Datenmodells, die Überleitung in den weiten Bereich der kontinuierlichen Simulation darstellen.
Wir verwenden die Einbettung der mathematischen Modellbildung in das Relationenmodell, um die Struktur des Modells im „elektronischen System“ zu erhalten. Ebenso erreichen wir durch die Zuordnung von Eindeutigkeitskriterien und funktionalen Abhängigkeiten eine Normalität dieses Modells, sodass ein flexibler Mechanismus zur Auswahl und Kopplung von Modellen beziehungsweise dessen Komponenten zur Verfügung steht.

2.1 Das Relationenmodell

Die Formulierung unserer theoretischen Überlegungen basieren auf den Grundbegriffen der Modellbildung als auch auf denen des Relationenmodells und der Relationenalgebra. Diese sollen hier kurz vorgestellt werden – genauere Ausführungen können wir der hiezugehörigen Spezialliteratur entnehmen.

2.1.1 Grundbegriffe des Entity - Relationship - Diagramms *ERD*

Datenmodelle wollen ein abstraktes Modell eines „Ausschnitts der realen Welt“ schaffen. Dazu formulieren wir folgende Begriffe:

Entitäten: Der Begriff leitet sich vom Englischen *Entities* ab und steht für wohlunterscheidbare Dinge, wie wir sie in der Realität vorfinden. Die Ei-

genschaften dieser Dinge haben jeweils einen genau definierten Wertbereich (Domäne bzw. engl. *Domain*).
Von jeder Entität wollen wir eindeutige Identifizierbarkeit fordern. Dies erreichen wir durch die Auszeichnung bestimmter Attribute einer Entität. Diese Attributkombination bezeichnen wir als *Primärschlüssel* K. Gibt es mehrere solcher Kombinationen, gelten die minimalen als *Schlüssel*, andere als *Schlüsselkandidaten*. Formal können wir nun eine Deklaration einer Entität anschreiben:

$$E = (\text{attr } (E), K)$$

Bezeichnen wir die Attribute einer Entitätsdeklaration E mit A_i, deren Wertebereich mit dom (A_i), dann folgt daraus die formale Beschreibung einer Entität e als Element des kartesischen Produkts aller Domänen:

$$e \in \text{dom } (A_1) \times \ldots \times \text{dom } (A_m)$$

Natürlich ist der Inhalt einer Entität, das *Entity-Set*, im Gegensatz zur obigen Beschreibung zeitveränderlich. Für den Inhalt der Deklaration E zum Zeitpunkt t schreiben wir daher:

$$E^t \subseteq \text{dom } (A_1) \times \ldots \times \text{dom } (A_m)$$

Beziehungen, engl. *Relationships***:** Beziehungen zwischen Entitäten bedürfen nun auch einer formalen Beschreibung. Diese Beziehungen sollen auch (eigene) Attribute besitzen. Wir schreiben daher eine *Relationship-Deklaration* R als

$$R = (\text{ent } (R), \text{attr } (R))$$

Wenn nun die Beziehung r zwischen den Entitäten ent $(R) = (E_1, \ldots, E_k)$ mit den Attributen attr $(R) = (B_1, \ldots, B_n)$ beschrieben werden soll, so gilt

$$r \in E_1^t \times \ldots \times E_k^t \times \text{dom } (B_1) \times \ldots \times \text{dom } (B_n)$$

Für das *Relationship-Set* R^t gilt die Inklusion analog dem *Entity-Set*. Der

Grad der Beziehung R wird mit grad $(R) = k$ bezeichnet.

Um die Häufigkeiten der Entitäten in einer Beziehung zu beschreiben, führen wir den Begriff der *Komplexität* comp (R, E_i) einer Beziehung ein.

$$\text{comp } (R, E_i) = (m, n) \colon (\forall t)(\forall e_i \in E_i^t)|\{r \in R^t | r[E_i] = e_i\}| \Big\{ \begin{matrix} \geq m \\ \leq n \end{matrix}$$

Die Methode, die in der Sprache des ER-Modells formulierbaren Informationen graphisch zu veranschaulichen, ist die Darstellung in einem *Entity-Relationship-Diagramm*. Wir verwenden dazu folgende Bausteine, die in Abbildung 2.1 dargestellt sind:

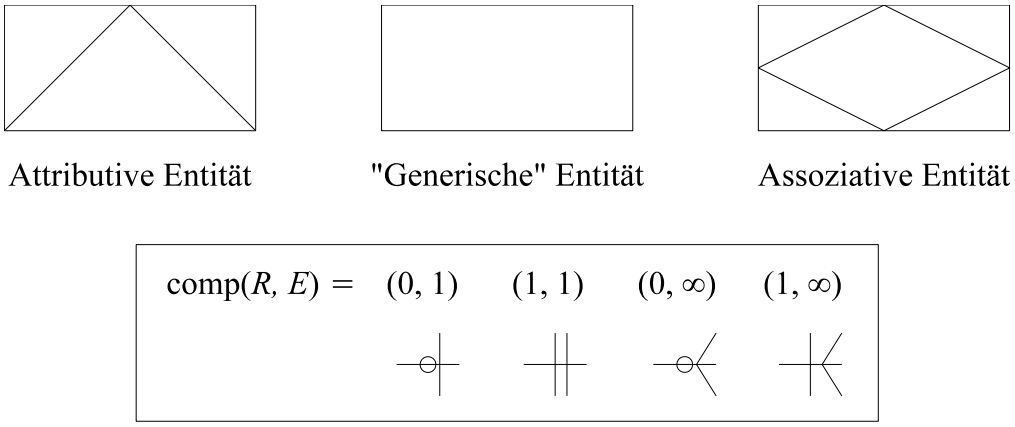


Abbildung 2.1: Die graphischen Elemente eines Entity-Relationship-Diagramms

1. Eine Entity-Deklaration wird durch ein Rechteck, das den Namen der Deklaration enthält, dargestellt. Die Attribute können in Kreissymbolen – mit der Entität durch ungerichtete Kanten verbunden – dargestellt werden. Attribute des Primärschlüssels werden unterstrichen.

2. Ist die Funktion einer Entität ist erster Linie der genaueren Beschreibung einer anderen, so bezeichnen wir diese als *attributive* Entität. In das Rechtecksymbol wird noch ein Dreieck eingeschrieben, um die leichte Unterscheidbarkeit zur regulären Entity-Deklaration zu gewährleisten.

3. Ist eine Beziehung mehrstellig (grad $(R) \geq 2$), wird diese oft als eigene Entiät formuliert. Das Symbol ist ebenso ein Rechteck, in das eine Raute eingeschrieben ist.

4. Schließlich wird die Komplexität einer Beziehung, die wir als ungerichtete Kanten zwischen den Entiäten darstellen, durch verschiedene Endsymbole an den Kanten veranschaulicht. Wiederum verweisen wir auf die Darstellung in Abbildung 2.1.

2.1.2 Die Relationenalgebra

Wir werden nun aus den Elementen des ER-Modells die des Relationenmodells herleiten. Ganz analog halten wir für eine Relation r mit Attributmenge $X = \{A_1, \ldots, A_m\}$ fest, dass diese Teilmenge des kartesischen Produkts aller Domänen ihrere Attribute (= dom (X)) ist. Betrachten wir konkrete Werte, d.h. ein Tupel über X, dann verstehen wir darunter eine Abbildung μ: $X \to$ dom (X) für die $(\forall A \in X)\mu(A) \in$ dom (A) gilt. Die Menge aller Tupel über X bezeichnen wir mit Tup (X), die aller Relationen mit Rel (X).
Innerhalb eines Systems von Relationen benötigen wir nun Operationen, die erlauben, auf konzeptioneller Ebene den Inhalt dieses Relationenschemas (das unter Erfüllung von Konsistenzbedingungen eine relationale Datenbank darstellt) abzufragen oder zu modifizieren. Diese *formalen* Ausdrücke basieren auf der Relationenalgebra bzw. dem Relationenkalkül.
Wir definieren vorerst drei wichtige Operationen, X, bzw. X_i sind Attributmengen, r bzw. r_i sind Elemente von Rel (X) bzw. Rel (X_i).

1. Sei nun $Y \subseteq X$, dann ist $\pi_Y(r) := \{\mu[Y] | \mu \in r\}$ die Projektion von r auf Y. $\mu[Y]$ ist die Einschränkung des Tupels μ auf Y.

2. Ist $A \in X$, $a \in$ dom (A) und $\Theta \in \{<, \leq, >, \geq, =, \neq\}$, dann ist $\sigma_{A\Theta a}(R) := \{\mu \in r | \mu(A)\Theta a\}$ die Selektion von r bzgl. $A\Theta a$.

3. Ebenso verfügen wir über die Mengenoperationen Durchschnitt, Vereinigung und Differenz.

4. Den natürlichen Verbund (*Natural Join*) von Relationen definieren wir folgendermaßen:

$$\bowtie_{i=1}^{n} r_i := \{\mu \in \text{Tup } (\cup_{i=1}^{n}) X_i | (\forall i = 1(1)n)\mu[X_i] \in r_i\}$$

5. Die *Konkatenation* von Relationen entspricht einer Hintereinanderschreibung der Tupel.

$$r_1 * r_2 := \{\mu_1\mu_2 | \mu_1 \in r_1 \wedge \mu_2 \in r_2\}$$

6. Schließlich benötigen wir als letzte Verbindungoperation den *Theta-Join*. Sind $r_i \in \text{Rel}\ (X_i)$, $A_i \in X_i$ für $i = 1, 2$ und $\Theta \in \{<, \leq, >, \geq, =, \neq\}$, dann ist

$$r_1[A_1 \Theta A_2] r_2 := \{\mu_1\mu_2 \in r_1 * r_2 | \mu_1(A_1) \Theta \mu_2(A_2)\}$$

der Θ-*Join* von r_i.

Ist nun eine Menge von Relationenschemata R_i bezüglich der Anwendung von Selektion, Projektion, Bilden des natürlichen Verbundes, Vereinigung und Differenz abgeschlossen, so stellen die daraus resultierenden Ausdrücke die Ausdrücke der Relationenalgebra dar. In der Literatur finden wir auch den Relationen-Tupelkalkül beschrieben, der im Gegensatz zur *prozeduralen* Sprache der Relationenalgebra eine *deskriptive* Sprache darstellt. Relationenalgebra und Relationentupelkalkül haben allerdings gleiche Ausdrucksstärke, sind daher äquivalent.

2.1.3 Normalformen

Führen wir noch den Begriff der „funktionalen Abhängigkeit“ von Attributen ein, dann müssen wir durch entsprechende Dekompositionsopertationen das Relationenschema derart modifizieren, dass Redundanzen und Speicheranomalien ausgeschlossen sind.
Formal schreiben wir, wenn V eine Attributmenge ist, und $X, Y \subseteq Y$ und $r \in \text{Rel}\ (V)$ gilt, dass die funktionale Abhängigkeit $X \to Y$ folgende semantische Bedingung ist:

$$(X \to Y)(r) := \begin{cases} 1 & \text{falls } (\forall \mu, \nu \in r)(\mu[X] = \nu[X] \Rightarrow \mu[Y] = \nu[Y]) \\ 0 & \text{sonst} \end{cases}$$

Die 1. Normalform: In einer 1NF-Relation sind nur einfache (keine men-

genmäßigen) Attributswerte zulässig. Das bedeutet, dass am Kreuzungspunkt einer Kolonne (Attribut) und einer Zeile (Tupel) jederzeit höchstens ein Element vorzufinden ist.

Die 2. Normalform: Eine 2NF-Relation ist dadurch gekennzeichnet, dass jedes nicht dem Schlüssel angehörende Attribut vom Gesamtschlüssel abhängig ist, nicht aber von Schlüsselteilen.

Die 3. Normalform: Eine 3NF-Relation zeichnet aus, dass jedes nicht dem Schlüssel angehörende Attribut vom Gesamtschlüssel abhängig ist, nicht aber von Schlüsselteilen (2NF). Weiters sind keine funktionalen Abhängigkeiten zwischen Attributen erlaubt, die nicht als Schlüsselkandidaten in Frage kommen.

Die 4. Normalform: Die 4NF-Relation ist dadurch gekennzeichnet, dass sie 3NF ist und keine mehrwertigen Abhängigkeiten aufweist.

2.2 Die hierarchische Modellbildung

Viele Aufgaben in der Simulation sprengen den „akademischen“ Umfang – sie sind groß in der topologischen Struktur und im mathematischen Ergebnis. Oft sind diese Aufgaben derartig aufgebaut, dass eine Modellbildung bis in die oberste Strukturebene nicht erforderlich ist. In diesem Fall identifizieren wir das Modell als hierarchisch gegliederter Aufbau von geeigneten Komponenten (vergleiche Abbildung 2.2).
Diese Komponenten, von denen wir voraussetzen, dass eine Wiederverwendbarkeit gegeben ist, stellen einen Teilaspekt des betrachteten Systems dar, der je nach der gewünschten Strukturtiefe entsprechende Bedeutung aufweist. Wir wissen, dass die Feinheit des Modells vom Zweck der Simulation abhängt.
Kursorisch werden wir einige Mechanismen zur hierarchischen Modellbildung anführen.

Makropräprozessoren: Viele Simulationsprachen erlauben es, Subsysteme, d.h. Komponenten, als Makros zu formulieren. Herkömmliche Programmiersprachen der dritten Generation verwenden dafür den Begriff der Pro-

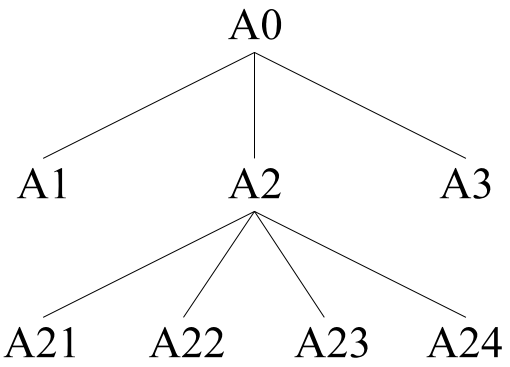


Abbildung 2.2: Anschaulicher Aufbau einer Modellhierarchie von $A0, \ldots, A24$ anhand eines Baumes

zedur oder Funktion. Diese Komponenten werden als eigenes Modell mit Parameterschnittstelle formuliert und von den Komponenten der darüberliegenden Hierarchieebene aufgerufen.

Der Makropräprozessor ersetzt vor jeder weiteren Verarbeitung rein textuell jeden Aufruf eines derartig formulierten Submodells durch dessen Programmtext – erst dann erfolgt eine Berechnung des Systems.

Modulare Präprozessoren: Wird ein Modell nicht in einer Simulationssprache, sondern mittels eines Präprozessors verfasst, der eigene modulare Konzepte beinhaltet, dann ist das System von Modellen und Submodellen in seiner Hierarchie außerhalb der Berechnung gelagert. Es muss auch der Quelltext der Submodelle nicht in der Simulationspsrache vorliegen. Innerhalb des Präprozessors wird daher ein „modularer Zustandsraum“ aufgebaut.

Modelle sehr großer Systeme: Wir wollen annehmen, es sei ein Modell entworfen, das sehr groß ist – in gleicher Größenordung ist natürlich auch der hierarchische Aufbau des Systems komplex. Würde eine Änderung des Systems erforderlich sein, dann erfolgt – innerhalb der gewählten Plattform – ein Neuaufbau des gesamten ausführbaren Codes. Diese Vorgangsweise stellt einen enormen Verbrauch von Ressourcen dar. Wir ziehen daher eine separate Compilierung der Modelle und Submodelle vor. So ist nur ein neuer Aufbau der Zusammenhänge der modifizierten Teile erforderlich.

2.3 Bondgraphen

Einige Grundbegriffe der Bondgraphen-Modellbildung sollen hier nun kurz dargestellt werden, denn eine der Anwendungsbeispiel von *DBMR* wird auch diese Art der Modellbildung sein. Deren Struktur erlaubt es, die Methoden des flexiblen Ein/Ausgabemechanismus anzuwenden.

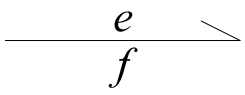


Abbildung 2.3: Das „Bond“ Symbol

Die graphische Darstellung eines „Bonds“ zeigt Abbildung 2.3. Dieser ist dargestellt als gerichteter Pfad, der den Leistungsfluss innerhalb eines Systems angibt. Dieser Leistungsfluss ist als Produkt zweier Größen zu sehen, der *effort variable* und der *flow variable* - diese sind mit e und f bezeichnet. Für diese variablen Größen wird in Knotenpunkten eines aus „Bonds“ aufgebauten Graphen die Gültigkeit der Kirchhoffschen Gesetze vorausgesetzt.
Ein Bond ist entweder mit einem Systemelement verbunden (*single port elements*) oder mit anderen Bonds. Wir unterscheiden zwischen der 1- und der 0-Verknüpfung (siehe Abbildung 2.4). In einer 0-Verknüpfung sind alle *effort* Variablen gleich, während die Summe der *flow* Variablen gleich 0 ist. Anders gelagert ist die Rechenvorschrift für die 1-Verknüpfung. Hier sind die *flow* Variablen gleich, die Summe der *effort* Variablen ist gleich 0. Gleichartige Verknüpfungen können zu einer einzigen zusammengezogen werden, daher sind die Verknüpfungen, die durch einen Bond verbunden werden, stets von unterschiedlichen Typen.
Im Unterschied zu vielen anderen Modellbildungsmethodologien, erhält ein Bondgraph die topologische Struktur eines Systems. Wie dann aber bei Berechnung anzusetzen ist, gibt die Kausalität eines Bondgraphen an. Ein kleiner Strich an einem Ende des Bonds zeigt an, welche der Variablen e und f Ein- oder Ausgangsgröße ist. Wir wollen einige Arten von Kausalitäten anführen:

- Unabdingbar erforderliche Kausalitäten, um die Quelle des *flow* oder *effort* and zugeben (SE, SF)

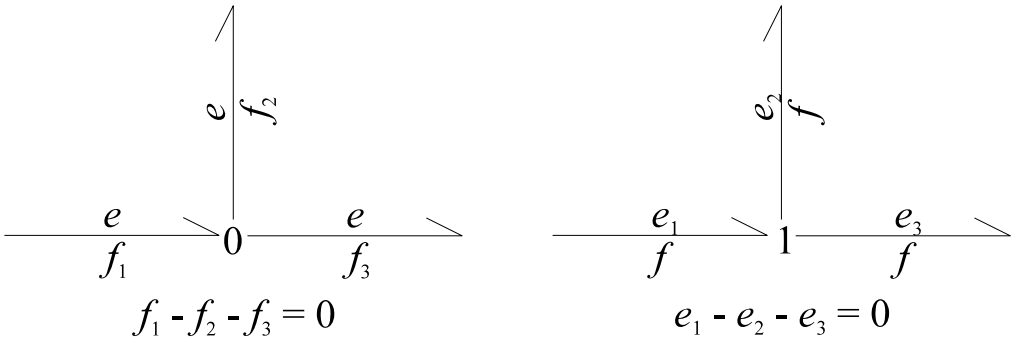


Abbildung 2.4: Die 0-Verknüpfung und die 1-Verknüpfung

- Beschränkte Kausalitäten für Umformer: Transformator (TF), Gyrator (GY), Modulierter Transformator (MTF), modulierter Gyrator (MGY)
- Kausalitäten für Integral (I) und Differential (D)
- beliebige Kausalitäten

Wir können nicht ausschließen, dass unter bestimmten Bedingungen erforderliche Kausalitäten nicht erfüllt werden können. Dies führt dazu, dass wir von einem *noncasual system* sprechen. Bei der Untersuchung eines derartigen Systems werden wir auf algebraische Schleifen stoßen.
Basierend auf den Grundforderungen eines Bondgraphen, nämlich der Energieerhaltung und der Kontiunität des Leistungsflusses, können ideale (nämlich verlustlose) Umformer definiert werden. Mit ihrer Einbindung in einen Bondgraph seien hier der ideale Transformator (Abbildung 2.5) und der ideale Gyrator (Abbildung 2.6) angeführt. Wir beachten, wie bei unterschiedlicher Kausalität die Bestimmungsgleichungen modifiziert werden.

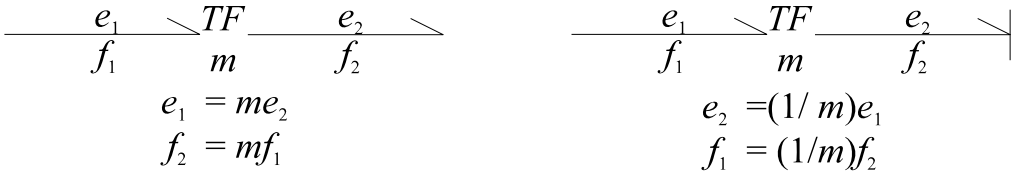


Abbildung 2.5: Bondgraph mit Kausalitäten für den idealen Transformator

Werden einzelne System von Bondgraphen an vorgesehenen Verknüpfungspunkten verbunden, so bildet sich ein Hierarchie dieser Systeme, die ebenso auch in den Modellen zu finden ist.

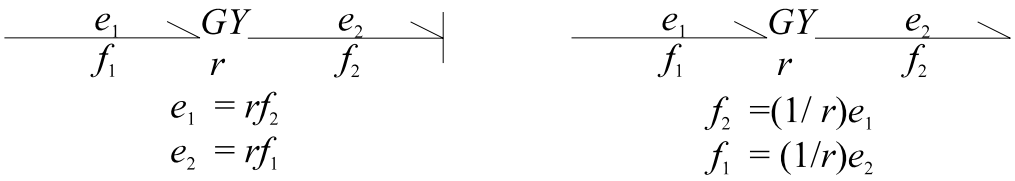


Abbildung 2.6: Bondgraph mit Kausalitäten für den idealen Gyrator

2.4 Die hybride Modellbildung

Wir bezeichnen mit „hybrider Modellbildung“ die Verbindung verschiedener Modellbildungsmethoden, in Hinblick auf die Kombination der Vorzüge dieser Verfahren. Sowohl im kontinuierlichen als auch diskreten Bereich findet diese Vorgangsweise Anwendung.

In der kontinuierlichen Modellbildung und Simulation betrachten wir Systeme, deren Modell als System von Differenzialgleichungen aufgestellt wird. Bei den Berechnungsmethoden für digitale Rechenanlagen ist es allerdings erforderlich, die freie Variable (= Zeit) zu diskretisieren. Der „Zahlenstrahl“ dieser Variable wird in – je nach Berechnungsmethode auch unterschiedlich große – Intervalle geteilt. Aufbauend auf der Lösung von vorhergehenden Intervallen wird der aktuelle Funktionswert errechnet.

Im Fall der diskreten Modellbildung und Simulation ist Verhalten des Systems keineswegs über den ganzen Definitionsbereich der freien Variable festgelegt. Wir kennen den Zustand des Systems nur zu bestimmten Zeitwerten und allfällig einige Übergangswahrscheinlichkeiten. Wird das System berechnet, werten wir eine Tabelle der Ereignisse und der erforderlichen Wahrscheinlichkeiten aus. Daraus können wiederum neue Ereignisse entstehen.

Betrachten wir nun ein Modell m mit hybriden Ansatz.

- Im Intervall $[t_0, t_1)$ gilt die Beschreibung des Differntialgleichungssystems D_1^m
- Im Intervall $[t_1, t_2)$ gilt D_2^m
- Mit einer gewissen Übergangswahrscheinlichkeit p_{t_2} erfolgt an t_2 die Beschreibung des Systems durch D_3^m, mit $1 - p_{t_2}$ durch D_4^m.

Im hybriden Sinn sind diese „Umschaltvorgänge“ teil des Modells und nicht des

experimental frame. Diese Sichtweise ist auch Teil von *DBMR*.

2.5 Die Datenbank basierende Modellverknüpfung - das mathematische Modell

Wir werden in diesem Abschnitt die Grundstrukturen kennenlernen, die, zueinander in Relation gesetzt, ein Modell, ein Experiment, hierarchische Modelle und andere Objekte bilden. Im Sinne einer leichten Identifizierbarkeit werden wir für dieses Konzept die Abkürzung „DBMR“ *Databased Model Relation* verwenden.

Obwohl bei der Erstellung dieser Theorien in erster Linie vom umfassenden Gebiet der kontinuierlichen Modellbildung ausgegangen wurde, wird doch die explizite Anführung der Zeitabhängigkeit bewusst vermieden und statt dessen wird der Begriff der „frei wählbaren Parameter“ im Gegensatz zu den „atomaren Mengen“ verwendet. Es wird davon ausgegangen, dass die Zeit auch als „frei wählbarer Parameter“ aufzufassen ist. Wir erreichen dadurch eine „generalisierte Relationendarstellung“ der meisten durch die Simulation darstellbaren Gebiete.

Die Methoden, die durch die Einbettung in das System der Relationenalgebra, die die Basis eines relationalen Datenbanksystems darstellt, ermöglicht werden, werden Mechanismen wie den „flexiblen IO Mechanismus“ und die „datenbank-basierende Modellverknüpfung“ selbst herleiten. Neben der Darstellung der Struktur eines Modells werden wir die Anwendung von zueinander in Relation stehenden Mengen als Mechanismus zur Modellinterpolation beziehungsweise Modelllinearisierung kennenlernen. Dadurch ist, wenn im mathematischen Modell hinreichende Stabilität gegeben ist, eine schnelle, dann auch in gewissen Maße genaue, Auswertung für andere frei wählbare Parameterkonstellationen möglich.

In den nachfolgenden Abschnitten werden wir die Definition einer formalen Sprache zur Instanzierung und Auswertung von Modellen und Modellverknüpfungen beschreiben. Neben diesen Sprachelementen werden wir auch jene kennenlernen, die bei der Durchführung von Experimenten auf die oben erwähnte

Art eine Evaluierung des Modells beschreiben.

2.5.1 Einige Definitionen

Um ein Modell als Verknüpfung von Mengen zu beschreiben, müssen wir einige dieser algebraischen Strukuren definieren, um diese geeignet in Relation setzen zu können. Diese Strukturen müssen folgenden Aspekten genügen:

- Identifizierung des Modells
- Kopplung von Modellen
- Identifizierung der Auswertung des Modells
- Verschiebung der auswertungsabhängigen Modellbeschreibung in die Experimentierphase (*hybride Modellbildung*, *flexibler IO Mechanismus*)
- Parallele Ausführung von Modellauswertungen

Da ein Modell, das insbesondere im dynamischen Bereich angesiedelt ist, die Verarbeitung von Eingangs- zu Ausgangsgrößen beschreibt, wollen wir die **von frei wählbaren Parametern abhängige Menge** einführen. Mit diesem Begriff formulieren wir einen der Grundbausteine der modellbeschreibenden Struktur. Wir erkennen, dass es hier notwendig ist, die Abhängigkeit von Parametern zu fordern, da dies eine fundamentale Eigenschaft von mathematischen Beschreibungen darstellt:

$$A_i \in A_I \colon a_i \in A_i \text{ mit } a_i(\hat{p}) = f(a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_m, \hat{p})$$

$\hat{p}$ ist ein Vektor von frei wählbaren Parametern.
Wir benötigen aber noch allgemeinere Bausteine der Datenbank-basierenden Modellverknüpfung. Diese sollen einerseits für die Aufnahme der Modellstruktur andererseits für die Speicherung der Modellergebnisse geeignet sein.
Wir führen dazu den Begriff der **atomaren Menge** ein. Wir folgern aufgrund der geforderten Eigenschaften der **atomaren Menge**, dass deren Elemente ebenfalls Mengen sein können.

Seien nun $A_1, A_2, \ldots, A_m$ **atomare Mengen**, dann bildet das kartesische Produkt dieser eine Menge, deren Elemente, die m-Tupel, als Beschreibung von Modellgleichungen gelten. Wir verwenden hier den Begriff der **Zustandsmenge** Z.

$$Z = A_1 \times A_2 \times \ldots \times A_m$$

Mit der **Zustandsmenge** bilden wir eine „Gesamtheit aller Modellgleichungen“. Da eine mathematische Beschreibung eines Systems aus voneinander abhängigen Gleichungen besteht, müssen wir aus dem in Beziehung setzen von Modellgleichungen wiederum eine Menge bilden, deren Elemente ein jeweiliges System beschreiben. Dazu verwenden wir den Begriff des **Zustandsraums**; wir wollen diesen mit $\bar{Z}$ bezeichnen und wie folgt definieren:

$$\bar{Z} = Z_{i_1} \times Z_{i_2} \times \ldots \times Z_{i_m} \text{mit } i_k \in \mathbf{N}$$

2.5.2 Einige spezielle Zustandsmengen

Wir werden nun, um geeignete Spezifikationen zu ermöglichen, den Begriff der **Zustandsmenge** erweitern und einige Typen von Zustandsmengen definieren. Diese werden funktionale Aufgaben übernehmen, die dann für die Beschreibung der Datenbank basierenden Methoden von Bedeutung sind. Wir denken an die Operatoren der Projektion, Selektion und des Joins von Mengen.
Um Abhängigkeiten von Parametern in den oben erwähnten funktionalen Bereich zu übertragen, wollen wir die **parametervariante Menge n-ter Ordnung** V_n einführen. Wir beabsichtigen mit dieser mengentheoretischen Struktur „Konstante“ in die Modellverknüpfung einzubringen; die geeignete Auswertung liefert zu einem Parametervektor $\hat{p}$ ein n-Tupel zurück, das wiederum als **Zustandsmenge** gesehen werden kann.

$$\begin{aligned} V_n &= \{(p_1, v_1^1, \ldots, v_n^1), (p_2, v_1^2, \ldots, v_n^2), \ldots\} \\ V_n(\hat{p}) &= \{(p_k, v_1^k, \ldots, v_n^k) | \hat{p} = p_k\} \end{aligned}$$

Besonders für den flexiblen Ein-/Ausgabemechanismus (*flexible input/output mechanism*) ist es erforderlich, in Abhängigkeit von den ausgewählten Zustands-

mengen eine resultierende daraus zu wählen. Diese muss aus der Definitionsmenge (dem **Zustandsraum**) in funktionaler Berücksichtigung der Abbildungsfunktion σ und dem frei wählbaren Parametervektor p ausgewählt werden.

$$\sigma: \bar{Z} \times P \to Z: (z_{i_1}, z_{i_2}, \ldots, z_{i_k}, p) \to z_{i_j} \quad \text{mit } j \in 1, \ldots, k$$

2.5.3 Aspekte der Datenmodelle

Im Rahmen der Frage, ob Modellbildung und -simulation, vor allem im kontinuierlichen Bereich, eine prinzipiell verallgemeinerte Lösung von Differentialgleichungssystemen sei, sehen wir, dass das Konzept der **atomaren Menge** auch Elemente verschiedenster Typen haben kann. Wir stellen fest, dass über die Gestalt dieser Elemente keinerlei einschränkende Ausssage gemacht wurde. Als *Elementtypen* erscheinen folgende geeignet:

- eindimensionale Größen
- mehrdimensionale Größen
- Festhalten von Ergebnissen mit der Kenntnis des Urspungs (*Experimentsdatenbank*)
- Modelle
- Modellverknüpfungen
- Modellauswertungen

Wir erkennen, nachdem wir einige Sachverhalte der folgenden Abschnitte vorwegnehmen, dass die *Experimentsdatenbank* eine attributive Entität der assoziativen Entität *Simulation* ist; daher ist die Zuordenbarkeit eindeutig. Deren Attribute bestehen aus den Elementen der atomaren Mengen, die für das Modell relevant sind. Durch Bilden von Linearkombinationen von derartigen Tupeln werden weitere Lösungen mit natürlich eingeschränkter Genauigkeit gebildet.

Herleitung des *Modells*

Wir betrachten ein *Modell* im klassischen mathematischen Sinn und werden dessen Elemente mit den nun definierten Begriffen beschreiben. Nachdem wir die

allgemeinen Zusammenhänge dieser Struktur erkannt haben, werden wir diese in mittels geeigneter Definitionen in das Begriffsgebäude von *DBMR* einbetten.

$$
\begin{aligned}
\text{Modell} \quad & m(\leftarrow: a, b, c \rightarrow: x, y, z \text{ Parametervektor } p) \\
\frac{\mathrm{d}x}{\mathrm{d}p} &= f_1(x, y, z, a, b, c, p) \\
\frac{\mathrm{d}y}{\mathrm{d}p} &= f_2(x, y, z, a, b, c, p) \\
\text{Wenn} \quad & \beta(x, y, z, \frac{\mathrm{d}x}{\mathrm{d}p}, \frac{\mathrm{d}y}{\mathrm{d}p}, \frac{\mathrm{d}z}{\mathrm{d}p}, a, b, c, p) \\
\frac{\mathrm{d}z}{\mathrm{d}p} &= f_3(x, y, z, a, b, c, p) \\
\text{Wenn} \quad & \gamma(x, y, z, \frac{\mathrm{d}x}{\mathrm{d}p}, \frac{\mathrm{d}y}{\mathrm{d}p}, \frac{\mathrm{d}z}{\mathrm{d}p}, a, b, c, p) \\
\frac{\mathrm{d}z}{\mathrm{d}p} &= f_4(x, y, z, a, b, c, p)
\end{aligned}
$$

Wir erkennen folgende fundamentale Beziehungen und Entsprechungen:

- Die ein- bzw. ausgehenden Größen sind als Elemente **atomarer Mengen** klassifizierbar. Sie sind aus deren Domänen ausgewählt.

- m ist im Sinne eindeutiger Identifizierbarkeit durch die Art und Typ der ein- und ausgehenden Größen bestimmt. (so sind in diesem Beispiel a, b und c als eingehende Gößen angegeben, x, y und z sind die ausgehenden Größen. p stellt den frei wählbaren Parametervektor dar. Wir werden diese Eigenschaft dazu Nutzen, der Entsprechung in der Relationenalgebra durch die Einführung geeigneter Auswahlfunktionen und -operatoren einen Schlüsselkandidaten zuzuweisen.)
  Eine Teilgleichung des Modells stellt daher ein Tupel des **Zustandsraums** dar. Im Falle von z erfolgt die Zuordnung folgenden Tupels:

$$z \rightarrow (f_4, x, y, z, a, b, c, p)$$

- Die Entscheidungsfunktionen β und γ sind Auswahlfunktionen. Dadurch werden für das Element z des Modellraums m die Modellgleichungen zugewiesen.

Wir wollen jetzt diese Erkenntnisse über die notwendigen Bausteine der

Begriffe des *Modells* und der *Modellgleichungen* in formaler Art beschreiben.
Eine **Modellgleichung** stellt eine Abbildung dar, die der „linken“ Seite den Ausdruck der „rechten“ Seite zuweist. Wir bezeichnen diese Abbildung mit κ.

$$\begin{aligned} \text{Abbildung } \kappa\colon A \to \bar{Z}\colon a \to (z_{i_1}, z_{i_2}, \ldots, z_{i_k}) \\ i_1, i_2, \ldots, i_k \in \mathbf{N} \end{aligned}$$

A benennt im Sinne der von uns gewählten Notation eine **atomare Menge**, $\bar{Z}$ ist der **Zustandsraum**. Es wird dem Element a einer atomaren Menge der „Wert“ des Tupels $z_{i_1}, \ldots z_{i_k}$ zugewiesen.
Ein *Modell* besteht natürlich aus einer Menge verschiedenster *Modellgleichungen*, die mitunter in höchst komplexen Zusammenhängen stehen. Wir werden daher zur Beschreibung die Zuordnung von „Parametern“ zu ihren Modellgleichungen als davon eindeutig ableitbare Gößen nutzen. Das Modell sei mit m bezeichnet; so ist dieses eine eindeutige Abbildung:

$$\begin{aligned} m\colon Z \to \bar{Z} \quad &: \quad (a_{i_1}, a_{i_2}, \ldots a_{i_k}) \to (z_{i_1}, z_{i_2}, \ldots z_{i_k}) \\ i_1, i_2, \ldots, i_k \quad &\in \quad \mathbf{N} \end{aligned}$$

Wir erkennen, dass mit der Bezeichnung m und der Angabe der Parameter $(a_{i_1}, a_{i_2}, \ldots, a_{i_k})$, die wir als Tupel aus Elementen der **Modellparametermenge** P bezeichnen, die Forderung nach eindeutiger Identifizierbarkeit erfüllt ist. Dies werden wir für weitere Beschreibungen benötigen.

2.6 Weitere Mechanismen

Mit der Klassifikation des Modells m, der Modellgleichungen als Zuordnungsfunktion κ und der Auswahlfunktion σ haben wir Klarheit in die mengenbasierende Struktur der mathematischen Modellbildung gebracht.
Diese kann aber nicht auf Flexibilität in der Auswahl der verschiedenen Typen von Modellgleichungen verzichten, um sämtliche Aspekte abzudecken. Es ist nun zu untersuchen, wie diese Abbildungen in diese Struktur aufgeprägt werden können.

2.6.1 Die Modellgenerierung

Ein Modell bildet die Parametermenge auf Modellgleichungen ab (Es erfolgt eine Zuordnung von Elementen atomarer Mengen auf Auszüge des Zustandsraums). Ist es aber erforderlich, diese Abbildung in Abhängigkeit der tatsächlich verwendeten Elemente der Parametermenge zu steuern, benötigen wir einen Mechanismus, der dann die endgültige Auswahl der Gleichungselemente durchführt. Es wird ein „allumfassendes“ Modell beschrieben, woraus für den zum Beispiel untersuchenden Fall die geeigneten kombiniert werden. Diese Vorgangsweise werden wir als **Modellgenerierung**, den Abbildungsmechanismus als **flexiblen input/output Mechanismus** bzw. *flexible input/output mechanism – FIM* bezeichnen.
Es sei nun m ein Modell:

$$m(a_1, a_2, \ldots, a_k) \to \bar{Z}$$

Wir führen jetzt die Auswahlfunktion φ_m ein. Diese muss als weiterer Parameter der Funktion m, dem Modell, gesehen werden.

$$m(a_1, a_2, \ldots, a_k, \varphi_m) \to \bar{Z}_{\varphi_m}$$

Exaktheit fordert nun eine Beschreibung der neuen Begriffe φ_m und $\bar{Z}_{\varphi_m}$.

Die Auswahlfunktion φ_m: Diese Funktion $\varphi_m \colon P \to \bar{Z}$ bildet die Parametermenge P des Modells m auf die zu φ_m assoziierte Bildmenge $\bar{Z}_{\varphi_m}$ ab. Wir verdeutlichen dies:

$$\begin{aligned} P &= \{a_1, a_2, a_3, \ldots, a_k\} \\ a_1|_{\varphi_m^1} &\to z_1^1, \ldots, a_k|_{\varphi_m^1} \to z_k^1 \\ &\vdots \\ a_1|_{\varphi_m^l} &\to z_1^l, \ldots, a_k|_{\varphi_m^l} \to z_k^l \end{aligned}$$

Wir folgern daraus, dass $\varphi_m = \bigcup_{i=1}^{l} \varphi_m^i$.

Die zu φ_m assoziierte Bildmenge $\bar{Z}_{\varphi_m}$: Ein Modell gilt als Abbildung von

der Parametermenge P auf Elemente des Zustandsraums $\bar{Z}$. Verwenden wir die Auswahlfunktion φ_m, die Bildmenge zu spezifizieren, wird diese davon über φ_m funktional abhängig. Es gilt daher:

$$m(P, \varphi_m) \to \bar{Z}_{\varphi_m} \subseteq \bar{Z}$$

bzw. mit obigen Bezeichnungen

$$\bar{Z}_{\varphi_m} = \{z_1^1, \ldots z_k^1, \ldots, z_1^l, \ldots, z_k^l\}$$

Einige Erläuterungen

Wir wollen anhand eines Beispiels die noch sehr abstrakte Definition der Auswahlfunktion φ_m in ein klareres Licht tauchen. Wiederum bezeichnen wir das Modell mit m:

$$\begin{array}{rcl}
m(a_1, a_2, a_3, \varphi_m) & & \text{mit} \\
\varphi_m(\leftarrow\!: a_1, a_2, \rightarrow\!: a_3) & \longrightarrow & a_1 \to (\xi_1, a_1, a_2, a_3) \\
 & & a_2 \to (\xi_2, a_1, a_2, a_3) \\
 & & a_3 \to (\xi_3, a_1, a_2, a_3) \\
\varphi_m(\leftarrow\!: a_1, \rightarrow\!: a_2, a_3) & \longrightarrow & a_1 \to (\nu_1, a_1, a_2, a_3) \\
 & & a_2 \to (\nu_2, a_1, a_2, a_3) \\
 & & a_3 \to (\nu_3, a_1, a_2, a_3) \\
\varphi_m(\leftarrow\!: a_3, \rightarrow\!: a_1, a_2) & \longrightarrow & a_1 \to (\mu_1, a_1, a_2, a_3) \\
 & & a_2 \to (\mu_2, a_1, a_2, a_3) \\
 & & a_3 \to (\mu_3, a_1, a_2, a_3)
\end{array}$$

Das Tupel (a_1, a_2, a_3) ist die Parmetermenge P, Elemente des Zustandsraums $\bar{Z}$ sind die Tupel (ξ_k, a_1, a_2, a_3), (ν_k, a_1, a_2, a_3) und (μ_k, a_1, a_2, a_3), wobei $k = 1, 2, 3$. In Hinblick auf m sind genau diese Tupel $\bar{Z}_{\varphi_m}$.
Berücksichtigen wir nun, dass ein Element der Parametermenge P sowohl als ein- ($\leftarrow$) als auch ausgehende ($\rightarrow$) Größe vorkommen kann, ist auch dies mittels der Auswahlfunktion φ_m berücksichtigbar. Hat P k Elemente, von denen l mit $l \leq k$ über φ_m ausgewählt werden können, kann es bis zu 2^l Bildmengen als

Teilmengen von $\bar{Z}$ geben, deren Vereinigung $\bar{Z}_{\varphi_m}$ bildet. Bei obigen Beispiel sind nicht alle Kombinationen „versorgt“, es wären nach dieser Abschätzung bis zu acht Teilbildmengen möglich.
Wir beachten aber, dass bei der Einführung der Teilfunktionale φ_m^k keine Einschränkung über den Wertevorrat gemacht wurde.

2.6.2 Die Verbindung von Modellen

Fundamentale Verfahren zur Abstraktion betrachten zur Lösung eines Problems M eine Aufteilung des Problems in Teile $m_1, m_2, \ldots, m_k$. Diese müssen, um die Lösung zu finden, geeignet verbunden werden. Prinzipielle Mechanismen wurden in den vorigen Abschnitten bereits vorgestellt. Die in *DMBR* Vorgesehenen werden einzeln vorgestellt und dann zur Definition der Relationenalgebra herangezogen.

Die „innere Verbindung“ von Modellen *Model-Join*

Erfolgt die Verbindung auf der Ebene der atomaren Mengen, sprechen wir von der *inneren Verbindung*. Mit dieser „Verknüpfungsvorschrift“ wird die Vereinigung der Tupel aus $\bar{Z}$, die die zu verbindenden Modelle bilden, zu neuen Tupeln zusammengefasst, die dann das „Ergebnismodell“ bilden.
Seien nun m_1 und m_2 Modelle mit Parametermengen $P_1 = (a_1^1, \ldots, a_i^1, \varphi_{m_1})$ und $P_2 = (a_1^2, \ldots, a_j^2, \varphi_{m_2})$, dann ist m_3 mit Parametermenge P_3 und Auswahlfunktion φ_{m_3} die *Modellverknüpfung - Model Join* von m_1 zu m_2 mit Verknüpfung von
$J_1 = (a_{i_1}^1, \ldots, a_{i_n}^1) = J_2 = (a_{j_1}^2, \ldots, a_{j_n}^2)$. Wir wenden folgende Verknüpfungsvorschrift an: In jeder Modellgleichung von m_1 wird $\{a_{i_1}^1, \ldots, a_{i_n}^1\}$ durch diejenige von $\{a_{j_1}^2, \ldots, a_{j_n}^2\}$ ersetzt.
Unmittelbar daraus folgt, dass

$$\begin{aligned} P_3 &= (P_1 \backslash J_1) \cup P_2 \\ \varphi_{m_3} &= \varphi_{m_2} \circ \varphi_{m_1} \end{aligned}$$

gilt. Kurz schreiben wir:

$$m_3 = m_1 \bowtie_{J_1=J_2} m_2$$

Einige Erläuterungen

Wir betrachten zwei Modelle, m_1 und m_2. Wir lassen die Anwendung von Auswahlfunktionen φ_{m_k} ausser acht, um den Mechanismus zu verdeutlichen (p stellt den frei wählbaren Parametervektor dar):

$$\begin{aligned} m_1(a_1, a_2, a_3)\colon a_1 &\rightarrow (g_1, a_1, a_2, a_3, p) \\ a_2 &\rightarrow (g_2, a_1, a_2, a_3, p) \\ a_3 &\rightarrow (g_3, a_1, a_2, a_3, p) \end{aligned}$$

m_2 habe vier Elemente in der Modellparametermenge:

$$\begin{aligned} m_2(b_1, b_2, b_3, b_4)\colon b_1 &\rightarrow (f_1, b_1, b_2, b_3, b_4, p) \\ b_2 &\rightarrow (f_2, b_1, b_2, b_3, b_4, p) \\ b_3 &\rightarrow (f_3, b_1, b_2, b_3, b_4, p) \\ b_4 &\rightarrow (f_4, b_1, b_2, b_3, b_4, p) \end{aligned}$$

Wir bilden nun m_3 als Modellverknüpfung *Model Join* von m_1 zu m_2 mit $(a_1, a_2) = (b_1, b_2)$ oder in oben gezeigter kurzer Notation:

$$m_3 = m_1 \bowtie_{(a_1,a_2)=(b_1,b_2)} m_2$$

Das Ergebnis sieht, nachdem wir $a_1 = b_1 = (f_1, b_1, b_2, b_3, b_4, p)$ und $a_2 = b_2 = (f_2, b_1, b_2, b_3, b_4, p)$ gesetzt haben, wie folgt aus:

$$\begin{aligned} m_3(a_3, b_1, b_2, b_3, b_4)\colon a_1 &\rightarrow b_1 \\ a_2 &\rightarrow b_2 \\ a_3 &\rightarrow (g_2, (f_1, b_1, b_2, b_3, b_4, p), (f_2, b_1, b_2, b_3, b_4, p), a_3, p) = \\ &\quad = (g_2, b_1, b_2, a_3, p) \\ b_1, \ldots, b_4 &\quad \text{wie in } m_2 \text{ beschrieben!} \end{aligned}$$

Da der Zustandsraum ein kartesisches Produkt von atomaren Mengen darstellt, kann statt einer derartigen Menge ein Tupel des betrachteten Zustandsraums gesetzt werden. Wir erhalten dadurch wiederum ein Tupel, das Element eines geeigneten Zustandsraums ist. In Abbildung 2.7 sehen wir diesen Ersetzungsmechanismus angewandt.

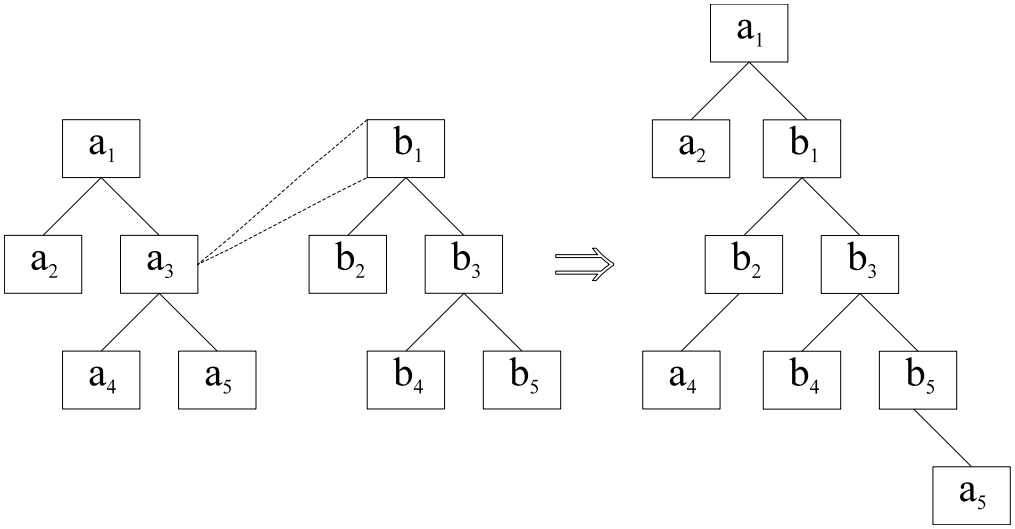


Abbildung 2.7: Darstellung der inneren Verknüpfung. Ein Element – a_3 – des Tupels $(a_1, a_2, a_3, a_4, a_5)$ wird durch das Tupel $(b_1, b_2, b_3, b_4, b_5)$ ersetzt

2.6.3 Die „äußere Verbindung“ von Modellen *Union coupled / Union Uncoupled*

Neben der „direkten“ oder „inneren“ Verbindung von Modellen untersuchen wir in diesem Abschnitt die Art von Verknüpfung, bei der die Modellbeschreibungen zu einer Einheit verbunden werden.

Die ungekoppelte Vereinigung

Seien $m_1, m_2, \ldots, m_k$ Modelle mit Parametermengen $P_1, P_2, \ldots, P_k$, und Auswahlfunktionen φ_{m_i}, $i = 1, \ldots, k$. Dann ist $\bar{m}$ die *ungekoppelte Vereinigung* der Modelle $\bar{\bigcup}_{j=1}^{k} m_j$ mit Parametermenge $\bigcup_{j=1}^{k} P_j$ und den Modellgleichungen $\bigcup_{j=1}^{k} P_j \to \bar{Z}$. Wir verstehen darunter die Zusammenfassung von Modellen, ohne dass ein kausaler Zusammenhang zwischen den Modellen aufgebaut wird.
Erfolgt bei einem Teilmodell m_l die Anwendung der dazugehörigen Auswahl-

funktion φ_{m_l}, so wird durch deren Bildbereich die Menge der Modellgleichungen, auf die die ungekoppelte Vereinigung abgebildet wird, wie folgt definiert:

$$\bar{m}\colon \bigcup_{j=1}^{k} P_j \to \bar{Z}_{m_1} \cup \ldots \cup \bar{Z}_{\varphi_{m_l}} \cup \ldots \cup \bar{Z}_{m_k}$$

Durch den Einfluss des Parametervektors p auf die Modellbildungskomponenten $m_1, \ldots, m_k$ wird bei der Anwendung der Simulation ($\gamma(\bar{m})$) auf die ungekoppelte Vereinigung eine gemeinsame Auswertung erreicht. Durch diese strukturelle Voraussetzung wird die Parameterabhängigkeit in das „Basismodell“ (in unserem Fall $\bar{m}$) übernommen.
Im Zuge des Transfers des Ergebnismodells $\bar{m}$ in eine „reale“ Simulationsumgebung können dadurch weitere Parameter generiert werden, die eine Parallelausführung der Teilmodelle erlauben. Wir betrachten ein Beispiel:

$$\begin{aligned} m_1(a_1, a_2, a_3) \quad &\text{mit} \quad \xi_1^1, \xi_2^1, \xi_3^1 \\ m_2(b_1, b_2, b_3) \quad &\text{mit} \quad \xi_1^2, \xi_2^2, \xi_3^2 \\ m_3(c_1, c_2) \quad &\text{mit} \quad \xi_1^3, \xi_2^3 \end{aligned}$$

Die ungekoppelte Vereinigung $\bar{m}(a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2)$ hat daher die Modellgleichungen $\xi_i^j \in \bar{Z}$ mit $i, j = 1, 2, 3$.

Die gekoppelte Vereinigung

Wir wollen nun eine Verbindung von Modellen betrachten, bei der die Verschaltung aufgrund von logisch ausgewerteten Bedingungen erfolgt. Das Ergebnis soll wiederum ein Modell im Sinne von *DBMR* sein.
Seien $m_1, m_2, \ldots, m_k$ Modelle mit den Modellparametermengen $P_1, P_2, \ldots, P_k$ in der Gestalt $\{a_i^1, \ldots, a_{j_i}^i\}$, allfällig vorhandenen Auswahlfunktionen φ_{m_i} und den Mengenabbildungsfunktionen σ_i, $i = 1, \ldots, k$.
Dann ist $\tilde{m}$ die *gekoppelte Vereinigung* dieser „Teilmodelle“ und hat folgende Gestalt:

$$\tilde{m}(\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_l) \to \sigma_i(m_1, m_2, \ldots, m_k, p)$$

Die Anwendung der Mengenabbildungsfunktionen zur Selektion der Gültigkeitsbereiche der jeweilgen Teilmodelle ist zulässig, da ein Modell als Zusammenfassung von Tupeln aus Zustandsmengen ebenso Element des Zustandsraums ist. Wir erkennen, dass jetzt der frei wählbare Parametervektor p fundamental in eine Defintion einer Basisstruktur von *DBMR* einfliesst. Wir „konstruieren“ ein Modell, dass eine l-elementige Parametermenge auf die oben angegebene Kopplung abbildet. Die Auswahlfunktionen werden ihre Funktion erfüllen, sobald das entsprechende Modell durch ein σ_i als Bildmenge angesprochen wird.
Wie leicht einzusehen ist, ist die gekoppelte Vereinigung – analog zu den entsprechenden Operationen der Relationenalgebra – assoziativ. Wir stellen daher den Formalismus der Parameterübergabe an den durch σ_i verursachten Übergängen dar.

$$\begin{aligned} m_1(A_1) &\rightarrow \bar{Z}|A_1 \\ m_2(A_2) &\rightarrow \bar{Z}|A_2 \\ \tilde{m}(P) &\text{ mit } A_1|P = A_2|P \\ \sigma(m_1, m_2, p) = \mathbf{1} &: P \rightarrow \bar{Z}|A_1 \\ \sigma(m_1, m_2, p) = \mathbf{2} &: P \rightarrow \bar{Z}|A_2 \end{aligned}$$

Auch hier wollen wir ein Beispiel, bei dem drei Modelle gegeben sind, betrachten:

$$m_1(a_1, a_2, a_3) \quad m_2(b_1, b_2) \quad m_3(c_1, c_2, c_3, c_4)$$

Wir betrachten nun die *gekoppelte Vereinigung*, $\tilde{m}$. Diese entsteht, indem im jeweiligen Gültigkeitsbereich für p m_1, m_2 oder m_3 angesprochen wird. Aus den Kopplungsvorschriften folgt, dass die durch eine allfällig schon durchgeführte Simulation (Funktion γ, Abschnitt 2.6.4) errechneten Werte für d_1 und d_2 an den Stellen π_k weitergegeben werden.[1]

$$\tilde{m}(d_1, d_2) \qquad \sigma_1(p < \pi_1)\colon m_1(a_1, a_2)$$

[1] In der Simulation wird, wenn eine Zielsprache generiert wird, ein `IF` oder besser ein `DISCRETE` Statement erforderlich sein, das dann, wenn die σ_i Bedingung eingetreten ist, die Werte der Variablen weitergibt.

$$\sigma_2(\pi_1 \leq p < \pi_2)\colon m_2(b_1, b_2)$$
$$\sigma_3(\pi_2 \leq p \leq \pi_3)\colon m_3(c_1, c_4)$$

2.6.4 Simulation

Wie wir schon in vorangegangenen Erläuterungen angegeben, werden wir bei der Definition im Sinne von *DMBR* auf die spezielle Auszeichnung eines Parameters verzichten. Dies erscheint dann auch für die Herleitung von *DBL/DBE* zielführend.

Die Zuordnung von Modell/Modellgleichung/Simulation

Durch ein geeignetes Verfahren (direkte Auswahl, gekoppelte/ungekoppelte Modellverknüpfung, Anwendung der Auswahlfunktion φ_m) wird einer Parametermenge $\hat{A}$ mit den Elementen $\{\hat{a}_1, \ldots, \hat{a}_k\}$ – wir nennen diese Parametermenge aufgrund des Auswahlmechanismus *ausgezeichnet* – aufgrund einer *frei wählbaren Parametermenge* $\hat{P}$ mit Elementen $\{\hat{p}_1, \ldots, \hat{p}_l\}$ eine Menge von Modellgleichungen zugewiesen. Diese sei

$$\hat{m} \subseteq \bar{Z}$$

Wegen der gewählten Eindeutigkeit aufgrund der ausgezeichneten Parametermenge $\hat{A}$ schreiben wir kurz:

$$\hat{m} = \bar{Z}|\hat{A}$$

Die Definition von Simulation

Um den Übergang zur eigentlichen Simulation, der Anwendung von numerischen Verfahren auf die Menge $\hat{m}$, zu verdeutlichen, ordnen wir jedem Element aus $\hat{A}$ einen Vektor aus $\mathbf{C}^{n_{\hat{a}_j}}$ zu. Ebenso verfahren wir mit $\hat{P}$ indem jedem Element ein Vektor aus $\mathbf{C}^{n_{\hat{p}_j}}$. Es sei nun $\sum n_{\hat{a}_j} = k$ und $\sum n_{\hat{p}_j} = l$. Die Bildmengen dieser Zuordnungsvorschrift bezeichnen wir mit $Z_{\hat{A}}$ und $Z_{\hat{P}}$. [2] Wir halten fol-

[2] $\zeta_{\hat{a}_j} \in Z_{\hat{A}}$ bleibt während Simulation konstant. Es werden die „Anfangswerte“gespeichert. Bei einer Simulation im Zeitbereich werden die Komponenten von $\zeta_{\hat{p}_j} \in Z_{\hat{P}}$ mitunter bis auf die der Zeit zugeordneten konstant bleiben.

gende Zusammenhänge fest:

$$\begin{aligned} \hat{a}_i \to \zeta^i_{\hat{a}_j} \qquad & \hat{p}_i \to \zeta^i_{\hat{p}_j} \\ \zeta_{\hat{a}_j} &= (\zeta^1_{\hat{a}_j}, \ldots, \zeta^k_{\hat{a}_j}) \\ \zeta_{\hat{p}_j} &= (\zeta^1_{\hat{p}_j}, \ldots, \zeta^l_{\hat{p}_j}) \end{aligned}$$

Simulation ist eine Abbildung γ mit:

$$\gamma \colon Z_{\hat{A}} \times Z_{\hat{P}} \to \mathbf{C}^k$$

Wir betrachten die Funktion γ als Relation. Die dadurch gebildeten Tupel können im Sinne einer Anwendung von Interpolationsverfahren als Stützstellen interpretiert werden und derart mit geeigneten Algorithmen verarbeitet werden.

2.7 Die Einbettung in ein *Entity-Relationship-Diagram*

Die gewonnenen Erkenntnisse über die in Beziehung stehenden Mengen und Abbildungen werden wir in ein Entity-Relationship Diagramm einbetten; dies verdeutlicht die Anwendung der Methoden der Relationenalgebra mit den neu definierten Auswahlfunktionen, um durch die Anwendung der Funktion γ zum gewünschten Ergebnis zu gelangen.
Die Abbildung 2.8 zeigt das Diagramm. Wir wissen, dass die Darstellung der Beziehungen Modell – Auswahlfunktion (φ_m) – Simulation (γ) – Experiment, die wir schon in der mengentheoretischen Beschreibung festgelegt haben, hier unter dem Aspekt der Grundsätze der Relationenalgebra neu beleuchtet werden. Wir betrachten die Elemente von *DBMR* und ihre entsprechenden Entitäten im *ERD*. Wir erkennen, dass durch den Aufbau einiger Attribute der einzelnen Entitäten die daraus resultierenden Relationen nicht in erster Normalform sind.
In den folgenden Beschreibungen führen wir bereits die Mengen an, aus denen die Tupel ausgewählt werden. Die Art der jeweiligen Attribute ist daraus unmittelbar zu folgern.

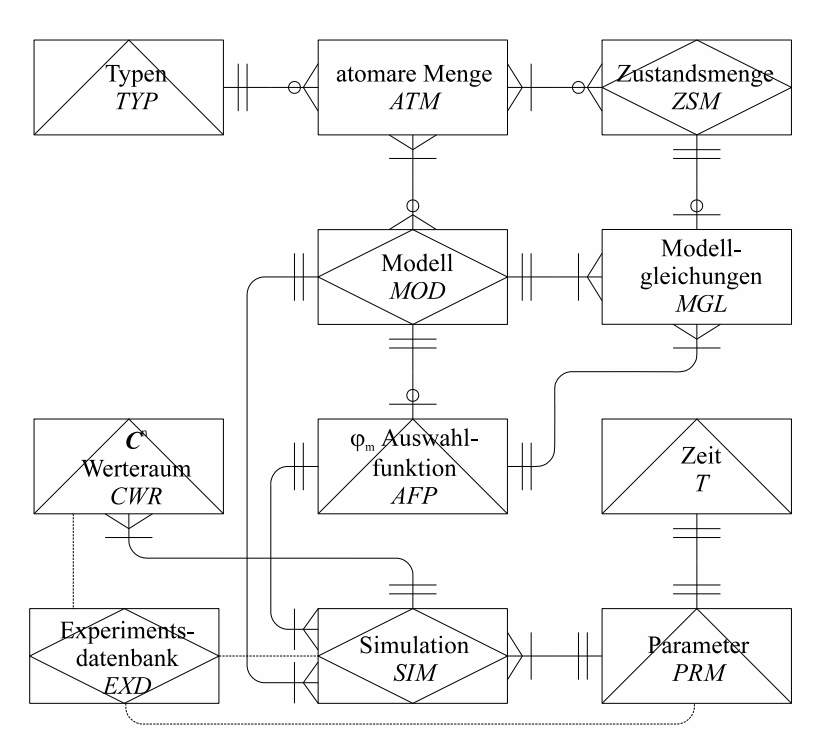


Abbildung 2.8: Das Entity Relationship Diagramm der *Databased Model Relation*

Die Entitäten

Entität „atomare Menge“ (ATM): Diese Entität beinhaltet sämtliche Elemente von Mengen, die in den theoretischen Überlegungen erläutert worden sind. Das Eindeutigkeitskriterium eines derartigen Elements ist die Bezeichnung und der Typ. Wir haben Mengen bzw. Elemente dieses Typs üblicherweise mit A, $\hat{A}$, $\tilde{A}$, P, $\hat{P}$, $\tilde{P}$ beziehungsweise mit a_k, $\hat{a}_k$, $\tilde{a}_k$, p_k, $\hat{p}_k$, $\tilde{p}_k$ bezeichnet.

$$\begin{aligned} ATM \quad \subseteq \quad & \text{dom (Mengenbezeichnung)} \times TYP \times \\ & \times \text{dom (Typattribut}_1\text{)} \times \ldots \times \text{dom (Typattribut}_{n_{TYP}}\text{)} \end{aligned}$$

Die Eindeutigkeit wollen wir durch das Attribut „Mengenbezeichnung“ gewährleisten.

Attributive Entität „Typen“ (TYP): Der Typ einer atomaren Menge (z.B. Parametermenge, frei wählbare Parametermenge) ist als derart wichtig

anzusehen, sodass eine eigene Entität berücksichtigt wird. Diese Entiät gibt die Art und andere Spezifika der auf sie weisenden atomaren Menge an.

$$\begin{aligned} TYP \quad \subseteq \quad & \text{dom (Typbezeichnung)} \times \text{dom (Attributspezifikation}_1) \times \\ & \times \ldots \times \text{dom (Attributspezifikation}_{n_{TYP}}) \end{aligned}$$

Assoziative Entität „Zustandsmenge“ (ZSM): Eine Zustandmenge ist ein Tupel aus Elementen atomarer Mengen. Wir setzen daher die Entität „atomare Menge“ zu sich selbst endlich oft in Beziehung und bilden dadurch (wie wir es auch im Abschnitt über das relationale Datenmodell getan haben) diese assoziative Entität. (Wir haben kurz geschrieben $Z = A_1 \times A_2 \cdots \times A_k$). Aus der speziellen Typenzuordnung (attributive Entität „Typen“) folgt sofort, dass mit dieser Entität auch die Zustandsräume ($\bar{Z} = Z_1 \times \cdots \times Z_l$) erfasst sind. Der Typ der Assoziativität folgt unmittelbar aus der eben beschriebenen Bildungvorschrift. Die Tupelmenge setzt sich daher wie folgt zusammen:

$$ZSM \subseteq ATM_1 \times ATM_2 \times \cdots \times ATM_k$$

Die folgenden drei Entitäten spiegeln den Mechanismus der Modellbildung wieder. Der Weg des Modells ist ja, wir wollen diese Zusammenhänge kurz in Erinnerung rufen, von der Art, dass wir Elementen atomare Mengen Elemente eines Zustandsraum zuordnen.

Assoziative Entität „Modell“(MOD): Wir haben ein Modell als Zuordnung von einer Menge P von Elementen atomarer Mengen zu Elementen des Zustandsraums $\bar{Z}$ beschrieben. Die Zuordnung schreiben wir daher als Tupel:

$$(p_1, p_2, \ldots, p_k, \kappa_1, \kappa_2, \ldots, \kappa_k) \quad \text{mit } p_i \in P \text{ und } \kappa_i \in A \times \bar{Z} \quad (= MGL)$$

Wir verwenden hier die Notation κ_i für die zugewiesenen Modellgleichungen. Das die Entität auszeichnende Eindeutigkeitskriterium ist die Angabe

der Menge P mit der Modellidentifikation m. Es folgt daher

$$MOD \subseteq \text{dom (Modellidentifikation)} \times ATM \times MGL$$

Entität „Modellgleichungen“ (MGL): Auch motivieren wir die Sichtweise dieser Abbildung von einer atomaren Menge auf den Zustandsraum als Teilmenge des kartesischen Produkts der Urbild- und Bildmenge:

$$MGL \subseteq ATM \times ZSM$$

Attributive Entität „Auswahlfunktion φ_m“(AFP): Die Auswahlfunktion φ_m bildet in Abhängigkeit ihres Zustandes ein Element der Parametermenge P eines Modells m auf verschiedene Modellgleichungen ab. Durch das Hinzufügen der Ausprägungen von φ_m spannen wir eine Produktmenge auf, die, berücksichtigend die Zugehörigkeit zum Modell m und den Typ der atomaren Menge der Parametermenge P, folgende Gestalt hat:

$$AFP \subseteq ATM \times \text{dom } (\varphi_m) \times MGL$$

Auch der Begriff der „Simulation“ ist durch unsere vorangestellten Überlegungen in ein Konzept überführt worden, das eine Einbettung in ein Beziehungsgeflecht mit den bereits eingeführten Entitäten darstellt. Dazu wollen wir die Folgenden beschreiben.

Attributive Entität „Parameter“ (PRM): In dieser Entität speichern wir die frei wählbaren Parameter. Wir schreiben daher mit der bisherigen Notation:

$$PRM \subseteq SIM \times \text{dom (Mengentyp)} \times (Z_{\hat{A}} \cup Z_{\hat{P}})$$

Attributive Entität „Zeit“ (T): Unsere bisherigen Überlegungen sind davon ausgegangen, keinen frei wählbaren Parameter speziell auszuzeichnen. Die Zeit ist daher nur eine Ausprägung der Entität PRM. Da wir die Zeit als speziell angeschriebene Größe in Hinblick auf die Realisierung MQL

bzw. *DABEQS* nicht missen wollen, führen wir diese attributive Entität ein. Es gilt daher folgende Beziehung

$$T \subseteq \text{Bezeichnung 'Zeit'} \times \mathbf{R}$$

Attributive Entität „Werteraum $\mathbf{C}^n$“ (CWR): Diese Entität definieren wir als Bildmenge der Abbildung γ. Die Tupelmenge, aus der diese Relation besteht, können wir, unter der allfälligen Hinzufügung eindeutiger Identifizierbarkeit, wie folgt anschreiben:

$$CWR \subseteq \text{dom (Werteraumidentifikation)} \times \mathbf{C}^n$$

Assoziative Entität „Simulation“ (SIM): Aus der Definition der Simulation als Funktion γ, die ausgehend von Parameterwerten eine Zuordnung auf Ergebnisvektoren beschreibt, leiten wir als Wertebereich der Entität folgende Beziehung ab:

$$SIM \subseteq MOD \times AFP \times PRM \times CWR$$

Entität „Experimentsdatenbank“ (EXD): Im Hinblick auf die Speicherung, Verwaltung und Auswertung von Experimenten, auch durch approximative Methoden, wollen wir eine Entität modellieren, die alle zu einem Experiment gehörigen Entitäten noch einmal zueinander in Verbindung setzt. Die Verbindung zu den Entitäten *MOD* und *AFP* ist durch die Verwendung der Entität *SIM* gegeben:

$$EXD \subseteq SIM \times CWR \times PRM$$

Relationen und Komplexitäten

Wie wir schon in Abbildung 2.8 gesehen haben, bestehen zwischen den die Relation aufbauenden Entitäten Beziehungen. Wir werden diese beschreiben und die darin enthaltenen Kardinalitäten motivieren. Die Beziehungen der Experimentsdatenbank werden wir in diesem Abschnitt nicht untersuchen – dies sei

Gegenstand des Abschnitts über die datenbankbasierende Modellinterpolation und Modellberechnung.

$TYP \longleftrightarrow ATM$**:** Jedem Tupel $atm \in ATM$ wird eindeutig ein Tupel $typ \in TYP$ zugeordnet. Diese Zuordnung können wir sofort aus der Definition der atomaren Menge folgern, ebenso die Kardinalität:

$$\begin{aligned} R_1 &= (ATM, TYP) \\ \text{comp}\,(R_1, ATM) &= (1,1) \\ \text{comp}\,(R_1, TYP) &= (0,\infty) \end{aligned}$$

$ATM \longleftrightarrow MOD$**:** Die Beziehung beschreibt, dass jedes Modell einer ausgewählten Parametermenge P gemäß der Abbildung κ Modellgleichungen zuordnet. Die Auswahl der Parametermenge ist als Relation daher wie folgt beschrieben:

$$\begin{aligned} R_2 &= (ATM, MOD) \\ \text{comp}\,(R_1, ATM) &= (0,\infty) \\ \text{comp}\,(R_1, MOD) &= (1,\infty) \end{aligned}$$

$ATM \longleftrightarrow ZSM$**:** Die Zustandsmenge haben wir als kartesisches Produkt von atomaren Mengen definiert. Dies hat folgende Beschreibung dieser Relation zur Folge:

$$\begin{aligned} R_3 &= (ATM, ZSM) \\ \text{comp}\,(R_3, ATM) &= (0,\infty) \\ \text{comp}\,(R_3, ZSM) &= (1,\infty) \end{aligned}$$

$ZSM \longleftrightarrow MGL$**:** Diese Beziehung beschreibt die Bildmenge der Funktion κ, die wir schon bei der Relation „$ATM \longleftrightarrow MOD$“, dort die Menge der Urbilder, untersucht haben.

$$R_4 = (ZSM, MGL)$$

$$\begin{aligned} \text{comp}\ (R_4, ZSM) &= (0, \infty) \\ \text{comp}\ (R_4, MGL) &= (1, \infty) \end{aligned}$$

MOD $\longleftrightarrow$ *AFP*: Wenn wir beim Bilden der Gesamtheit der Modellgleichungen eine Auswahl treffen wollen, benützen wir hiezu die Funktion φ_m. Zur jeweiligen Auswahl werden die Indikatorwerte in der Entität *AFP* abgelegt. Da, wie gefordert, die Zuordnung zum Modell in der Entität *MOD* eindeutig ist, die Verwendung einer Auswahlfunktion nicht zwingend abgeleitet werden kann, folgt daraus:

$$\begin{aligned} R_5 &= (MOD, AFP) \\ \text{comp}\ (R_5, MOD) &= (0, 1) \\ \text{comp}\ (R_5, AFP) &= (1, 1) \end{aligned}$$

MGL $\longleftrightarrow$ *AFP*: Analog wie für R_5 und R_2 folgern wir, da bei Anwendung einer Auswahlfunktion φ_m die Zuordnung der Modellgleichungen über diese Funktion erfolgt, dass Eindeutigkeit in Bezug auf das betrachtete Tupel aus *AFP* gegeben ist. Ein Tupel aus *MGL* kann zu mehreren Tupeln aus *AFP* in Beziehung stehen.

$$\begin{aligned} R_6 &= (MGL, AFP) \\ \text{comp}\ (R_6, MGL) &= (1, \infty) \\ \text{comp}\ (R_6, AFP) &= (1, 1) \end{aligned}$$

MGL $\longleftrightarrow$ *MOD*: Diese Relation können wir analog R_6 beurteilen. Diese Relation ist dann von besonderer Bedeutung, wenn die Auswahl der Modellgleichungen ohne Definition einer Auswahlfunktion erfolgt.

$$\begin{aligned} R_7 &= (MGL, MOD) \\ \text{comp}\ (R_7, MGL) &= (1, \infty) \\ \text{comp}\ (R_7, MOD) &= (1, 1) \end{aligned}$$

MOD $\longleftrightarrow$ *SIM*: Wir haben den Begriff *Simulation* als Funktion γ dargestellt, die einem Modell und bestimmten Parameterwerten numerische Ergebnisse zuordnet. Diese Zuordnung übernimmt die assoziative Entität Simulation. Aufgrund dieser Eigenschaft erhalten wir folgende Beschreibung dieser Relation.

$$\begin{aligned} R_8 &= (MOD, SIM) \\ \text{comp}\ (R_8, MOD) &= (1, \infty) \\ \text{comp}\ (R_8, SIM) &= (1, 1) \end{aligned}$$

AFP $\longleftrightarrow$ *SIM*: Wiederum müssen wir, wenn mittels φ_m die Auswahl der Modellgleichungen erfolgen soll, von der Entität AFP ausgehen. Daher ergibt sich, analog R_8, die Komplexität dieser Relation:

$$\begin{aligned} R_9 &= (AFP, SIM) \\ \text{comp}\ (R_9, AFP) &= (1, \infty) \\ \text{comp}\ (R_9, SIM) &= (1, 1) \end{aligned}$$

CWR $\longleftrightarrow$ *SIM*: Die Entität CWR stellt das numerische Ergebnis einer Simulation dar. Wir sehen leicht ein, dass ein gewisser Wertevektor durchaus als Ergebnis von mehreren Simulationsläufen vorkommen kann. Allerdings haben wir durch die Definition der Funktion γ Eindeutigkeit im anderen Sinne gefordert:

$$\begin{aligned} R_{10} &= (CWR, SIM) \\ \text{comp}\ (R_{10}, CWR) &= (1, \infty) \\ \text{comp}\ (R_{10}, SIM) &= (1, 1) \end{aligned}$$

PRM $\longleftrightarrow$ *SIM*: Bei der Betrachtung dieser Relation erhalten wir, unter der Verwendung zu R_{11} analoger Sachverhalte, folgendes:

$$R_{11} = (PRM, SIM)$$

$$\begin{aligned} \text{comp}\ (R_{11}, PRM) &= (1, \infty) \\ \text{comp}\ (R_{11}, SIM) &= (1, 1) \end{aligned}$$

$T \longleftrightarrow PRM$**:** Durch die Einführung der Zeit als Entität T, muss diese zur Entität der Parameterwerte PRM in Beziehung gesetzt werden. Die Beschreibung folgt sofort aus dem attributiven Charakter:

$$\begin{aligned} R_{12} &= (T, PRM) \\ \text{comp}\ (R_{12}, T) &= (1, 1) \\ \text{comp}\ (R_{12}, PRM) &= (0, 1) \end{aligned}$$

2.7.1 Normalisierung dieser Beziehungen

Ähnlich dem experimentellen System XSQL in [50], das auf nicht normalisierten Relationen (NF^2-Relationen - kurz für *N*on *F*irst *N*ormal *F*orm) basiert, müssen wir im Konzept *DBMR* die soeben eingeführten Entitäten entflechten, sodass diese in der ersten Normalform zur Verfügung stehen. Diese Relationen können dann in ein herkömmliches Datenbankmanagementsystem eingebettet werden. [3]

Wir wählen einige Entitäten aus, um unsere Vorgangsweise daran zu illustrieren. Das gesamte Ergebnis dieses Vorgehens ist in der Datenbankbeschreibung des Systems *DABEQS* zu finden.

Betrachten wir nun die „Relationenkette“ $TYP \longleftrightarrow ATM \longleftrightarrow MOD \longleftrightarrow MGL$. Schon die Definition der Bildmengen der Entitäten zeigt, dass die daraus abgeleiteten nicht in erster Normalform sind. Die Verfahren, die wir wählen, um das gewünschte Ergebnis zu erzielen, folgen direkt aus den Algorithmen zur Überführung von Relationen in gewünschte Normalformen. Bei der Anführung der Attribute der Relationen verwenden wir für deren Schlüssel die Notation $[< rel\_name > \_key]$.

[3] Hier begegnet uns die Hauptmotivation für die vorangegangenen Untersuchungen. Durch die weite Verbreitung und Anwendung von relationalen Datenbanksystemen stellen diese wohl die ausgereifteste und sicherste Basis für ein Simulationswerkzeug dar.

Die Attribute eines Tupels $typ \in TYP$ sind

$$([typ\_key], typ\_bez, attr_1, \ldots, attr_{n_{typ}})$$

Die Menge der Attribute ist variabel, wir müssen daher diese Relation, basierend auf den funktionalen Abhängigkeiten, in weitere Relationen aufteilen. Wir schreiben daher:

$$\begin{aligned} typ &= ([typ\_key], typ\_bez, n_{[typ\_key]}) \\ typ\_attr &= ([typ\_key], typ\_attr\_num, typ\_attr\_spec) \end{aligned}$$

Analog gehen wir für Tupel $atm \in ATM$ vor. Deren Attribute sind

$$([atm\_key], atm\_bez, [typ\_key], t\_attr_1, \ldots, t\_attr_{n[typ\_key]})$$

Die aufgrund der Forderung nach 1NF-Relationen und konsolidierter funktionaler Abhängigkeit erhaltenen Relationen sind:

$$\begin{aligned} atm &= ([atm\_key], atm\_bez, [typ\_key]) \\ atm\_typ\_attr &= ([atm\_key], typ\_attr\_num, atm\_typ\_value) \end{aligned}$$

Das Modell und seine Gleichungen sind in dieser „Relationenkette“ leicht in die geforderte Normalform zu bringen. Da diese Entitäten Beziehungen beschreiben, erreichen wir dies durch die Einführung von Fremdschlüsselattributen. Wir schreiben daher:

$$\begin{aligned} mod \in MOD &: \text{attr } (mod) = (mod\_bez, [atm\_key], [mgl\_key]) \\ mgl \in MGL &: \text{attr } (mgl) = ([mgl\_key], [atm\_key], [zsm\_key]) \end{aligned}$$

Für die umfassende Überführung in die dritte Normalform in einer vereinfachten *ERD*-Darstellung verweisen wir auf Abbildung 4.1.

2.7.2 Einige Abbildungen innerhalb dieses Relationenschemas

Die für bestimmte Durchführungen erforderlichen Datenbankoperationen seinen nun beschrieben. Diese folgen aus der Struktur (vergleiche Abbildung 2.8) und den daraus aufgebauten Relationen:

Berechnung des Modells m: Wir wissen, dass die Berechnung eines Modells gleich der Anwendung der Funktion γ auf eine Teilmenge des Zustandsraums $\bar{Z}$ ist. Wir können daher schreiben:

$$\gamma(\pi_{ZSM}(\sigma_{[mod\_key]=m}(MOD) \bowtie MGL \bowtie ZSM))$$

Berechnung eines Modells m mit Anwendung der Auswahlfunktion φ_m: In diesem Fall müssen wir noch eine Verbindung mit der Entität AFP herstellen, um so den Gültigkeitsbereich und den Auswahlmechanismus von φ_m miteinzubeziehen:

$$\gamma(\pi_{ZSM}(\sigma_{[mod\_key]=m}(MOD) \bowtie \sigma_{[afp\_key]=\varphi_m}(AFP) \bowtie MGL \bowtie ZSM))$$

Die innere Modellverknüpfung von m_1 und m_2: Die Verknüpfungsvorschrift besagt allgemein, dass die Elemente atm_{1_k} durch die Elemente atm_{2_k} für $k = 1, \ldots, n$ zu ersetzen sind.

$$\gamma(\pi_{ZSM}(\pi_{ZSM-\{ATM_{1_1},\ldots,ATM_{1_n}\}}(\sigma_{[mod\_key]=m_1}(MOD) \bowtie MGL \bowtie ZSM) * \\ *(\sigma_{[mod\_key]=m_2}(MOD) \bowtie \sigma_{atm_{2_k}}(ATM) \bowtie MGL \bowtie ZSM)))$$

Wird bei der Modellbildung φ_{m_i} benutzt, dann ist der Term der jeweiligen Modellauswertung durch die Verbindung mit AFP und Selektion der gewünschten Auswahlfunktion analog dem oberen Muster zu ergänzen.

die gekoppelte äußere Verknüpfung von $m_1, \ldots, m_l$: Um eine Unterscheidbarkeit der Mengenauswahlfunktion zur Selektionsfunktion sicherzustellen, bezeichnen wir diese im folgenden mit $\bar{\sigma}$. p bezeichnet den Vektor frei wählbarer Parameter, der als Eingangsgröße für die $\bar{\sigma}_i$ Funktionen benötigt wird.

Es sollen die Elemente der jeweiligen Modellparametermenge $atm_{1_k}, \ldots, atm_{l_k}$ für $k = 1, \ldots, n$ „verschalten“ werden.

$$\gamma(\pi_{ZSM}((\bar{\sigma}_1 \circ \ldots \circ \bar{\sigma}_l)(\sigma_{[mod\_key]=m_1}(MOD) \bowtie MGL \bowtie ZSM, \ldots, \sigma_{[mod\_key]=m_l}(MOD) \bowtie MGL \bowtie ZSM, p)))$$

die ungekoppelte äußere Verknüpfung von $m_1, \ldots, m_l$: Wir haben diesen Begriff als gleichzeitige Berechnung von mehreren Modellen eingeführt. Daher schreiben wir:

$$\gamma(\pi_{ZSM}(\sigma_{[mod\_key]=m_1}(MOD) \bowtie MGL \bowtie ZSM), \ldots, \pi_{ZSM}(\sigma_{[mod\_key]=m_l}(MOD) \bowtie MGL \bowtie ZSM))$$

2.8 Der resultierende Modellbegriff

Unsere Überlegungen, bei denen der Modellbegriff als Ausprägung von Tupeln eines Relationenschemas eingeführt wurde, verlangen nun eine differenzierte Betrachtung im Vergleich zur bisherigen Sichtweise (vergleiche [52]). Darin wird von einer strikten Trennung von *Modell* und *Experiment* ausgegangen. Diese Trennung ist unter Berücksichtigung des *experimental frame* eines Modells sinnvoll, da dieser die Menge aller gültigen Experimente für eine vorgegebene Modellbeschreibung darstellt.
Da wir zeigen möchten, dass *DBMR* mit seinen Mechanismen Teile der „klassischen“ Sichtweise eines *experimental frames* bereits in der Beschreibung des Modells aufnimmt, wollen wir diesen und die die Grundlage dafür bildenden Begriffe kurz formal darstellen.
Ein *Base Model* B charakterisiert die Menge von errechenbaren Daten eines realen Systems R. Dieses System ist als folgendes Tupel anzuschreiben:

$$B = \langle T_B, X_B, (X_B, T_B), Q_B, \delta_B \rangle$$

Die Menge **E** beschreibt alle *Experimental Frames* E, die einen Zugriff auf das Modell B eines realen Systems R beschränken. Diese sind analog folgender

Struktur aufgebaut:

$$E = \langle \Omega_E, Y_E, \lambda_E, V_E \rangle$$

wobei Ω_E die Menge der in E möglichen Eingabesegmente, Y_E die Menge aller möglichen Ausgabewerte, λ_E die Beobachtungsfunktion und V_E die Menge der Gültigkeitsbereiche ist.
Erst durch die Zusammenführung eines Basismodells B mit dem dazugehörigen Experimentsrahmen ist eine „Auswertbarkeit“ von B gegeben. So ergibt sich die Beobachtung des realen Systems unter Anwendung des *Experimental Frame* auf das *Base Model* von R. Wir beachten, dass das Modell an sich keine Ein/Ausgangsgrößen kennt – diese werden erst durch den *Experimental Frame* definiert.
Speziell für den Fall diskreter ereignisorientierter Modellbildung finden wir den Formalismus *DEVS*. Dieser besteht aus

- einer Menge X von Eingabeereignissen
- einer Menge Y von Ausgaben
- einer Menge S von internen Zuständen
- einer externen Überführungsfunktion δ_{ext}, die bestimmt, wie das System auf Eingabeereignisse reagieren soll
- einer internen Überführungsfunktion δ_{int}, die bestimmt, wie das System auf zeitlich vorgesehene Ereignisse reagieren soll
- einer Zeitfortschreitungsfunktion ta, die die Zeit bis zum nächsten internen Ereignis angibt
- einer Ausgabefunktion λ, die die Ausgabe des Systems bestimmt.

Atomare Modelle, die mittels des DEVS Formalismus aufgebaut wurden, können zu komplexeren Modellen zusammengebaut werden – dies ermöglicht der Formalismus der gekoppelten DEVS Systeme, die aus folgenden Teilen bestehen:

- einer Menge X von Eingabeereignissen
- einer Menge Y von Ausgaben

- einer Menge C von Komponentensystemen
- der externen Eingabekopplung EIC, die die Kopplung der Eingabeports des Netzwerks zu Eingabeports von Komponenten angibt
- der externen Ausgabekopplung EOC, die die Kopplung von Ausgabeports zu Ausgabeports des Netzwerkes angibt
- der internen Kopplung IC, die die Kopplung der Komponenten untereinander bestimmt
- einer Funktion $SELECT$, die bei Gleichzeitigkeit interner Ereignisse mehrerer Komponenten die Reihenfolge der Ausführung festlegt.

Basierend auf diesem Konzept werden die drei grundsätzlichen Systemtypen – die Differentialsysteme (DESS), die diskreten Zeitsysteme (DTSS) und die diskreten Ereignissysteme (DEVS) – in der Literatur ([39], [40]) zusammengeführt. Eine etwas andere Sichtweise des Modellbegriffs finden wir in [5], [6]. Hier wird die übliche Trennung von *model frame* und *experimental frame* um den *method frame* erweitert. Die dadurch beschriebenen Methoden sind analytische und numerische Algorithmen, die ein Modell entsprechend dem durch sie durchführbaren Verfahren auswerten und analysieren. Ein Experiment ist dann die Anwendung einer Methode auf ein Modell. Formal können wir daher schreiben, wenn ein Modell mit MO_i, eine Methode mit ME_j bezeichnet wird, dass für ein Experiment $E_{ij} = ME_j(MO_i)$ gilt.
DBMR nutzt bei der Beschreibung der Modelle die Einbettung der Zuordnung *Modelldefinition* $\longleftrightarrow$ *Modellauswertung (=Experiment)* $(ATM \longleftrightarrow MGL(\longleftrightarrow AFP) \longleftrightarrow MOD \longleftrightarrow SIM)$ in Tupel, deren Verknüpfung und damit Kopplung den Methoden der Relationenalgebra genügt. Dadurch erreichen wir, dass Definition dieser Verknüpfungsvorschriften in die Modellbeschreibung eingeht. Erst durch die Anwendung der Simulationsfunktion γ auf dieses Ausschnittstupel des Zustandsraums $\bar{Z}$ wird eine Überführung in Daten erreicht.
Wird nun einem Modell m mit Parametermenge A eine Menge von Modellgleichungen zugeordnet, dann ist diese Zuordnung als Erweiterung der Modelldefinition (des *Base Model*) zu sehen. Die Auswertung des Modells (die Anwendung

eines *Experimental Frame* auf ein *Base Model*), d.h. Bilden von $\gamma(m)$, ist daher ohne Berechnung einer allfälligen parameterabhängigen Zuordnungsvorschrift durchzuführen.

Bei der Untersuchung und dem Vergleich der verschiedenen Kopplungsmechanismen (wir verwenden die für *DBMR* eingeführten Begriffe der inneren und äußeren Verknüpfung) sehen wir, dass durch deren Anwendung die Rechenregeln der Relationenalgebra als Kopplungsoperation herangezogen werden, daher die Beschreibung einer Kopplungs- oder Auswahlfunktion nicht erforderlich ist. Wir haben auch definiert, dass ein Modell durch die Auswahlfunktion φ_m charakterisiert werden kann. Da keine Einschränkung von deren Wertevorrat postuliert wurde, können verschiedenste Abhängigkeiten – nicht nur, wenn auch die häufigste Anwendung, die der ein- und ausgehenden Größen – konstruiert werden.

Kapitel 3

Datenbank basierende Interpolation/Modell Berechnung *Databased Interpolation/Databased Calculation „DBI/DBC“*

Im Sinne von *DBMR* behandelt dieser Abschnitt die verschiedenen Möglichkeiten ein numerisches Ergebnis der Simulation zu erhalten, d.h. $\gamma(\hat{m})$ zu berechnen. Wir wollen zuerst einen kurzen Überblick über die numerischen Lösungsmöglichkeiten des zugrunde liegenden Anfangswertproblems geben, dann nach der Betrachtung von Interpolationsmethoden diese im Erweiterungskonzept zu *DBMR DBI/DBC* einbetten.

3.1 Numerische Integrationsverfahren

Wir geben ein Anfangswertproblem in folgender Form an, wobei wir voraussetzen, dass f in einem Gebiet der x, y-Ebene lipschitzstetig ist.

$$\begin{cases} y' = f(x,y) = f(x, y(x)) \text{ mit Anfangsbedingung} \\ y(x_0) = y_0, \qquad (x_0, y_0) \end{cases}$$

Im Integrationsintervall I des Anfangswertproblems erklären wir äquidistante Stützstellen x_i mit $i = 0, \ldots, n$ und Schrittweite h. Durch numerische Verfahren wollen wir an den Stützstellen x_i für die Werte $y(x_i)$ der gesuchten Lösung $y(x)$ Näherungswerte $Y(x_i)$ bestimmen. Integrieren wir formal, dann erhalten wir für $i = 0, \ldots, n-1$:

$$y(x_{i+1}) = y(x_i) + \int_{x_i}^{x_{i+1}} f(x, y(x))dx$$

Im wesentlichen unterscheiden wir die numerischen Verfahren durch die näherungsweise Berechnung des oben angegebenen Integrals. Wir teilen sie daher ein in:

- Einschrittverfahren (*one-step methods*)
- Mehrschrittverfahren (*multi-step methods*)
- Extrapolationsverfahren (*extrapolation algorithms*)

Die Einschrittverfahren verwenden zur Berechnung eines weiteren Näherungswertes Y_{i+1} nur einen vorangehenden Wert Y_i.
Die Mehrschrittverfahren verwenden $s+1, s \leq 1$ vorangehenden Werte $Y_{i-s}, \ldots, Y_i$ zur Berechnung von Y_{i+1}.
Das Extrapolationsverfahren stellt einen zum Verfahren von Romberg (für numerische Quadratur) analogen Algorithmus dar.
Unter den Ein- und Mehrschrittverfahren bilden die Praediktor-Korrektor-Verfahren eine spezielle Klasse. Es sind Verfahren, die einen Näherungswert $Y_{i+1}^{(0)}$ zunächst nach einem Einschritt- oder Mehrschrittverfahren bestimmen. Wir bezeichnen diese Bildungsvorschrift als Praediktor. Dieser Wert wird dann mit

dem Korrektor verbessert (mitunter iterativ). Diese Verbesserungen heissen dann $Y_{i+1}^{(1)}, Y_{i+1}^{(0)}, \ldots$.

3.2 Interpolation

Wir haben $n+1$ Wertepaare $(x_i, y_i) \in \mathbf{R}$ gegeben, wobei die Stützstellen x_i paarweise verschieden sein sollen, und suchen eine hinreichend „glatte“ Funktion, die durch diese Wertepaare bestimmt wird.
Wir kennen einige Verfahren zur Interpolation:

- Die eindeutige Bestimmung einer noch dazu abzählbar oft differenzierbaren Funktion gelingt mittels der Interpolation durch ein algebraisches Polynom. Ein Problem stellt das starke Schwingverhalten dieser Funktion an den Intervallgrenzen dar.

- Wenden wir Splinefunktionen an, d.h. zwischen den Stützstellen erfolgt die Interpolation durch ein kubisches Polynom, das bestimmten Forderungen an seine Ableitungen genügen muss, erhalten wir eine „glattere“ Kurve, deren Schwingverhalten an den Rändern im Vergleich zu der Polynominterpolation ein günstigeres ist.

- Durch die Anwendung der rationalen Interpolation erhalten wir zu dem Vorteil der Glattheit dieser Näherungsfunktion die Darstellung in einer einzigen mathematischen Gleichung.

Da die ursprüngliche Funktion oft nicht bekannt ist, ist eine Aussage über den durch die Anwendung der Interpolation gemachten Fehler schwierig.

3.3 DBI/DBC und DBMR

Innerhalb von *DBMR* stellt Simulation [1] die Auswertung der Funktion γ dar, die den als Parametermenge ausgezeichneten Elementen geeigneter atomarer

[1] Wir betrachten in diesem Kapitel in erster Linie die *Einbettung* einer interpolierenden Modellberechnung – hier der Einfachheit halber der linearen – in *DBMR*. Für genauere Untersuchungen bezüglich der Auswirkungen durch die Auswahl der jeweiligen Interpolationsalgorithmen verweisen wir auf weitere laufende Arbeiten innerhalb der ARGESIM.

Mengen numerische Werte zuordnet. Es ist leicht einzusehen, dass jede Auswertung der Funktion und Bilden der Ergebnisse einen „Werteraum“ [2] bildet. Betrachten wir die darin enthaltene numerische Information, so erhalten wir eine (allgemeine $\mathbf{C}^{n\times m}$) Tabelle. Diese Tabelle besitzt als „Spalten“ die betrachteten Ausprägungen der Menge $Z_{\hat{A}}$, als Zeilen die durch die Funktion γ basierend auf den Parametern $Z_{\hat{P}}$ zugewiesenen Werte $\in \mathbf{C}^n$.
Wir möchten untersuchen: Seien in dieser Tabelle k Ausprägungen [3] $\in Z_{\hat{P}}$ zum Modell m aufgenommen und sei nun der Werte des Modells (wir schreiben dafür $\hat{m}$) an der Stelle $\hat{p}$ gefragt. Im folgenden schreiben wir für die einzelnen Elemente aus $Z_{\hat{A}}$ m_j, für die aus $Z_{\hat{P}}$ p_i, für den durch die Funktion γ zugewiesenen numerischen „Wertevektor“ $\gamma_{m_i p_j}$
Die lineare Interpolation liefert:

$$\begin{aligned} \hat{p} &= p_i + \alpha(p_j - p_i) \\ \Rightarrow \hat{m} &= m_i + \alpha(m_j - m_i) \\ \Rightarrow \gamma_{\hat{m}\hat{p}} &= \gamma_{m_i p_i} + \alpha(\gamma_{m_j p_j} - \gamma_{m_i p_i}) \end{aligned}$$

Es seien $\hat{p}$, p_i, p_j paarweise verschieden. Wir wählen nun $r = i, j$ so, dass

$$\|p_r - \hat{p}\| < \min_3 \|p_l - \hat{p}\|$$

Wir bezeichnen mit $\min_3$ den „drittkleinsten Wert“ des folgenden Ausdrucks, $p_{i,j}$ sind daher die Elemente aus $Z_{\hat{P}}$, die $\hat{p}$ am nächsten liegen. [4]
Diese Überlegungen wollen wir nun zu einer Verallgemeinerung für *DBI/DBC* mit $\hat{\gamma}_{m_i p_i} \in \mathbf{C}^n$ ausformulieren. Es sei nun $\hat{p} \neq p_i$ für alle Elemente aus $Z_{\hat{P}}$. Gesucht ist eine Interpolation für $\gamma_{m_i \hat{p}}$, die wir mit $\hat{\gamma}_{m_i \hat{p}}$ bezeichnen.
Als ersten Fall betrachten wir, dass *hatp* von $p_{i,j}$ linear abhängig ist. Dann gelten die oben festgehaltenen Überlegungen für $\hat{\gamma}_{m_l \hat{p}}$. Ist dies nicht der Fall,

[2] Hier sehen wir die Analogie zur assoziativen Entität *CWR*.
[3] Es leuchtet ein, dass $k \geq 2$ gelten muss.
[4] Um bei der Implementierung iterierte Berechnungen zu vermeiden, wählen wir das Quadrat der Norm, um eine Bildung der Wurzel zu vermeiden.

müssen wir untersuchen, ob Elemente p_i existieren, sodass gilt

$$\hat{p} = \sum \alpha_i p_i \text{ mit } \sum \alpha_i = 1$$

Es sei dann für $\gamma_{m_l\hat{p}}$ $\sum \alpha_i \gamma_{m_l p_i}$ eine geeignete Approximation. Die Güte dieser Näherung wollen wir durch folgende Fehlerabschätzung qualifizieren. Wir setzen voraus, dass γ dehnungsbeschränkt mit Konstante D sei. Sodann gilt

$$\begin{aligned}
\|\gamma_{m_l p_j} - \gamma_{m_l p_i}\| &\leq D\|p_j - p_i\| \\
\|\hat{\gamma}_{m_l\hat{p}} - \gamma_{m_l\hat{p}}\| = \|\hat{\gamma}_{m_l\hat{p}} - \gamma_{m_l p_i} + \gamma_{m_l p_i} - \gamma_{m_l\hat{p}}\| &\leq \\
\leq |\alpha_1|\|\gamma_{m_l p_1} - \gamma_{m_l p_n}\| + \ldots + |\alpha_{n-1}|\|\gamma_{m_l p_{n-1}} - \gamma_{m_l p_n}\| &\leq \\
\beta_1\|p_1 - p_n\| + \ldots + \beta_{n-1}\|p_{n-1} - p_n\| + \beta_n\|p_n - \hat{p}\| &
\end{aligned}$$

β_i sind zu D proportionale Konstante. Liegen daher die Stützstellen p_i nahe genug beieinander, dann ist der Gesamtfehler hinreichend klein.

3.4 Die Einbettung in DBMR

Schon die Herleitung der Darstellung von *DBMR* in einem Entity Relationship Diagramm beinhaltet durch die Bereitstellung der Entität *CWR* die erforderlichen Daten. Abbildung 3.1 zeigt den Ablauf des jeweiligen Algorithmus. Es ist leicht festzustellen, dass die Schritte des Algorithmus unterschiedlich von der Art der Modellauswertung – interpoliert oder direkt berechnet – ausfallen. Die in diesem Ablauf beschriebenen Punkte erfolgen dann folgendermaßen - wir nehmen die Entität *Experimentsdatenbank (EXD)* als „generische“ Entität an:

Bereitstellen der Modellgleichungen: Die Modellgleichungen werden analog dem in Kapitel 2 dargestellten Zugriff auf die Entitäten *MOD*, allenfalls *AFP*, *MGL* und *ZSM* ausgewählt. Weitere zulässige Verfahren für dieses Bereitstellen sind das Bilden innerer und äußerer Modellverknüpfungen. Gemäß unserer vereinbarten Notation ist diese Menge $\bar{Z}|\hat{A}$.

Feststellen der Existenz von *DBI/DBC* Wir führen einen Zugriff auf die

Experimentsdatenbank *EXD* durch:

$$\pi_{MOD}(\sigma_{[mod\_key]=m}(EXD))$$

Ist das Modell durch die in *DBMR* vorgesehenen Kopplungsmechanismen zustande gekommen, müssen wir das im System abgelegte Modell feststellen. Wir erreichen das durch einen Zugriff auf den Natural Join von Modell, Auswahlfunktion φ_m, atomare Menge, Modellgleichung und Zustandsmenge mit den Gleichungen des Ergebnismodells.

$$\pi_{MOD}(EXD \bowtie \pi_{MOD}(\sigma_{[atm\_key]=P \wedge [zsm\_key]=\bar{Z}_m}(MOD \bowtie \\ \bowtie AFP \bowtie ATM \bowtie MGL \bowtie ZSM)))$$

Erstellen des Vektors $\hat{p}$: Als geforderter Parameter wird $\hat{p}$ rein mathematisch ausserhalb des Relationenschemas erstellt.

Interpolation: Wir greifen nun auf die *Experimentsdatenbank* zu, um die Stützstellen der Interpolation zu finden.

$$\pi_{WRC}(\sigma_{[mod\_key=m]}(EXD) \bowtie \sigma_{\|prm-\hat{p}\|<\min}(PRM))$$

Mit der Ergebnismenge dieser Anfrage berechnen wir mit einem geeigneten Interpolationsverfahren den zuzuordnenden Wert.

Berechnung von $\gamma(\bar{Z}|\hat{A})(\hat{p})$: Ist beim Zugriff auf *EXD* keine geeignete Interpolationsbasis gefunden worden, müssen wir γ durch direkte Berechnung feststellen.

Sind $a_i \in \bar{Z}|\hat{A}$ basierend auf *DBI/DBC* $\Rightarrow$ Berechnung: Analog obigen Zugriffs greifen wir wiederum auf die Experimentsdatenbank zu und erwarten ein Ergebnis-Set. Kann dieser Zugriff kein Ergebnis bringen, erfolgt auch hier eine direkte Berechnung.

Speichern: Wir haben nun ein Tupel *exd* erstellt, das in die Relation eingefügt wird.

Visualisieren: Die errechneten bzw. interpolierten Werte werden mit einem geeigneten Werkzeug graphisch dargestellt.

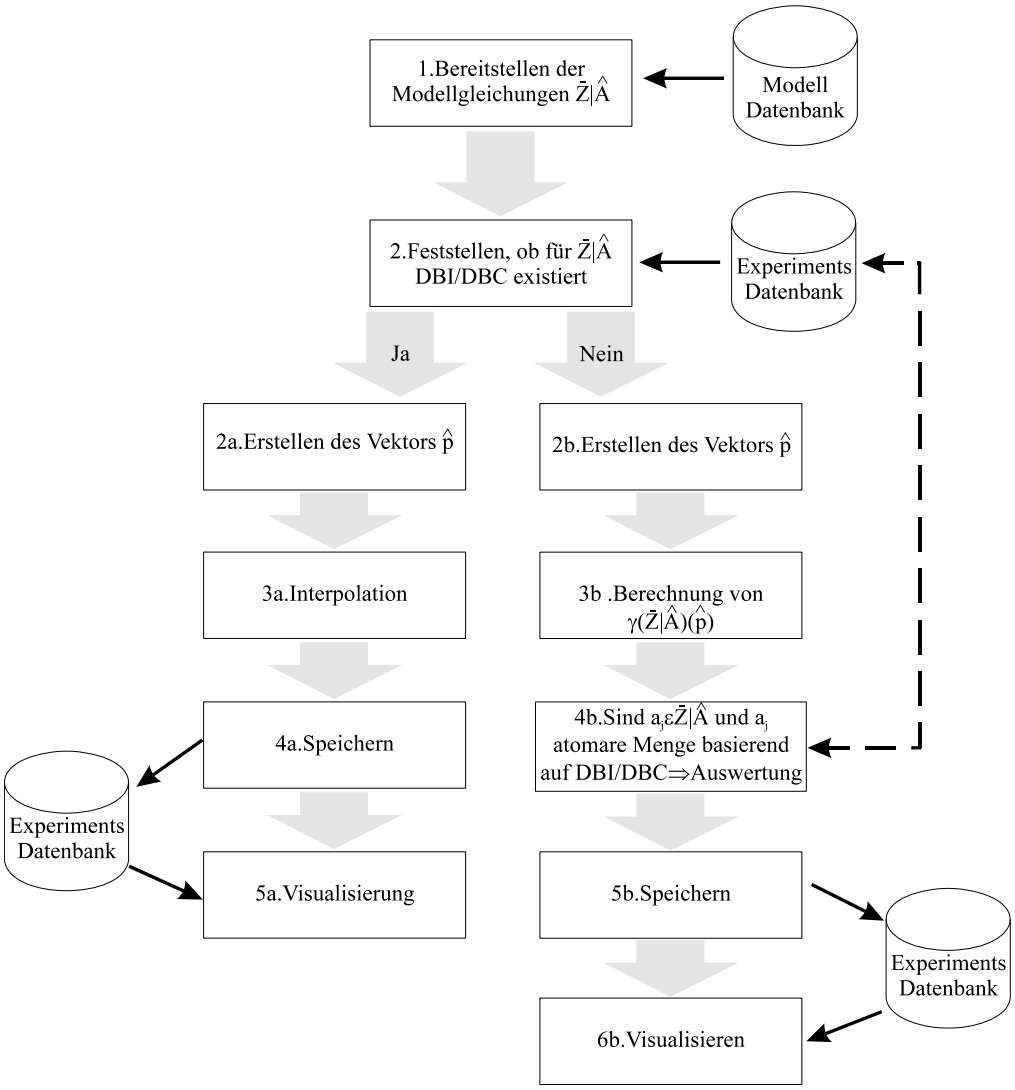


Abbildung 3.1: Eine schematische Darstellung der Ablaufs einer Modellauswertung

Kapitel 4

MQL (Model Query Language) - Definition einer Sprache für die Realisierung von Modellverknüpfungen

4.1 Die Datenbank

In den Grundlagen haben wir die Struktur der Relationenalgebra von *DBMR* kennengelernt. Ebenso war uns bewusst, dass für eine Realisierung des Konzepts eine Umformung in die dritte Normalform erforderlich ist. Für einen Teil des Datenbankschemas haben wir diese ja in Abschnitt 2.7.1 schon gezeigt. Nun verweisen wir auf Abbildung 4.1, aus der die normalisierte Struktur hervorgeht. Die Methode der Herleitung wollen wir anhand der aufgelösten Entitäten noch einmal anführen.
Wir definieren zwei Abbildungen, die uns bei dieser Herleitung hilfreich sein werden:

1. Die Funktion ind(α) gibt die Position von α in einem Tupel an, falls α dem Typ der atomaren Menge angehört. Ist α eine Funktion (wie die Zuordnungsfunktion der Modellgleichungen oder die Abbildungsfunktion der Simulation), dann wird die Anzahl der möglichen Zuordnungswerte

angegeben.

2. Mit der Funktion $\mathrm{val}(\alpha, f_\alpha)$ erstellen wir einen Indikator der besagt, ob bei dem Zustand von α die Anwendung der Funktion f_α zulässig ist. Diese Hilfsfunktion ist für die Abbildung der Auswahlfunktion φ_m erforderlich.

MOD $\rightarrow$ { { MOD_E $\rightarrow$ MOD_TUP_E } $\rightarrow$ AFP_E }: Bei der Einführung des Tupelkonzepts von *DBMR* haben wir einerseits als Zuordnungsfunktionen das Modell an sich andererseits eine allfällig im gewünschten Zusammenhang definierte Auswahlfunktion φ_m.

Durch die Entität MOD_TUP_E beschreiben wir die Modellparametermenge. Um die einzelnen Bildmengen $\bar{Z}_{\varphi_m}$ ansprechen zu können, führen wir eine Auszeichnung dieser Bildmengen mittels eines Indikators durch. Ebenso vereinbaren wir, dass bei undefinierter Auswahlfunktion φ_m derartige Modell mit einer der identischen Abbildung äquivalenten Auswahlfunktion ergänzt werden. Diese Übereinkunft gestattet uns, sämtlichen Modellen ein geeignetes φ_m zuzuordnen.

AFP $\rightarrow$ { AFP_E $\rightarrow$ { AFP_TUP_E $\rightarrow$ AFP_MGL_E } $\rightarrow$ ATM_E }: Mit diesen Relationen beschreiben wir die einem Modell zugeordnete Funktion φ_m. Hierin ist ebenso die oben angeführte identische Abbildung, als auch die flexible Zuordnung von Modellgleichungen zu den Elementen der Modellparametermenge enthalten.

$$
\begin{aligned}
& m(a_1, a_2, \ldots, a_n, \varphi_m) \\
& \qquad (a_1, \ldots, a_n)_{\varphi_m^1} \quad \rightarrow \quad \bar{Z}_{\varphi_m^1} \\
& \qquad\qquad\qquad\qquad \vdots \\
& \qquad (a_1, \ldots, a_n)_{\varphi_m^l} \quad \rightarrow \quad \bar{Z}_{\varphi_m^l}
\end{aligned}
$$

bilden wir folgendermassen in das Tupelschema ab

$$
\begin{aligned}
a_1, a_2, \ldots, a_n \quad & \in \quad \text{ATM\_E} \\
(a_1, \varphi_m^1, \mathrm{val}(a_1, \varphi_m^1), \mathrm{MGL}(\varphi_m^1)) \quad & \in \quad \text{AFP\_TUP\_E} \\
& \vdots
\end{aligned}
$$

$$
\begin{aligned}
(a_n, \varphi_m^l, \mathrm{val}(a_n, \varphi_m^l), \mathrm{MGL}(\varphi_m^n)) &\in \text{AFP\_TUP\_E} \\
(m, \varphi_m, \text{AFP\_TUP\_E}(\varphi_m^{1,\ldots,l})) &\in \text{AFP\_E} \\
(\varphi_m^1, \mathrm{MGL}(\varphi_m^1)) &\in \text{AFP\_MGL\_E} \\
&\vdots \\
(\varphi_m^n, \mathrm{MGL}(\varphi_m^n)) &\in \text{AFP\_MGL\_E}
\end{aligned}
$$

AFP_TUP_E → MGL_E: Wie bei der Beschreibung der Auswahlfunktion modellieren wir mittels dieser Entitäten die Funktion κ, die Zuordnung von Zustandsmengen zu den Elementen atomarer Mengen.

ZSM → { ZSM_E → ZSM_TUP_E }: Wir haben die Zustandsmenge als Element des Zustandsraums, der wiederum als karthesisches Produkt atomarer Mengen definiert wird, beschrieben. Hier ist Normalisiertheit nicht gegeben, daher schreiben wir wir für die Zuordnung

$$a \to ((b_1^1, \ldots, b_{n_1}^1), \ldots, (b_1^l, \ldots, b_{n_l}^l))$$

$$
\begin{aligned}
(a, \text{ZSM\_E}(a)) &\in \text{MGL\_E} \\
(b_1^1, \mathrm{ind}(b_1^1)) &\in \text{ZSM\_TUP\_E} \\
&\vdots \\
(b_{n_l}^l, \mathrm{ind}(b_{n_l}^l)) &\in \text{ZSM\_TUP\_E} \\
(\text{ZSM\_TUP\_E}(a)) &\in \text{ZSM\_TUP\_E}
\end{aligned}
$$

γ **→ { SIM_E → SIM_TUP_E }:** Im Abschnitt 2.6.4 haben wir die Definition der Simulation im Sinne von *DBMR* derart erklärt, dass durch eine Funktion γ den Modellgleichungen eines Modells $\hat{m}$ und den durch sie beschriebenen atomaren Mengen Vektoren aus $\mathbf{C}^n$ zugeordnet werden.
Sei nun $\gamma_{\hat{m}}$ eine Simulation des Modells $\hat{m}$ mit Modellparametermenge $\{\hat{a}_1, \ldots, \hat{a}_n\}$ und freiwählbarer Pamrametermenge $\{\hat{p}_1, \ldots, \hat{p}_l\}$, dann er-

geben sich aufgrund der Anwendung von γ folgende Tupel:

$$\begin{aligned}
(\mathrm{SIM}(\gamma), 1, \hat{m}, Z^1_{\hat{A}}, Z^1_{\hat{P}}) &\in \mathrm{SIM_TUP_E}(\gamma) \\
&\vdots \\
(\mathrm{SIM}(\gamma), \mathrm{ind}(\gamma), \hat{m}, Z^{\mathrm{ind}(\gamma)}_{\hat{A}}, Z^{\mathrm{ind}(\gamma)}_{\hat{P}}) &\in \mathrm{SIM_TUP_E}(\gamma) \\
(\mathrm{SIM}(\gamma)) &\in \mathrm{SIM_E}
\end{aligned}$$

$Z_{\hat{A}} \rightarrow$ **{ PRM_E $\rightarrow$ VALUE_E $\rightarrow$ DIM_E $\rightarrow$ DIM_TUP_E }:** Diese „Relationenkette“ beschreibt die Zuordnung der Vektoren aus $Z_{\hat{A}}$ zu der entsprechenden Simulation γ. Für die Herleitung der Notation verweisen wir wiederum auf Abschnitt 2.6.4 und können daher schreiben

$$\zeta_{\hat{a}_j} \in \mathbf{C}^{n_{\hat{a}_j}}$$

Die Dimensionsangabe wird folgendermaßen in den zugehörigen Relationen eingefügt

$$\begin{aligned}
(1,1) &\in \mathrm{DIM_TUP_E} \\
&\vdots \\
(1, n_{\hat{a}_j}) &\in \mathrm{DIM_TUP_E} \\
\mathrm{DIM_TUP_E}(n_{\hat{a}_j}) &\in \mathrm{DIM_E}
\end{aligned}$$

Schließlich werden die Elemente des Vektors gebildet

$$\begin{aligned}
\mathrm{VALUE_E}(\zeta_{\hat{a}_j}) &\in \mathrm{PRM_E} \\
(1, \mathrm{DIM_TUP_E}(1), \zeta^1_{\hat{a}_j}) &\in \mathrm{VALUE_E}(\zeta_{\hat{a}_j}) \\
&\vdots \\
(n_{\hat{a}_j}, \mathrm{DIM_TUP_E}(n_{\hat{a}_j}), \zeta^{n_{\hat{a}_j}}_{\hat{a}_j}) &\in \mathrm{VALUE_E}(\zeta_{\hat{a}_j})
\end{aligned}$$

$Z_{\hat{P}} \rightarrow$ **{ CWR_E $\rightarrow$ VALUE_E $\rightarrow$ DIM_E $\rightarrow$ DIM_TUP_E }:** Analog obigem Punkt wird für die Elemente der frei wählbaren Parametermenge

vorgegangen

$$\zeta_{\hat{p}_j} \in \mathbf{C}^{n_{\hat{p}_j}}$$

Die Dimensionsangabe wird folgendermaßen in den zugehörigen Relationen eingefügt

$$\begin{aligned} (1,1) &\in \text{DIM\_TUP\_E} \\ &\vdots \\ (1, n_{\hat{p}_j}) &\in \text{DIM\_TUP\_E} \\ \text{DIM\_TUP\_E}(n_{\hat{p}_j}) &\in \text{DIM\_E} \end{aligned}$$

Schließlich werden die Elemente des Vektors gebildet

$$\begin{aligned} \text{VALUE\_E}(\zeta_{\hat{p}_j}) &\in \text{PRM\_E} \\ (1, \text{DIM\_TUP\_E}(1), \zeta^1_{\hat{p}_j}) &\in \text{VALUE\_E}(\zeta_{\hat{p}_j}) \\ &\vdots \\ (n_{\hat{p}_j}, \text{DIM\_TUP\_E}(n_{\hat{p}_j}), \zeta^{n_{\hat{p}_j}}_{\hat{p}_j}) &\in \text{VALUE\_E}(\zeta_{\hat{p}_j}) \end{aligned}$$

4.2 Beschreibung der Syntax mit Beispielen für die Anwendung

Anhand einiger Beispiele soll nun der Aufbau der Modellbeschreibungssprache *MQL* beschrieben werden. Es wird gezeigt, wie ein Modell definiert wird und da dies ja auch ein Hauptaugenmerk der Simulation ist, einige Beispiele für Experimente (Modellverknüpfung, Parametervariation, Extremaaufsuche, etc.) dargestellt.

Beispiel 1:

$$\begin{aligned} y' &= a_1 x - b_1 y \\ x' &= a_2 y - b_2 y \end{aligned}$$

Das Modell wird durch folgendes `CREATE` Statement beschrieben:

```
CREATE MODEL BEISPIEL1
( X      STATE      WITH DEFAULT,
  Y      STATE      WITH DEFAULT,
  A( 2 ) FLOAT      WITH DEFAULT ( 1, 2 ),
  B( 2 ) FLOAT      WITH DEFAULT ( 2, 3 ),
  EQU
  (  DRV( Y ) = A( 1 )*X - B( 1 )*Y,
     DRV( X ) = A( 2 )*Y - B( 2 )*X ),
  INP( A, B ), OUTP ( X, Y ) );
```

Als Experiment soll nun eine Parametervariation des Parameters a_1 durchgeführt werden:

```
SELECT X, Y FROM BEISPIEL1 VARYING A( 1 ) FROM 1 TO 5 BY 0,5;
```

Um nun die Modellverknüpfung (*DBMR*) zu zeigen, nehmen wir ein anderes Modell, das einen Schwingungsvektor erzeugt und dann mit dem oben angegebenen Beispiel 1 verknüpft wird.

Beispiel 1A:

```
CREATE MODEL BEISPIEL1A
(  A( 2 ) STATE WITH DEFAULT ( 1, 1 ),
   EQU (
      DRV( A ) = - A ),
   OUTP( A ) );
```

Nun führen wir oben genanntes Experiment durch. Durch den `SELECT` Befehl werden unter Angabe einer `WHERE` Bedingung die beiden Modelle verknüpft. Die Verknüpfung erfolgt über die Gleichsetzung, wobei die Gleichsetzung nicht nur auf skalare Parameter beschränkt ist, sondern auch gleichartig für Vektoren gültig ist:

```
SELECT X, Y FROM BEISPIEL1 M1 MODEL JOIN BEISPIEL1A M2 ON M1.A = M2.A;
```

Das Herstellen der Modellverknüpfung (=) ist im Fall von *MQL* keine Gleichsetzung im herkömmlichen Sinn, sondern das Zuführen von Werten von einem Modell in ein anderes. Sind die Variablen, die durch diese Gleichsetzung (=)

verbunden werden, in INP bzw. OUTP Statements deklariert, dann werden die Werte von OUTP Variablen in die INP Variablen übergeführt. Erfolgt die Modellverknüpfung über Variablen, die nicht derart deklariert sind, dann erfolgt eine Zuweisung von der rechten zur linken Seite.

Beispiel 2:

Im folgenden sollen nun die erweiterten Kontrollstrukturen dargestellt werden. Diese ermöglichen einerseits die Erstellung von Makros, andererseits die einfache Umschaltung zwischen Teilmodellen. Im weiteren wird ebenso gezeigt werden, daß dieses Umschalten zwischen Teilmodellen entweder in einem Modell, in dem die verschiedenen Teilmodelle ausprogrammiert sind oder durch Modellverknüpfung (*DBMR*) durchgeführt werden kann, indem die verschiedenen Teilmodelle bei der Durchführung des Experiments verknüpft werden. Wir betrachten folgendes Modell:

$$\begin{aligned} \forall x < a \colon x' &= x - \sin(x) \\ \forall x \geq a \colon x' &= x \end{aligned}$$

Version 1: Die Teilmodelle werden in einem *MQL* CREATE Statement deklariert:

```
CREATE MODEL BEISPIEL2V1
( A     FLOAT   WITH DEFAULT,
  X     STATE   WITH DEFAULT,
  EQU (
     IF X < A
DRV( X ) = X - SIN( X )
     ELSE
DRV( X ) = X
     ENDIF
  ),   INP( A ), OUTP( X ) );
```

Auch dieses Beispiel kann über ein SELECT Statement zur Simulation aufgerufen werden. Eventuelle andere Parameterwerte als im WITH DEFAULT Abschnitt der können im PARAMETERS Abschnitt des SELECT Statements angegeben werden:

```
SELECT X FROM BEISPIEL1V1 PARAMETERS A=1, X=1, T0 = 0, TEND = 10;
```

Bei der Angabe von Zustandsvariablen (`STATE`) wird durch den Wert in dem `PARAMETERS` Abschnitt der Anfangswert angegeben.
Version 2: Jetzt werden die beiden Teilmodelle in separaten `CREATE` Statements angegeben.

```
CREATE MODEL BEISPIEL2V2A
( A     FLOAT   WITH DEFAULT, X     STATE   WITH DEFAULT,
  EQU (
    DRV( X ) = X - SIN( X ) ),
  INP( A ), OUTP( X ) );
CREATE MODEL BEISPIEL2V2B
( A     FLOAT   WITH DEFAULT, X     STATE   WITH DEFAULT,
  EQU (
     DRV( X ) = X ),
  INP( A ), OUTP( X ) );
```

Die folgende Modellverknüpfung erfolgt folgendermaßen:

```
SELECT X FROM BEISPIEL2V2A WHERE X < A PARAMETERS A = 1
UNION
SELECT X FROM BEISPIEL2V2B WHERE X >= A PARAMETERS A = 1;
```

Das `UNION` Statement wird verwendet, um Experimente „gleichzeitig“ durchführen zu können; diese Experimente werden derart zusammengefaßt, als ob sie in einem Modell beschreiben seien. Diese Verknüpfung, deren Ausprägung im Konzept der *DBMR* beschrieben ist, kann in mehreren Arten auftreten bzw. ausgewertet werden. Wir unterscheiden daher zwei Kommandos: `UNION UNCOUPLED` und `UNION COUPLED` (Default)
Bei `UNION UNCOUPLED` werden die Experimente, die durch dieses `UNION` Statement verbunden werden parallel ausgeführt. Im Sinne dieser Parallelität werden mehrere Experimente gleichzeitig durchgeführt und stehen dann in der Experimentdatenbank [1] zur Verfügung und können danach zur Anzeige gebracht werden.
`UNION COUPLED` ermöglicht ein Experimentieren, bei dem die Zustandsgrößen

[1] In Kapitel 2 haben wir dafür die Entität Simulation (*SIM*) definiert

laut ihrer Reihenfolge in den zusammengefaßten `SELECT` Statements als Ein- bzw. Ausgangsgrößen der folgenden `SELECT` Statements dienen. Beim Zusammenfassen der „Teilexperimente“ erfolgt die Berücksichtigung der angegebenen `WHERE` bzw. `MODEL JOIN` Bedingung, die eine logische Gültigkeit des Teilexperiments angibt. So wird, wenn wir die Integration in der Zeit betrachten, zu Beginn mit dem Teilexperiment begonnen, das zum Zeitpunkt t_0 gültig ist (hier ist, bei mehreren Teilexperimenten, die zugleich gültig sind, die Reihenfolge der angegebenen `SELECT` Statements relevant; Im Gegensatz dazu ist das `UNION UNCOUPLED` zu verstehen, das ja eine gleichzeitige Durchführung der Teilexperimente zur Folge hat).

Es liegt in der Natur der Anforderungen an die kontinuierliche Simulation, dass Aufsuchen von bestimmten Zuständen des Modells aufgesucht werden sollen. Diese Zustände werden durch die Angabe der entsprechenden Funktion aufgerufen, wozu auch die Angabe der Parameter zu erfolgen hat, deren Variation das Erreichen des gewünschten Stadiums erzielen soll. Als Beispiel sei das `SELECT` Statement angeführt, das ein Aufsuchen einer Nullstelle einer Zustandgrösse durchführt, wobei die angegebenen Parameter zu variieren sind:

```
SELECT ZERO( X ) FROM MODEL1 VARYING A, B;
```

Analoges Vorgehen erfolgt bei einem Aufruf der Funktionen, die Extremaaufsuche (`OPTIM`), das Aufsuchen des Stabilitätspunktes (`STEADYSTATE`) und ähnlichen Funktionen.

Um Teilfunktionen flexibel realisieren zu können, erlaubt *MQL* auch die Definition von Makros. Diese Syntaxelemente werden in der Makrodatenbank abgelegt, stehen daher allen Modellen zur Verfügung. Da Makros im Sinne von *DBMR* Objekte sind, werden auch diese mittels eines `CREATE` Statements erzeugt. Da die Makroverarbeitung eine textuelle Ersetzung während der Verarbeitung darstellt, ist ein Makro nicht an einen bestimmten Typ von Eingangsgröße gebunden; die Namen der in den `INP` bzw. `OUTP` Blöcken angegebenen Parameter werden durch die im Makroaufruf angegebenen Ausdrücke (das können Zustandsvariablen, Parameterwerte, oder Ausdrücke an sich sein) ersetzt. Soll ein Wert zurückgegeben werden, dann ist dem symbolischen Namen `MACROVALUE` ein Wert zuzuweisen. Dieser wird dann dem Parameter, der durch

eine Makroauswertung zu bestimmen ist, zugewiesen. Falls „Zwischenwerte“ erforderlich sind, müssen diese mit Type `VOID` deklariert werden. Die Erzeugung eines Makros erfolgt wieder mit einem `CREATE` Statement, wobei in diesem Fall der Objekttyp (nahliegenderweise) `MACRO` ist.

Beispiel 3:

Ein Beispiel für ein Makro, das eine Maximumsfunktion von drei Werten feststellt, sei nun im folgenden angegeben. Es werden dann die zusätzlichen Erläuterungen angegeben:

```
CREATE MACRO MAXIMUM3A
(  D VOID,
   EQU
   (  D = MAX( A, B ),
      MACROVALUE = MAX( D, C ) ),
   INP( A, B, C ) );
```

Da natürlich Ausdrücke, die in Makros oder Funktionen eingehen, beliebig formulierbar sind, sei nun als zweite Möglichkeit eine Version des oben angeführten Makros angegeben, die darausfolgend auf eine Zwischenvariable verzichtet:

```
CREATE MACRO MAXIMUM3B
(
   EQU (
     MACROVALUE = MAX( MAX( A, B ), C ) ),
   INP( A, B, C ) );
```

Wenn ein Modell ein Makro aufrufen soll, dann genügt es, den Namen des Makros anzugeben, wobei in der folgenden Klammer die Parameter, falls in `INP` bzw. `OUTP` Blöcken deklariert, angegeben werden. Stellt das Makro eine Funktion dar, die einen Wert an das aufrufende Modell zurückliefern soll, wird der gewünschten Zustands- oder Parametervariable der Wert in gewohnter Form zugewiesen. Als Beispiel für die Verwendung eines Funktionenmakros betrachten wir folgendes Modell:

$$x' \quad = \quad x - y + \sin(z)$$

$$
\begin{aligned}
y' &= -z + \ln(x + y) \\
z' &= \frac{-xz}{y} \\
\forall t < \alpha\colon d &= \max(x, y, z) \\
\forall t \geq \alpha\colon d &= \max(y, z, \beta)
\end{aligned}
$$

```
CREATE MODEL BEISPIEL3
(
  ALPHA FLOAT   WITH DEFAULT,  BETA  FLOAT   WITH DEFAULT,
  X     STATE   WITH DEFAULT,  Y     STATE   WITH DEFAULT,
  Z     STATE   WITH DEFAULT,
  EQU
  ( DRV( X ) = X - Y + SIN( Z ),
    DRV( Y ) = Z + LN( X + Y ),
    DRV ( Z ) = ( X * Y ) / Y,
    IF T < ALPHA
      D = MAXIMUM3B( X, Y, Z )
    ELSE
      D = MAXIMUM3B( Y, Z, BETA )
    ENDIF
  ),
  INP( ALPHA, BETA ), OUTP( X, Y, Z )
);
```

Beispiel 4:

Eine oft wiederkehrende Problemstellung ist es, daß in einem Modell mehrere gleichgestaltige Gleichungen in einem Zusammenhang stehen, der dann die Gesamtheit dieses Modells beschreibt. Es sei folgendes Beispiel gegeben:

$$
\begin{aligned}
x' &= \alpha x - \beta y + z \\
y' &= \chi y - \delta y z + x \\
z' &= \phi z + \gamma z y - y
\end{aligned}
$$

Um dieses Modell effizient beschreiben zu können, wird ein Makro kreiert, das eine Teilgleichung beschreibt; dieses Makro wird dann vom folgenden `CREATE MODEL` Statement verwendet:

```
CREATE MACRO BSP4_MAKRO
(  INP( X, Y, Z, A, B ),
   EQU (
     DRV(X) = A * X + B * X * Y + Z ) );
```

Basierend auf die Möglichkeit, diese Makro zur Formulierung der beschreibenden Differentialgleichungen verwenden zu können, wird nun das *MQL* Statement angegeben:

```
CREATE MODEL BEISPIEL4
(
   INP( A, B, C, D, E, F ),
   OUTP( X, Y, Z ),
   A    FLOAT, B    FLOAT, C    FLOAT, D    FLOAT,
   E    FLOAT, F    FLOAT,
   X    STATE, Y    STATE, Z    STATE,
   EQU (
      BSP4_MAKRO( X, Y, Z, A, B),
      BSP4_MAKRO( Y, Z, X, C, D),
      BSP4_MAKRO( Z, Y, -Y, F, -G) )
);
```

Beispiel 5:

Wenn bereits eine große Anzahl von „Standardmodellen“ in der Datenbank abgelegt sind, können diese, angelehnt an den objektorientierten Ansatz, von weiteren Modellen verwendet werden; hier ist der Aufruf analog eines Makros durchzuführen. Ein Makro kann wie auch ein Modell Ein- und Ausgangsparameter haben. Es erfolgt der Aufruf eines Modells durch Angabe des Modellnamens und der gewünschten Parameter.

Folgendes Gleichungssystem sei nun durch die Anwendung des Aufrufs eines

Modells aus einem Modell heraus in *MQL* dargestellt:

$$\begin{aligned} x' &= -ax \\ y' &= \cos(y) - x\frac{w}{t} \end{aligned}$$

Um nun die Möglichkeit eines „Model-Calls“ zu zeigen, wird die erste Gleichung als eigenes Modell programmiert. Dieses wird dann über den „Model-Call“ Mechanismus aus einem zweiten `CREATE MODEL` Statement als Teil des gesamten Gleichungssystems aufgerufen.

```
CREATE MODEL BSP5_TEIL1
(
   A FLOAT WITH DEFAULT,
   X STATE WITH DEFAULT,
   INP(A), OUTP(X),
   EQU
   (
      DRV(X,2) = -A*X
   )
);
```

Der zweite Teil dieses Beispiels wird nun dargestellt:

```
CREATE MODEL BSP5_TEIL2
(
   FLOAT A,
   FLOAT W,
   STATE X,
   STATE Y,
   INP( A, W ), OUTP( X, Y ),
   EQU
   (
      BSP5_TEIL1(A, X),
      DRV(Y) = COS( Y ) - X * (W/T)
   )
```

```
);
```

Als besonderen Aspekt der Modellbeschreibungsprache *MQL* haben wir die Modellverknüpfung angeführt; daneben ist aber auch der flexible IO Mechanismus ein Punkt, der in diesem System eine Realisierung gefunden hat. Mit diesem Auswahlverfahren von modellbeschreibenden Gleichungen auf der Experimentsebene ist eine einfache Realisierung von z.b. Bondgraphenmodellen möglich. Um aber dieses Verfahren zu erläutern, wählen wir ein einfaches „akademisches“ Beispiel.

Beispiel 6:

Ein System bestehe aus den zeitvarianten Parametern x_1, x_2, x_3, x_4 und den Parametern p_1, p_2, p_3.

Falls an x_1 und p_1 Eingangsgrößen (Konstant im Falle von p_1, bzw. z.B. eine zeitvariante Größe, wie eine Schwingung oder Spannung, im Falle von x_1 seien die funktionalen Zusammenhänge zwischen diesen Parametern in der Art:

$$\begin{aligned} x_2' &= p_1 x_1 + p_2 x_3 x_4 \\ x_3' &= p_3 x_2' - p_4 x_2 \\ x_4' &= -x_1 \end{aligned}$$

Falls an x_2, p_1, p_4 Eingangsgrößen anliegen, dann sei:

$$\begin{aligned} x_1' &= p_1 x_1 + p_2 x_3 x_4 \\ x_3' &= p_3 x_1' - p_4 x_2 \\ x_4' &= -x_1 \end{aligned}$$

Diesen Sachverhalt kodieren wir in *MQL* folgendermaßen:

```
CREATE MODEL BEISPIEL6 (
X1      STATE,  X2      STATE,  X3      STATE,
X4      STATE,  P1      FLOAT,  P2      FLOAT,
P3      FLOAT,  P4      FLOAT,
EQU (
CASE WHEN INP(X1) AND INP(P1) THEN
```

```
DRV(X2) = P1 * X1 + P2 * X3 * X4,
DRV(X3) = P3 * DRV(X2) - P4* X2,
DRV(X4) = -X1
     WHEN INP(X2) AND INP(P1) AND INP(P4) THEN
DRV(X1) = P1 * X1 + P2 * X3 * X4,
DRV(X3) = P3 * DRV(X1) - P4 * X2,
DRV(X4) = -X1
END))
```

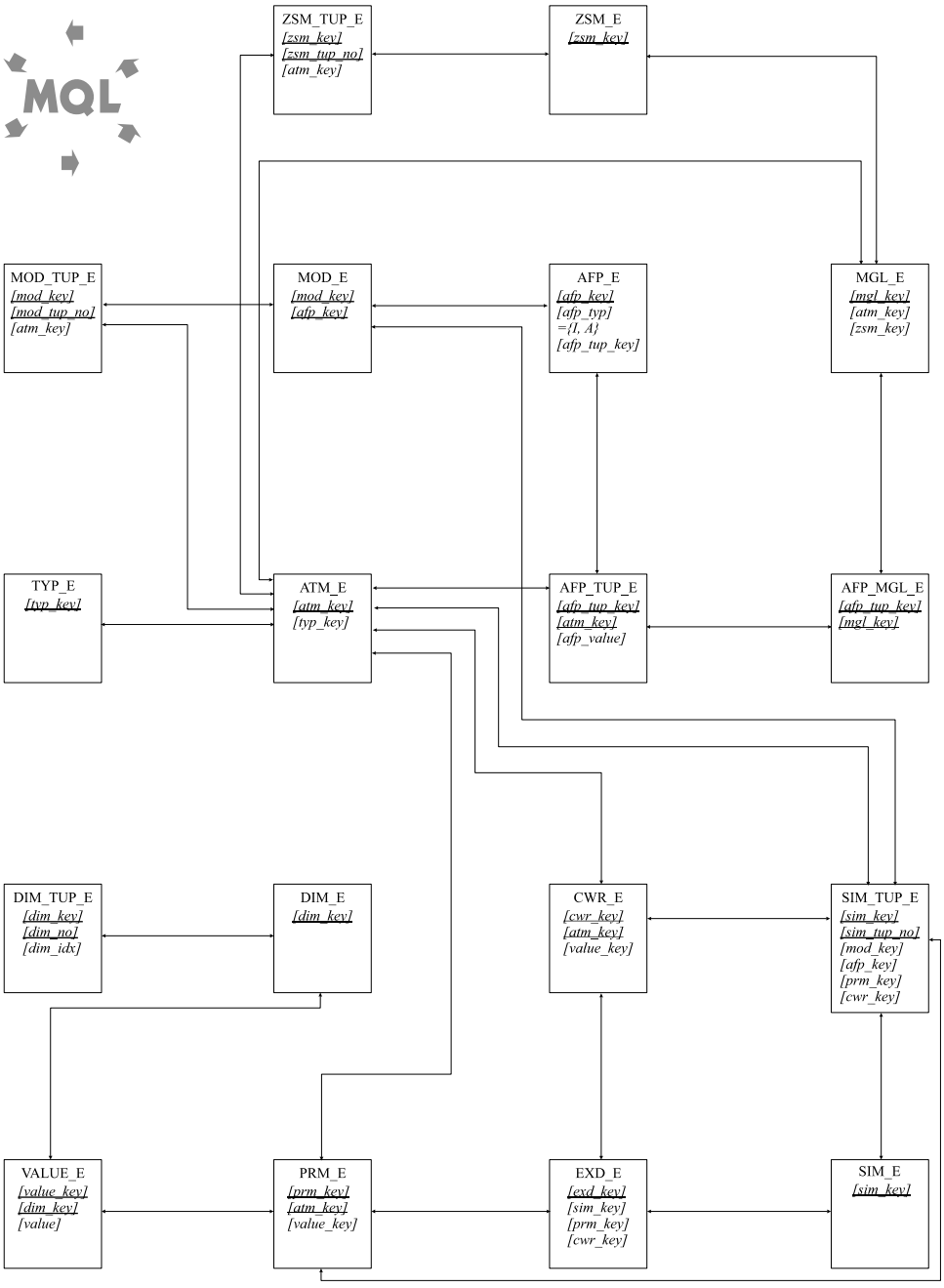


Abbildung 4.1: Das Datenbankschema von *DBMR* in dritter Normalform

Kapitel 5

Das Konzept der Realisierung – *DABEQS - Databased equation Solver*

5.1 Die Auswahl der zu realisierenden Teilmenge von *MQL*

Für das experimentelle System *DABEQS* wählen wir die Elemente von *MQL* aus, die für die Beschreibung von Modellen und Experimenten erforderlich sind. Das sind

`CREATE`-Statement Mit dieser Klausel der Sprache *MQL* werden Modelle analysiert und in den entsprechenden Relationen abgelegt. Nach Überprüfung von allfälliger gleichartiger Existenz erfolgt die Analyse des Modells und die Erstellung der Tupelstruktur.

`SELECT`-Statement Diese Klausel führt einen Zugriff auf die *DBMR* Relationen durch. Durch die Verbindung von Modellen, die ebenso durch dieses Statement beschrieben wird, wird ein Modell $\hat{m}$ erzeugt. Dieses Ergebnismodell steht dann für die Anwendung von *DBI/DBC*, die Berechnung eines Simulationsergebnisses oder den Export in eine Zielsprache zur Verfügung.

`EXPORT`-**Statement** Nachdem durch die `SELECT` Klausel ein Modell $\hat{m}$ aufgebaut wurde, können dessen Tupel mittels dieser Anweisung in eine Modellbeschreibung einer Zielsprache überführt werden. Es müssen die Eigenheiten von *DBMR* (wie die Auswahlfunktionen φ_m, σ) in geeignete äquivalente Anweisungen übersetzt werden.

5.2 Einige Bemerkungen zum Implementationskonzept

Analog dem *Software Life-Cycle* werden wir hier ein Detailkonzept einer Implementation der in den vorigen Kapiteln angegebenen theoretischen Überlegungen angeben. Wir beschreiben sowohl die dahinterliegende Datenbank als auch die Verarbeitungslogik.

5.2.1 Das Relationenschema

Aus der Normalisierung des Relationenschemas und der Auswahl der zu analysierenden Teilmenge von *MQL* folgt die Art der erforderlichen Tabellen und deren Beziehungen untereinander.

Eine Namenskonvention für Datenbankobjekte

<*Kürzel*>**_E:** 3 stelliges *Kürzel* für die dadurch dargestellte Entität. (Eine Ausnahme stellt die Tabelle VALUE_E dar; Diese ist für die Speicherung von Simulationergebnisen und -parametern erforderlich)

<*Kürzel*>**_TUP_E:** Tupeltabelle zur Auflösung der NF^2 Sachverhalte.

<*Kürzel der „primären" Tabelle*>**_KEY:** Attributname bei generischem Schlüssel.

<*Kürzel der „primären" Tabelle*>_<*3 stellig frei*>**:** bei sonstigen Feldern $\Rightarrow$ Felder auf die eine Fremdschlüsselbeziehung zutrifft, heissen wie in der Ursprungstabelle.

<*Tabellenname*>**_PK oder _I***n***:** Bezeichnung des Primärindex (PK) oder eines Hilfsindex (In), wobei n eine fortlaufende Numerierung angibt.

Eine Beschreibung der Datenbankobjekte

Generische Schlüssel werden als 32stellige Zeichenketten angelegt, wobei diese analog Zahlen zur Basis 36 (sämtliche alphanumerische Zeichen) bei entsprechenden Neuanlagen inkrementiert werden.
Bei den Beziehungen ergibt sich eine allfällige Löschungsregel aufgrund der in Abschnitt 2.7 angegebenen Kardinalitäten.

MOD_E: Attribute: [*mod_bez*] – Modellbezeichnung – Zeichenkette(32)
[*mod_typ*] – Typ des Modells – Zeichenkette(1)

Schlüsselattribute: [*mod_key*] – Modellschlüssel – Zeichenkette(32)
[afp_key] – Auswahlfunktion φ_m – Zeichenkette(32)

MOD_TUP_E: Attribute: [*atm_key*] – Schlüssel atomare Menge – Zeichenkette (32)

Schlüsselattribute: [*mod_key*] – Modellschlüssel – Zeichenkette(32)
[*mod_tup_no*] – Positionsangabe in der Modellparametermenge – Integer

Beziehungen: MOD_E – Löschungsregel *unbedingt*
ATM_E – Löschungsregel *verbieten*

AFP_E: Attribute: [afp_tup_key] – Schlüssel Tupel Auswahlfunktion – Zeichenkette(32)
[afp_typ] – Typ der Auswahlfunktion (identische Abbildung, explizite Angabe einer Auswahlfunktion) – Zeichenkette(1)

Schlüsselattribute: [afp_key] – Schlüssel Auswahlfunktion – Zeichenkette(32)

Beziehungen: AFP_TUP_E – Löschungsregel *verbieten*
Gilt als Kernentität für AFP_TUP_E.

AFP_TUP_E: Attribute: [afp_value] – Art der Auswahl – Zeichenkette(1)

Schlüsselattribute: [afp_tup_key] – Schlüssel Auswahlfunktion – Zeichenkette(32)
[*atm_key*] – Schlüssel atomare Menge – Zeichenkette(32)

Beziehungen: ATM_E – Löschungsregel *verbieten*

AFP_MGL_E: Schlüsselattribute: [afp_tup_key] – Schlüssel Auswahlfunktion – Zeichenkette(32)
[mgl_key] – Schlüssel Modellgleichungen – Zeichenkette(32)

Beziehungen: ATM_TUP_E – Löschungsregel *unbedingt*
MGL_E – Löschungsregel *verbieten*

MGL_E: Attribute: [atm_key] – Schlüssel der atomaren Menge – Zeichenkette(32)
[zsm_key] – Schlüssel der Zustandsmenge – Zeichenkette(32)

Schlüsselattribute: [mgl_key] – Schlüssel der Modellgleichung – Zeichenkette(32)

Beziehungen: ATM_E – Löschungsregel *verbieten*
ZSM_E – Löschungsregel *verbieten*

ZSM_E: Attribute: [zsm_typ] – Typ der Zustandmenge – Zeichenkette(32)

Schlüsselattribute: [zsm_key] – Schlüssel der Zustandsmenge – Zeichenkette(32)

Beziehungen: Gilt als Kernentität für ZSM_TUP_E.

ZSM_TUP_E: Attribute: [atm_key] – Schlüssel der Zustandsmenge – Zeichenkette(32)

Schlüsselattribute: [zsm_key] – Schlüssel der Zustandsmenge – Zeichenkette(32)
[zsm_tup_no] – Position der Tupelkompononente – Integer

Beziehungen: ATM_E – Löschungsregel *verbieten*
ZSM_E – Löschungsregel *verbieten*

ATM_E: Attribute: [typ_key] – Schlüssel der Zustandsmenge – Zeichenkette(32)
[$value\_key$] – Schlüssel des Wertes – Zeichenkette(32)

Schlüsselattribute: [atm_key] – Schlüssel der atomaren Menge – Zeichenkette(32)

Beziehungen: TYP_E – Löschungsregel *verbieten*

TYP_E: Attribute: [*typ_ptr*] – Zeiger auf die den Typ enthaltende Entität – Zeichenkette(32)
[*typ_ent*] – Angabe der Typsentität – Zeichenkette(3)

Schlüsselattribute: [*typ_key*] – Schlüssel des Typs – Zeichenkette(32)

Beziehungen: [*typ_ent*]_E – Löschungsregel *verbieten*
Diese Beziehung ist durch geeignete Implementierungslogik vorzusehen, da ein RDBMS diese variablen Fremdschlüsselattribute nicht beherrscht.

SIM_E: Attribute: [*sim_bez*] – Bezeichnung der Simulation – Zeichenkette(32)

Schlüsselattribute: [*sim_key*] – Schlüssel der Simulation – Zeichenkette(32)

Beziehungen: Gilt als Kernentität für SIM_TUP_E.

SIM_TUP_E: Attribute: [*mod_key*] – Schlüssel des Modells – Zeichenkette(32)
[afp_key] – Schlüssel der Auswahlfunktion φ_m – Zeichenkette(32)
[prm_key] – Schlüssel des Parametervektors – Zeichenkette(32)
[cwr_key] – Schlüssel des Wertevektors des Ergebnisses – Zeichenkette(32)

Schlüsselattribute: [*sim_key*] – Schlüssel der Simulation – Zeichenkette(32)
[*sim_tup_no*] – Position des Simulationstupels – Integer

Beziehungen: SIM_E – Löschungsregel *unbedingt*
MOD_E – Löschungsregel *verbieten*
PRM_E – Löschungsregel *verbieten*
CWR_E – Löschungsregel *verbieten*

EXD_E: Attribute: [*sim_key*] – Schlüssel des Wertevektors – Zeichenkette(32)

[*prm_key*] – Schlüssel des primären Parametervektors – Zeichenkette(32)

[*cwr_key*] – Schlüssel des primären Wertevektors des Ergebnisses – Zeichenkette(32)

Schlüsselattribute: [*exd_key*] – Schlüssel der Experimentsdatenbank – Zeichenkette(32)

Beziehungen: SIM_E – Löschungsregel *unbedingt*

PRM_E – Löschungsregel *verbieten*

CWR_E – Löschungsregel *verbieten*

Gilt als Zusammenfassung der Entität SIM_E.

PRM_E: Attribute: [*value_key*] – Schlüssel des Wertevektors – Zeichenkette(32)

Schlüsselattribute: [*prm_key*] – Schlüssel des Parametervektors – Zeichenkette(32)

[*atm_key*] – Schlüssel der atomaren Menge – Zeichenkette(32)

Beziehungen: ATM_E – Löschungsregel *unbedingt*

VALUE_E – Löschungsregel *unbedingt*

CWR_E: Attribute: [*value_key*] – Schlüssel des Wertevektors – Zeichenkette(32)

Schlüsselattribute: [*cwr_key*] – Schlüssel des Wertevektors des Ergebnisses – Zeichenkette(32)

[*atm_key*] – Schlüssel der atomaren Menge – Zeichenkette(32)

Beziehungen: ATM_E – Löschungsregel *unbedingt*

VALUE_E – Löschungsregel *unbedingt*

VALUE_E: Attribute: [*value*] – Wert an der angegebenen Dimensionsposition – Komplex mit doppelter Genauigkeit

Schlüsselattribute: [*value_key*] – Schlüssel des Wertevektors – Zeichenkette(32)

[*dim_key*] – Schlüssel der Dimensionsposition – Zeichenkette(32)

Beziehungen: DIM_E – Löschungsregel *verbieten*

DIM_E: Schlüsselattribute: [*dim_key*] – Schlüssel der Dimensionsposition – Zeichenkette(32)

Beziehungen: Gilt als Kernentität für DIM_TUP_E.

DIM_TUP_E: Attribute: [*dim_idx*] – Index an der Dimensionsposition – Integer

Schlüsselattribute: [*dim_key*] – Schlüssel der Dimensionsposition – Zeichenkette(32)
[*dim_no*] – Position der Dimensionsangabe – Zeichenkette(32)

Beziehungen: DIM_E – Löschungsregel *unbedingt*

5.2.2 Die Durchführung der Operationen

Kreieren eines Modells

Die Anweisung zur Erstellung eines Modells (`CREATE`) erstellt einen Baum, der das Modell beinhaltet. Kann diese Datenstruktur nicht erstellt werden, erfolgt die Ausgabe der in Abschnitt 7.1.2 entsprechend angegebenen Meldung. Die Blätter des Baumes werden durch Elemente atomarer Mengen, Analogien zu den funktionalen Mechanismen von *DBMR* und Angaben zum Modell gebildet. Folgende Prüfungen müssen erfolgen, dass ein Eintrag des Modells und der darausfolgenden Tupel in die entsprechenden Entitäten erfolgen kann.

1. $\pi_{[mod\_key]}(\sigma_{[mod\_key]=m}(MOD\_E))$ um die geforderte Eindeutigkeit der Modellbezeichnung und so des Modells zu überprüfen.

2. $\pi_{[atm\_key]}(\sigma_{[atm\_key]=a_i \wedge [typ\_key]=\mathrm{typ}(a_i)}(ATM\_E))$ um die geforderte Eindeutigkeit der Angaben über die Elemente der Modellparametermenge zu überprüfen. a_i steht für sämtliche derartige Größen in der Modelldefinition.

3. Wir bezeichnen die zu untersuchende Modellgleichung mit κ, die dazu gehörigen Elemente mit $A_\kappa = \{a_\kappa^1, \ldots, a_\kappa^n\}$. Für diese Elemente erfolgt ein Zugriff analog Punkt 2.

4. Wurde eine Auswahlfunktion φ_m durch die Angabe einer `CASE` Klausel angeführt, gehen wir folgendermaßen vor: Die jeweilige `INP`/`OUTP` Klausel

beschreibt, unter welchen Bedingungen die darin beschriebene Menge von Modellgleichungen anzuwenden ist. Wiederum erfolgt eine Prüfung der spezifizierten atomaren Mengen gemäß Punkt 2. Die Menge der Modellgleichungen haben wir bereits in Punkt 3 verifiziert.

Konnten alle Verifikationen erfolgreich durchgeführt werden, erfolgt das Einfügen der Tupel in die entsprechenden Relationen.

1. Einfügen der Modellparameter in die Entität der atomaren Mengen

$$\text{ATM\_E} = \text{ATM\_E} \cup \{(a_i, \text{TYP\_E}(a_i), \text{VALUE\_E}(a_i))\}$$

2. Einfügen des Modells in dessen Entitäten

$$\begin{aligned} \text{MOD\_E} &= \text{MOD\_E} \cup \{(m, \mathcal{C}, \text{AFP\_E}(m))\} \\ \text{MOD\_TUP\_E} &= \text{MOD\_TUP\_E} \cup \{(m, \mathcal{C}, \text{AFP\_E}(m))\} \end{aligned}$$

3. Ist keine explizite Definition von φ_m erfolgt, dann muss die in Abschnitt 4.1 geforderte „identische Auswahlfunktion“ erstellt werden. Indizierte Operationen werden für sämtliche Elemente durchgeführt.

$$\begin{aligned} \text{AFP\_E} &= \text{AFP\_E} \cup \{(\text{AFP\_E}(m), \mathcal{I}, \text{AFP\_TUP\_E}(m, \mathcal{I})\} \\ \text{AFP\_TUP\_E} &= \text{AFP\_TUP\_E} \cup \{(\text{AFP\_TUP\_E}(m, \mathcal{I}), a_i, \mathcal{I})\} \\ \text{AFP\_MGL\_E} &= \text{AFP\_MGL\_E} \cup \{(\text{AFP\_TUP\_E}(m, \mathcal{I}), \text{MGL\_E}(a_i))\} \end{aligned}$$

4. Folgende Datenbankoperationen sind für die Erstellung einer explizit angegebenen Auswahlfunktion φ_m erforderlich.

$$\begin{aligned} \text{AFP\_E} &= \text{AFP\_E} \cup \{(\text{AFP\_E}(m), \mathcal{A}, \text{AFP\_TUP\_E}(m, \mathcal{A})\} \\ \text{AFP\_TUP\_E} &= \text{AFP\_TUP\_E} \cup \{(\text{AFP\_TUP\_E}(m, \mathcal{A}), a_i, \varphi_m(a_i))\} \\ \text{AFP\_MGL\_E} &= \text{AFP\_MGL\_E} \cup \{(\text{AFP\_TUP\_E}(m, \mathcal{A}), \text{MGL\_E}(a_i|\varphi_m))\} \end{aligned}$$

Abfragen von Modellen

Führen wir Abfragen und Verknüpfungen von in der Datenbank befindlichen Modellen durch, dann müssen wir die in Abschnitt 2.7.2 definierten Operationen für die Anwendung in der normalisierten Datenbank anpassen. Sind Verknüpfungen rekursiv aufgebaut [1], so muss ebenso rekursiv das endgültige Ergebnismodell $\hat{m}$ aufgebaut werden.
Zu Berechnung eines Ergebnisses (Anwendung der Funktion γ) muss schließlich in inverser Art zur Konstruktion der Tupel aus dem Syntaxbaum dieser nun aus den Tupeln erstellt werden, sodass durch interpretierende Auswertung Lösungsalgorithmen angewendet werden können.

1. Wir werten ein Modell $\hat{m}$ aus, bei dem durch die Angabe einer `WHERE` Klausel eine Auswahlfunktion φ_m angesprochen wird. Die Auswahl der Modellgleichungen erfolgt durch

$$\begin{aligned}\pi_{\text{MGL\_E.}[atm\_key],\text{ZSM\_TUP\_E.}[atm\_key],\text{ZSM\_TUP\_E.}[zsm\_tup\_no]}\\ (\sigma_{[mod\_key]=\hat{m}}(\text{MOD\_E}) \bowtie \text{AFP\_E}\\ \bowtie \sigma_{[afp\_value]=\text{val}(a_1,\varphi_{\hat{m}})\wedge\ldots\wedge\text{val}(a_l,\varphi_{\hat{m}})}(\text{AFP\_TUP\_E})\\ \bowtie \text{AFP\_MGL\_E} \bowtie \text{MGL\_E} \bowtie \text{ZSM\_E} \bowtie \text{ZSM\_TUP\_E})\end{aligned}$$

   Durch diesen Zugriff erhalten wir als Ergebnismenge alle Modellgleichungskonponenten, geordnet nach den Elementen der Modellparametermenge und der Position im Tupel.

2. Der Zugriff bei einer inneren Modellverknüpfung der Modelle m_1 und m_2 muss Elemente a_{1_k} durch Elemente a_{2_k} für $k = 1, \ldots, n$ ersetzen (Wir führen der Übersichtlichkeit wegen $\bar{Z}|m_r$ mit $r = 1, 2$ ein, die als Ergebnisse von Operationen innerhalb des Relationenschemas ebenfalls Relationen sind.):

$$\bar{Z}|m_r \quad = \quad \pi_{\text{MGL\_E.}[atm\_key],\text{ZSM\_TUP\_E.}[atm\_key],\text{ZSM\_TUP\_E.}[zsm\_tup\_no]}$$

[1] siehe dazu die Nichtterminale `JoinedModel` und `ModelSpec` in Abschnitt 7

$$
\begin{aligned}
&(\sigma_{[mod\_key]=m_1}(\text{MOD\_E}) \bowtie \text{AFP\_E} \\
&\bowtie \sigma_{[afp\_value]=\text{val}(a_{r_1},\varphi_{m_r})\wedge\ldots\wedge\text{val}(a_{r_n},\varphi_{m_r})}(\text{AFP\_TUP\_E}) \\
&\bowtie \text{AFP\_MGL\_E} \bowtie \text{MGL\_E} \bowtie \text{ZSM\_E} \bowtie \text{ZSM\_TUP\_E})
\end{aligned}
$$

Nun können wir mit diesen Ergebnisrelationen die eigentliche innere Modellverknüpfung durchführen (Für das Ergebnis schreiben wir $\bar{Z}|m_1 \bowtie m_2$):

$$
\begin{aligned}
\bar{Z}|m_1 \bowtie m_2 \quad = \quad & \sigma_{\text{MGL\_E}.[atm\_key]\notin\{a_{1_1},\ldots,a_{1_n}\}}(\bar{Z}|m_1) \cup \\
& \sigma_{\text{MGL\_E}.[atm\_key]\in\{a_{2_1},\ldots,a_{2_n}\}}(\bar{Z}|m_2) \cup \\
& \{(\pi_{\text{MGL\_E}.[atm\_key]}(\sigma_{\text{MGL\_E}.[atm\_key]\in\{a_{1_1},\ldots,a_{1_n}\}}(\bar{Z}|m_1)) \\
& [\text{MGL\_E}.[atm\_key]_{m_1} = a_{1_l} \wedge \text{MGL\_E}.[atm\_key]_{m_2} = a_{2_l}] \\
& \pi_{\text{MGL\_E}.[atm\_key]}(\sigma_{\text{MGL\_E}.[atm\_key]\in\{a_{2_1},\ldots,a_{2_n}\}}(\bar{Z}|m_2)), 1)\}
\end{aligned}
$$

3. Zur Berechnung der Ergebnismodellgleichungen bei einer gekoppelten äußeren Verknüpfung $\bar{m}$ der Modelle m_1 und m_2 mit Auswahlfunktionen $\bar{\sigma}_1$ und $\bar{\sigma}_2$ und Verknüpfungsparametern a_{1_k} und a_{2_k} für $k = 1, \ldots, n$ sind folgende Operationen innerhalb des Relationenschemas erforderlich (Wir bilden analog Punkt 2 die Ergebnisrelationen $\bar{Z}|m_1$ und $\bar{Z}|m_2$; die Funktion $\text{ind}(\bar{\sigma}_r)$ gibt an, ob der Auswahlmechnismus der jeweiligen $\bar{\sigma}_r$ Funktion Gültigkeit besitzt):

$$
\begin{aligned}
\bar{m} \quad : \quad & \{\bar{a}_1 := (\bar{\sigma}_1 \circ \bar{\sigma}_2)(a_{1_1}, a_{2_1}), \ldots, \bar{a}_n := (\bar{\sigma}_1 \circ \bar{\sigma}_2)(a_{1_n}, a_{2_n})\} \\
& \rightarrow (\bar{\sigma}_1 \circ \bar{\sigma}_2)(\bar{Z}|m_1, \bar{Z}|m_2) \\
\bar{Z}|\bar{m} \quad = \quad & \bigcup_{r=1}^{2} \bar{Z}|m_r \cup (\bigcup_{r=1}^{2} \{(\bar{a}_1, (\bar{\sigma}_r \circ \text{ind}(\bar{\sigma}_r))(a_{1_r}, a_{2_r}), 1)\})
\end{aligned}
$$

4. Im Falle einer ungekoppelten äußeren Verknüpfung $\bar{m}$ der Modelle m_1 und m_2 bei Modellparametermengen a_{1_l} und a_{2_k} für $k = 1, \ldots, n_k$ und $l = 1, \ldots, n_l$ und Auswahlfunktionen φ_{m_1} und φ_{m_2} bilden wir wiederum

analog Punkt 2 die Ergebnisrelationen $\bar{Z}|m_1$ und $\bar{Z}|m_2$. Es gilt daher:

$$\bar{m}\colon \{a_{1_1}, \ldots, a_{1_{n_l}}, a_{2_1}, \ldots, a_{2_{n_k}}\} \rightarrow \bigcup_{r=1}^{2} \bar{Z}|m_r$$

5. Ist nun durch eine `SELECT` Klausel ein Modell $\hat{m}$ beschrieben, das aus mehreren Zugriffen untereinander verschachtelter `SELECT` Klauseln besteht, müssen wir diese, bei der größten Verschachtelungstiefe beginnend, gemäß der Punkte 1, 2, 3 und 4 erstellen, sodass dann eine paarweise Berechnung des Ergebnismodells erfolgt.
   Die erstellten Teil- als auch das Ergebnismodell werden, mit einem generischem Namen versehen, in den Entitäten abgelegt.

5.2.3 Die Erstellung von Zielsprachen

Wollen wir Zielsprachen wie MOSIS ([45]), ACSL ([1]) oder MatLab ([2]) aus unseren Relationen erzeugen, müssen wir unsere Tupeloperationen in das dem CSSL Standard entsprechende Grundgerüst einbetten. Welche Teilmenge der Tupel als Modell exportiert wird, ist durch vorhergehende Definition mittels einer `SELECT` Anweisung zu spezifizieren.

Deklarationen: Um das nun erstellte Ergebnismodell $\hat{m}$ in die Zielsprache überzuführen, müssen Deklarationen von Konstanten als auch logischer Variablen zur Verarbeitung der Auswahlfunktionen $\bar{\sigma}_i$ bereitsgestellt werden.

INITIAL Section: Hier verarbeiten wir die `WITH DEFAULT` und `PARAMETERS` Klauseln. Hilfsgößen müssen für Startbedingungen allfällig erforderlicher numerischer Integrationen erzeugt werden.

DYNAMIC Section: Auf die DERIVATIVE Section folgendend werden die für Nachbildung der Auswahlfunktionen erforderlichen „Schaltvorgänge“ eingetragen.

DERIVATIVE Section: Dieser Abschnitt beinhaltet die durch die Tupel erzeugten Modellgleichungen. Bei deren Erstellen müssen wir gemäß der

Definition der Funktion `DRV(.)` ensprechend viele Integratorenaufrufe vorsehen.

TERMINAL Section: Durch Generieren kann nur eine Endbedingung durch Erreichen der Systemvariable `TEND` beschrieben werden.

Die Verarbeitung der Auswahlfunktion $\bar{\sigma}_i$: Für jede $\bar{\sigma}_i$-Bedingung erzeugen wir eine logische Variable und ein der `SCHEDULE` Anweisung (vergleiche [1]) äquivalentes Kommando, das die Auswahlbedingung von $\bar{\sigma}_i$ entsprechend formuliert. Dazu gehörend ist ein `DISCRETE` Block anzugeben, der unter Berücksichtigung der Hilfsfunktion ind($\bar{\sigma}_r$) die unter Abschnitt 5.2.2 Punkt 3 angegebenen Zuweisungen erstellt.

5.3 Die Benutzerschnittstelle

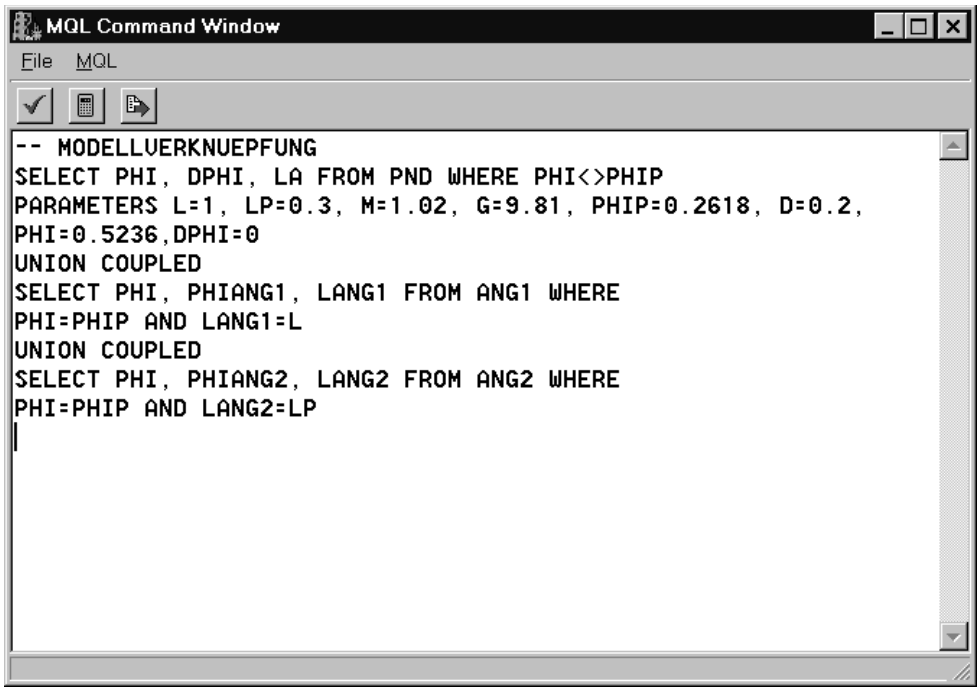


Abbildung 5.1: Das *MQL*-Kommandofenster von *DABEQS*

Das Implementationskonzept *DABEQS* sieht eine Bedienung über ein einziges Kommandofenster vor (vergleiche Abbildung 5.1). Mittels des Menüs *File* kann eine vorhandene *MQL* Befehlsdatei geöffnet oder direkt innerhalb des

Kommandofensters formuliert werden.
Zentrales Bedienelement ist die Buttonleiste. Die Funktionen, die durch Betätigen der Buttons (von links nach rechts) ausgelöst werden, sind

1. Es erfolgt hier nicht die Berechnung des Ergebnismodells, sondern die Durchführung der in Abschnitt 5.2.2 vorgesehenen Prüfungen. Ist das *MQL* Kommando fehlerhaft, so wird darauf durch Ausgabe der entsprechenden Fehlermeldungen hingewiesen.

2. Für rasche Überprüfungen ist eine interpretierende Berechnung eines `SELECT` Kommandos vorgesehen. Des experimentellen Status wegen ist der "Universalintegrator" DASSL ([9]) vorzusehen.

3. Analog zur `EXPORT` Klausel kann ein eben erstelltes Ergebnismodell in die dann auszuwählende Zielsprache generiert werden. Zur Auswahl können Systeme wie ACSL, MOSIS oder MatLab stehen.

Kapitel 6

Einige Anwendungen von *DABEQS*

Nachdem wir die wichtigsten Grundüberlegungen, die zur Herleitung von *DBMR* erforderlich waren, betrachtet haben, wollen wir mittels einiger Beispiele die Anwendung dieses Konzepts vorstellen.

6.1 Das Pendel mit Anschlag

Als erste Anwendung von Verknüpfung von Modellen wollen wir das Pendel mit Anschlag untersuchen. Bevor wir auf die Modellbildung für dieses System eingehen, sehen wir, dass die grundlegende Struktur – zwei Modellkomponenten werden verbunden, das frei schwingende Pendel und das um den Anschlag verkürzt schwingende Pendel – in *DBMR* derart formuliert werden kann, indem mittels des Mechanismus zur äußeren Vereinigung, in diesem Fall der gekoppelten, die Modellkomponenten verbunden werden. Das Schwingen eines Pendels wird durch eine gewöhnliche Differentialgleichung zweiter Ordnung beschrieben.

$$ml\ddot{\varphi} = -mg\sin\varphi - dl\dot{\varphi}$$

φ bezeichnet den Winkel gemessen zwischen der Position des Pendels und der Vertikalen. Die Masse ist mit m, die Länge mit l und der Dämpfungsfaktor mit d bezeichnet.

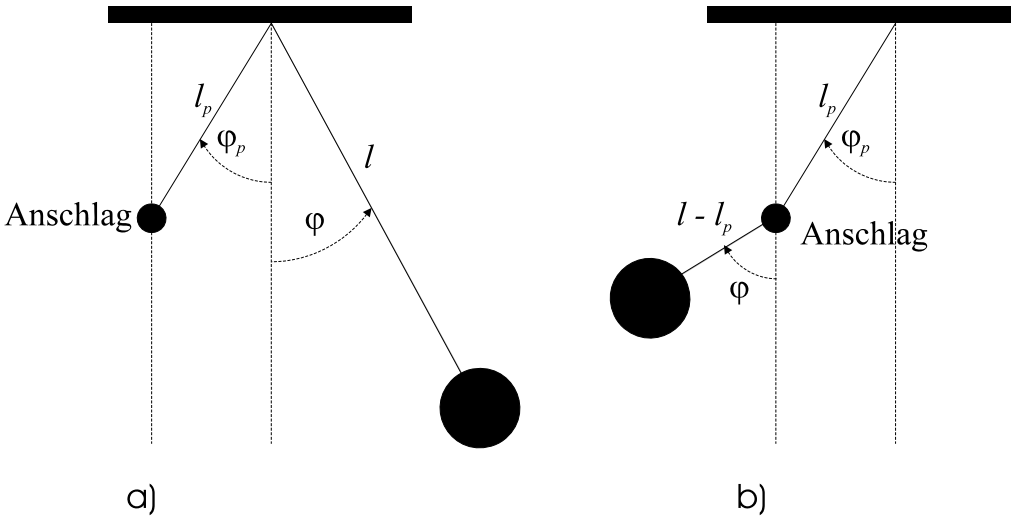


Abbildung 6.1: Das Pendel mit Anschlag – a) freie Bewegung, b) Auslenkung am Anschlag

Schwingt das Pendel, dann berührt es unter geeigneten Konstellationen den Anschlag, der mit Winkel φ_p und Radius l_p vom Aufhängungspunkt des Pendels gelegen ist. In diesem Fall ist der Anschlag das Zentrum der Pendelbewegung mit Länge $l_s = l - l_p$. Ebenso verändert sich die Winkelgeschwindigkeit $\dot{\varphi}$. An der Position φ_p verändert sich die Winkelgeschwindigkeit von $\dot{\varphi}$ auf $\dot{\varphi}\frac{l}{l_s}$. Die Gültigkeit der systembeschreibenden Gleichungen bleibt gegeben. Schwingt das Pendel zurück, dann verkleinert sich der Auslenkungswinkel kontinuierlich, sodass dieser kleiner als φ_p wird. Dann verhält sich das Pendel analog der Ausgangssituation mit Länge l. Somit ändert sich die Winkelgeschwindigkeit an φ_p von $\dot{\varphi}$ auf $\dot{\varphi}\frac{l_s}{l}$ und so weiter.
Wollen wir auf dieses Modell die Methoden von *DBMR* anwenden, dann müssen wir es in mit diesem Konzept verträgliche Elemente „unterteilen“. Die in dieser Gleichung angwandten Elemente atomarer Mengen sind

Die Zeit t: Diese Größe erscheint der Beschreibung kein einziges Mal explizit auf – sie ist aber trotzdem eine fundamentale Variable, da aufgrund der Übereinkunft für derartige Differentialgleichungssysteme die Zeit als freie Variable t anzunehmen ist. *DBMR* verzichtet auf die spezielle Auszeichnung dieser Variable, die daher nun speziell in unsere Tupel aufgenommen werden muss.

Die Winkelgeschwindigkeit φ: Ebenso stellt diese Zustandsfunktion eine atomare Größe dar.

Die Parameter der Differentialgleichung $\{m, g, d, l, l_p, \varphi_p\}$: Diese Elemente können eine spezielle Auszeichnung erfahren, indem wir sie in die Modellparametermenge aufnehmen. Aufgrund dieses Vorgehens kann auch die spezielle Auswählbarkeit durch eine Auswahlfunktion φ_m ermöglicht werden.

Aufgrund der systembeschreibenden Gleichung können wir nun die Tupel der jeweiligen Entitäten anschreiben. Unser Modell sei mit *pnd* bezeichnet.

$$\begin{aligned} atm_{pnd} &= (\varphi, \dot{\varphi}, m, g, d, l_a, l, l_p, \varphi_p) \\ mdl_{pnd} &= (pnd, [atm\_key]_{pnd}) \\ atm \bowtie mgl_{pnd} &= \{(\dot{\varphi}, \mathrm{drv}(1), m, g, *, -, \varphi, \sin, *, d, l_a, *, \dot{\varphi}, *, -, m, l_a, *), \\ &\quad (\varphi, \mathrm{drv}(1), \dot{\varphi})\} \end{aligned}$$

Das Anstossen am Anschlag φ_p müssen wir durch ein eigenes Modell formulieren. Hier erfolgt die augenblickliche „Längenveränderung“ des Pendels und das Ändern der Winkelgeschwindigkeit $\dot{\varphi}$. Die Tupel dieses Modells, das wir mit ang_1 bezeichnen wollen, haben folgende Gestalt:

$$\begin{aligned} atm_{ang_1} &= (\varphi, \varphi_{ang_1}, l_{ang_1}, l, l_p) \\ mdl_{ang_1} &= (ang_1, [atm\_key]_{ang_1}) \\ atm \bowtie mgl_{ang_1} &= \{(\varphi_{ang_1}, \varphi_{ang_1}, l, l, l_p, -, /), (l_{ang_1}, l, l_p, -)\} \end{aligned}$$

Bei der Modellformulierung haben wir ebenso den Fall zu berücksichtigen, bei dem das Pendel den Anschlag als Drehpunkt wieder verlässt und die freie Schwindung mit der gesamten Länge des Pendels erfolgt. Dies beschreibt das Modell ang_2.

$$\begin{aligned} atm_{ang_2} &= (\varphi, \varphi_{ang_2}, l_{ang_2}, l, l_p) \\ mdl_{ang_2} &= (ang_2, [atm\_key]_{ang_2}) \\ atm \bowtie mgl_{ang_2} &= \{(\varphi_{ang_2}, \varphi_{ang_2}, l, l_p, -, l, /), (l_{ang_2}, l)\} \end{aligned}$$

Wir verbinden diese Modelle nun mit dem Mechanismus der gekoppelten äußeren Verknüpfung. Um die Verbindung gewissermaßen über sämtliche relevanten Größen des Modells durchführen zu können, müssen wir einen „Durchschleusungsparameter“ φ in die Modelle ang_i aufnehmen. Das Ergebnismodell

$$\widetilde{pnd}(\tilde{\varphi}, \tilde{\dot{\varphi}}, \tilde{l})$$

baut daher auf der Selektionsfunktion σ auf:

$$\begin{aligned} \sigma: \quad & \tilde{l} = l \wedge \tilde{\varphi} = \varphi_p \rightarrow ang_1 \\ & \tilde{l} = l_p \wedge \tilde{\varphi} = \varphi_p \rightarrow ang_2 \\ & \varphi \neq \varphi_p \rightarrow pnd \end{aligned}$$

Wir verwenden nun *MQL*, um dieses Modell zu formulieren:

```
--  MODELL PENDEL
CREATE MODEL PND (
  DPHI STATE, PHI STATE,
  M FLOAT,    G FLOAT,   D FLOAT,    LA FLOAT,   L FLOAT,
  LP FLOAT,  PHIP FLOAT,
  EQU ( DRV(PHI)  = DPHI,
        DRV(DPHI) = (- M * G * SIN(PHI) - D * LA * DPHI)
                    / (M * LA)),
  INP(LA, DPHI, PHI), OUTP(DPHI, PHI));
--  MODELL ANSCHLAG 1
CREATE MODEL ANG1 (
   PHI STATE,   PHIANG1 STATE,
   LANG1 FLOAT, L FLOAT,       LP FLOAT,
   EQU ( PHIANG1 = PHIANG1 * L / (L - LP),
         LANG1  = L - LP),
   INP(PHI, PHIANG1, LANG1), OUTP(PHI, PHIANG1, LANG1));
--  MODELL ANSCHLAG 2
CREATE MODEL ANG2 (
```

```
   PHI     STATE, PHIANG2 STATE,
   LANG2 FLOAT,   L FLOAT,       LP FLOAT,
   EQU ( PHIANG2 = PHIANG2 * (L - LP) / L,
         LANG2  = L),
   INP(PHI, PHIANG2, LANG2), OUTP(PHI, PHIANG2, LANG2));
-- MODELLVERKNUEPFUNG
SELECT PHI, DPHI, LA FROM PND WHERE PHI<>PHIP
PARAMETERS L=1, LP=0.3, M=1.02, G=9.81, PHIP=0.2618, D=0.2,
PHI=0.5236,DPHI=0
UNION COUPLED
SELECT PHI, PHIANG1, LANG1 FROM ANG1 WHERE
PHI=PHIP AND LANG1=L
UNION COUPLED
SELECT PHI, PHIANG2, LANG2 FROM ANG2 WHERE
PHI=PHIP AND LANG2=LP
```

6.2 Bondgraphen

Eine klassische Anwendung der hierarchischen Modellbildung sind Bondgraphen. Diese bestehen, wie wir es im Kapitel 2 dargelegt haben, aus einer Vernetzung spezieller Elemente. Wir werden nun diese Elemente in *DBMR* formulieren und zur Verschaltung wiederum die Mechanismen zur Kopplung von Modellen verwenden. Da in der Verwendung von Bondgraphen bei der Formulierung auf die Kausalität zu achten ist, muss bei diesen Teilmodellen auf eine variable Gleichungsformulierung geachtet werden. *DBMR* stellt dazu den die Auswahlfunktion φ_m zu Verfügung.

Die wichtigsten Elemente und Verknüpfungen von Bondgraphen seien nun in der Tupeldarstellung angeführt, wobei wir aufgrund der Gültigkeit der Kirchhoff'schen Gesetze mehrere Ausprägungen der Auswahlfunktion φ_{m_i} vorsehen müssen. Durch Hintereinanderausführen dieser „Teilfunktionen“ erhalten wir, wie bereits im Kapitel über die Grundlagen ausgeführt, die für die gegebene Situation geeignete Auswahlfunktion.

Spannungsquelle SE:

$$\begin{aligned} atm_{SE} &= (e, f, u_0) \\ mdl_{SE} &= (SE, [atm\_key]_{SE}) \\ atm \bowtie mgl_{SE} &= \{(e, u_0)\} \end{aligned}$$

Bondgraphsymbol $bond$:

$$\begin{aligned} atm_{bond} &= (e_1, f_1, e_2, f_2) \\ mdl_{bond} &= (bond, [atm\_key]_{bond}) \\ atm \bowtie mgl_{bond}|\varphi^1_{bond} &= \{(e_1, e_2), (f_1, f_2)\} \\ atm \bowtie mgl_{bond}|\varphi^2_{bond} &= \{(e_2, e_1), (f_2, f_1)\} \end{aligned}$$

Widerstand R:

$$\begin{aligned} atm_R &= (e, f, r) \\ mdl_R &= (R, [atm\_key]_R) \\ atm \bowtie mgl_R|\varphi^1_R &= \{(e, f, r, *)\} \\ atm \bowtie mgl_R|\varphi^2_R &= \{(f, e, r, /)\} \end{aligned}$$

Kapazität C:

$$\begin{aligned} atm_C &= (e, f, c) \\ mdl_C &= (C, [atm\_key]_C) \\ atm \bowtie mgl_C|\varphi^1_C &= \{(e, \mathrm{drv}(1), f, 1, c, /, *)\} \\ atm \bowtie mgl_C|\varphi^2_C &= \{(f, c, e, \mathrm{drv}(1), *)\} \end{aligned}$$

Induktivität I:

$$atm_I = (e, f, i)$$

$$
\begin{aligned}
mdl_I &= (I, [atm\_key]_I) \\
atm \bowtie mgl_I|\varphi_I^1 &= \{(f, \mathrm{drv}(1), e, 1, i, /, *)\} \\
atm \bowtie mgl_I|\varphi_I^2 &= \{(e, i, f, \mathrm{drv}(1), *)\}
\end{aligned}
$$

Tranformator TF:

$$
\begin{aligned}
atm_{TF} &= (e_1, f_1, e_2, f_2, m) \\
mdl_{TF} &= (TF, [atm\_key]_{TF}) \\
atm \bowtie mgl_{TF}|\varphi_{TF}^1 &= \{(e_1, m, e_2, *)\} \\
atm \bowtie mgl_{TF}|\varphi_{TF}^2 &= \{(f_2, m, f_1, *)\} \\
atm \bowtie mgl_{TF}|\varphi_{TF}^3 &= \{(e_2, 1, m, /, e_1, *)\} \\
atm \bowtie mgl_{TF}|\varphi_{TF}^4 &= \{(f_1, 1, m, /, f_2, *)\}
\end{aligned}
$$

Gyrator GY:

$$
\begin{aligned}
atm_{GY} &= (e_1, f_1, e_2, f_2, r) \\
mdl_{GY} &= (GY, [atm\_key]_{GY}) \\
atm \bowtie mgl_{GY}|\varphi_{GY}^1 &= \{(e_1, r, f_2, *)\} \\
atm \bowtie mgl_{GY}|\varphi_{GY}^2 &= \{(e_2, r, f_1, *)\} \\
atm \bowtie mgl_{GY}|\varphi_{GY}^3 &= \{(f_2, 1, r, /, e_1, *)\} \\
atm \bowtie mgl_{GY}|\varphi_{GY}^4 &= \{(f_1, 1, r, /, e_2, *)\}
\end{aligned}
$$

0-Verbindung J_0

$$
\begin{aligned}
atm_{J_0} &= (e_1, f_1, e_2, f_2, e_3, f_3) \\
mdl_{J_0} &= (TF, [atm\_key]_{J_0}) \\
atm \bowtie mgl_{J_0}|\varphi_{J_0}^1 &= \{(e_2, e_1), (e_3, e_1), (f_3, f_1, f_2, +, -)\} \\
atm \bowtie mgl_{J_0}|\varphi_{J_0}^2 &= \{(e_1, e_2), (e_3, e_2), (f_3, f_1, f_2, +, -)\} \\
atm \bowtie mgl_{J_0}|\varphi_{J_0}^3 &= \{(e_1, e_3), (e_2, e_3), (f_3, f_1, f_2, +, -)\} \\
atm \bowtie mgl_{J_0}|\varphi_{J_0}^4 &= \{(e_2, e_1), (e_3, e_1), (f_2, f_1, f_3, +, -)\}
\end{aligned}
$$

$$
\begin{aligned}
atm \bowtie mgl_{J_0}|\varphi^5_{J_0} &= \{(e_1, e_2), (e_3, e_2), (f_2, f_1, f_3, +, -)\} \\
atm \bowtie mgl_{J_0}|\varphi^6_{J_0} &= \{(e_1, e_3), (e_2, e_3), (f_2, f_1, f_3, +, -)\} \\
atm \bowtie mgl_{J_0}|\varphi^7_{J_0} &= \{(e_2, e_1), (e_3, e_1), (f_1, f_2, f_3, +, -)\} \\
atm \bowtie mgl_{J_0}|\varphi^8_{J_0} &= \{(e_1, e_2), (e_3, e_2), (f_1, f_2, f_3, +, -)\} \\
atm \bowtie mgl_{J_0}|\varphi^9_{J_0} &= \{(e_1, e_3), (e_2, e_3), (f_1, f_2, f_3, +, -)\}
\end{aligned}
$$

1-Verbindung J_1

$$
\begin{aligned}
atm_{J_1} &= (f_1, e_1, f_2, e_2, f_3, e_3) \\
mdl_{J_1} &= (TF, [atm\_key]_{J_1}) \\
atm \bowtie mgl_{J_1}|\varphi^1_{J_1} &= \{(f_2, f_1), (f_3, f_1), (e_3, e_1, e_2, +, -)\} \\
atm \bowtie mgl_{J_1}|\varphi^2_{J_1} &= \{(f_1, f_2), (f_3, f_2), (e_3, e_1, e_2, +, -)\} \\
atm \bowtie mgl_{J_1}|\varphi^3_{J_1} &= \{(f_1, f_3), (f_2, f_3), (e_3, e_1, e_2, +, -)\} \\
atm \bowtie mgl_{J_1}|\varphi^4_{J_1} &= \{(f_2, f_1), (f_3, f_1), (e_2, e_1, e_3, +, -)\} \\
atm \bowtie mgl_{J_1}|\varphi^5_{J_1} &= \{(f_1, f_2), (f_3, f_2), (e_2, e_1, e_3, +, -)\} \\
atm \bowtie mgl_{J_1}|\varphi^6_{J_1} &= \{(f_1, f_3), (f_2, f_3), (e_2, e_1, e_3, +, -)\} \\
atm \bowtie mgl_{J_1}|\varphi^7_{J_1} &= \{(f_2, f_1), (f_3, f_1), (e_1, e_2, e_3, +, -)\} \\
atm \bowtie mgl_{J_1}|\varphi^8_{J_1} &= \{(f_1, f_2), (f_3, f_2), (e_1, e_2, e_3, +, -)\} \\
atm \bowtie mgl_{J_1}|\varphi^9_{J_1} &= \{(f_1, f_3), (f_2, f_3), (e_1, e_2, e_3, +, -)\}
\end{aligned}
$$

Mit diesem Rüstzeug wollen wir ein Beispiel aus der Elektrotechnik formulieren. Abbildung 6.2 stellt einen einfachen Schaltkreis dar, in dem zwei Elemente zur Energiespeicherung vorkommen (Der Kondensator C und die Spule L). Wir nehmen aufgrund dessen an, dass zwei Zustandsgleichungen das Modell bilden werden. Die herkömmlichen Methoden diese Gleichungen zu finden sind die Netz- beziehungsweise die Knotengleichungen. Mit verschiedenen Schritten unter Beachtung allfällig vorhandener algebraischer Schleifen werden diese Gleichungen abgeleitet. Führen wir diesen Schaltkreis in die topologisch äquivalente

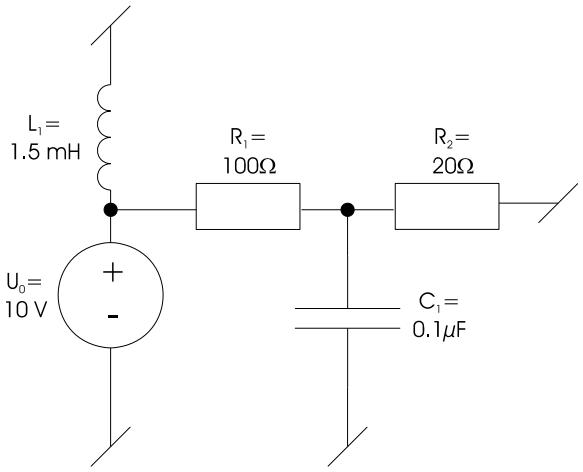


Abbildung 6.2: Diagramm des Schaltkreises

Bondgraphenstruktur über, erhalten wir die Darstellung laut Abbildung 6.3. Wir schreiben nun die analoge Verbindung von *DBMR* Teilmodellen zu einem

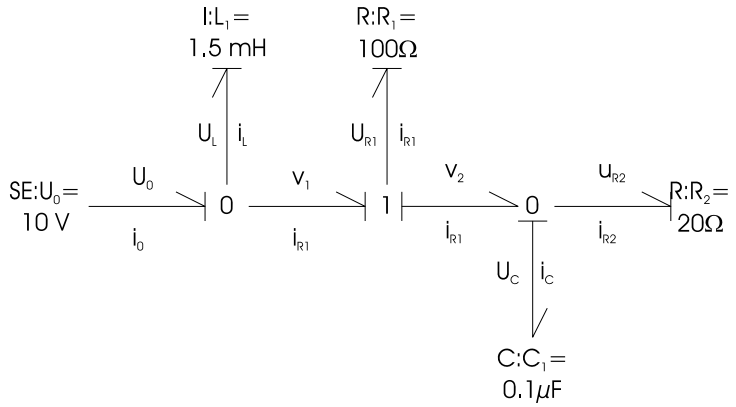


Abbildung 6.3: Bondgraph mit bereits eingezeichneten Kausalitäten

Modell des gesamten Systems an.

Teilmodell	Typ (wie oben beschreiben)
$se(U_0, i_0)$	SE
$j_0^1(U_0, i_0, u_L, i_L, v_1, i_{R1})$, $j_0^2(v_2, i_{R1}, i_C, u_C, u_{R2}, i_{R2})$	J_0
$j_1^1(v_1, i_{R1}, u_{R1}, i_{R1}, v_2, i_{R1})$	J_1
$r_1(u_{R1}, i_{R1}, r_1)$, $r_2(u_{R2}, i_{R1}, r_2)$	R
$l_1(u_L, i_L, l_1)$	I
$c_1(u_C, i_C, c_1)$	C

Durch die Anwendung der inneren Verknüpfung erfolgt die endgültige Beschrei-

bung des Modells rlc:

$$
\begin{aligned}
atm_{rlc} &= (u_0, r_1, r_2, c_1, l_1) \\
atm \bowtie mgl_{rlc} &= (se \bowtie_{(se.f, j_0^1.e)=(j_0^1.f_1, se.e)} j_0^1 \cup \\
&\quad (l_1 \bowtie_{(l_1.e, j_0^1)=(j_0^1.e_2, l_1.f)} j_0^1 \cup (j_1^0 \bowtie_{(j_0^1.f_3, j_1^1.e_1)=(j_1^1.f_1, j_0^1.e_3)} j_1^1) \cup \\
&\quad (r_1 \bowtie_{(r_1.f, j_1^1.e_2)=(j_1^1.f_2, r_1.e)} j_1^1) \cup (j_0^2 \bowtie_{(j_0^2.e_1, j_1^1.f_3)=(j_1^1.e_3, j_0^2.f_1)} j_1^1) \cup \\
&\quad (j_0^2 \bowtie_{(j_0^2.f_3, r_2.e)=(r_2.f, j_0^2.e_3)} r_2 \cup c_1 \bowtie_{(j_0^2.f_2, c_1.e)=(c_1.f, j_0^2.e_2)} j_0^2))
\end{aligned}
$$

Wir verzichten auf eine Anschreibung der *MQL*-Modelldefinitionen, führen aber zur Verdeutlichung der Durchführung einer inneren Modellverknüpfung deren *MQL*-Statement an:

```
SELECT C1.E, R2.F
FROM SE AS SE, R AS R1, R AS R2, C AS C1, I AS L1,
     J0 AS J01, J0 AS J02, J1 AS J11
WHERE
     J01.E1 = U0.E AND L1.E = J01.E2 AND J11.E1 = J01.E3 AND -- SE, J01
     J01.F1 = U0.F AND J01.F2 = L1.F AND J11.F1 = J01.F3 AND -- L1
     R1.F = J11.F2 AND J02.F1 = J11.F3 AND                   -- J11
     J11.E2 = R1.E AND
     J02.E1 = J11.E3 AND C1.E = J02.E2 AND R2.E = J03.E3 AND -- J02
     J02.F2 = C1.F AND J02.F3 = R2.F                         -- C1, R2
PARAMETERS SE.U0 = 10, R1.R=100, R2.R=20, C1.C1=15, L1.I=2
```

6.3 Parametervariation

In der Ökologie stellen Untersuchungen von Populationen, die in einer Wechselbeziehung stehen, ein wichtiges Untersuchungsgebiet dar. In diesem Gebiet erfolgt die Modellbildung durch die Anwendung der Volterra Gleichungen. Einen äquivalenten Ansatz stellt das Konzept des begrenzten Wachstums dar. Verallgemeinert können dieses Problemstellungen durch folgende systembeschrei-

bende Gleichungen charakterisiert werden:

$$\begin{aligned} \dot{x} &= ax(1 - cx - ey) \\ \dot{y} &= by(1 - dy - fx) \end{aligned}$$

Die Klammer drückt den Einfluss der Ressourcen oder der anderen Spezies auf die Fruchtbarkeit der betrachteten Spezies aus. Wir betrachten folgende Vorzeichenkombinationen.

a	b	c	d	e	f	Modell
$+$	$-$	0	0	$+$	$+$	Einfache Raubtier-Beute-Modelle (*Lotka-Volterra*)
$+$	$-$	$+$	0	$+$	$+$	Verbessertes Raubtier-Beute-Modelle
$+$	$+$	$+$	$+$	$-$	$-$	Symbiose
$+$	$+$	$+$	$+$	$+$	$+$	Wettbewerb, allgemein
$+$	$+$	$+$	$+$	$+$	$+$	$c = f$,$d = e$ Wettbewerb um identische Ressourcen

Die Stabilitätsverhältnisse werden durch die Eigenwerte der Systemmatrix beschrieben.

$$\begin{pmatrix} a(1 - 2cx - ey) & -aex \\ -bfy & b(1 - 2dy - fx) \end{pmatrix}$$

Betrachten wir das Lotka-Volterra-Modell, so sehen wir, dass bei der Anwendung dieses Modell zur Validierung viele Parameterstudien erforderlich sein werden. Um diese möglichst effizient vorzusehen, nutzen wir den Mechanismus der ungekoppelten Vereinigung von Modellen. Wir haben bei deren Definition nicht eine paarweise Verschiedenheit der beteiligten Modelle gefordert, daher ist es möglich beliebig viele „Kopien“ desselben Modells unter verschiedenen Konstellationen zu vereinigen. Wir bezeichnen nun das Modell mit lvm und schreiben die erforderlichen Tupel an.

$$\begin{aligned} atm_{lvm} &= (x, y, a, b, c, d, e, f) \\ mdl_{lvm} &= (lvm, [atm\_key]_{lvm}) \\ atm \bowtie mgl_{lvm} &= \{(x, \mathrm{drv}(1), a, x, *, 1, c, x, *, -, e, y, *, -, *), \\ &\qquad (y, \mathrm{drv}(1), b, y, *, 1, d, y, *, -, f, x, *, -, *)\} \end{aligned}$$

Wir bezeichnen nun mit lvm_i mit $i = 1, \ldots, n$ n Kopien des Modells. Daher sind auch die jeweiligen Elemente der Parametermengen mit $x_i, y_i, a_i, \ldots$ zu bezeichnen. $\overline{lvm}$ genügt als ungekoppelte Vereinigung dieser Modelle folgender Bildungsvorschrift:

$$\overline{lvm}: \bigcup_{i=1}^{n} P_i \rightarrow \bigcup_{i=1}^{n} \bar{Z}_{lvm_i}$$

Wir nutzen diesen Mechanismus bei der Variation eines Parameters; dieser sei a. Dann werden bei dem jeweiligen „Komponentenmodell“ im Zuge der Simulation (also der Anwendung der von uns definierten Funktion γ) die jeweiligen Werte zugewiesen. Betrachten wir diesen Sachverhalt im Sinne der Teilung in *model frame* und *experimental frame* (vergleiche [52]), dann sehen wir, dass das *Experiment* der Parametervariation nun Teil des *Modells* ist.
Wir schreiben nun als Beispiel der Durchführung einer ungekoppelten Vereinigung deren *MQL*-Statement an:

```
SELECT X AS X1, Y AS Y1 FROM LVM AS LVM1
   PARAMETERS A=<Wert a(1)>, B=<Wert b(2)>
UNION UNCOUPLED
SELECT X AS X2, Y AS Y2 FROM LVM AS LVM2 PARAMETERS
   A=<Wert a(2)>, B=<Wert b(2)>
UNION UNCOUPLED
-- weitere SELECT Statements fuer i=3, ..., n-1
SELECT X AS XN, Y AS YN FROM LVM AS LVMN PARAMETERS
   A=<Wert a(n)>, B=<Wert b(n)>
```

6.4 Das Doppelpendel

Als weiteres Beispiel betrachten wir ein geregeltes System. Abbildung 6.4 zeigt ein stehendes Doppelpendel, das zur Stabilisierung geregelt sein soll. Wir beschreiben das System derart, dass eine Masse m_m reibungsfrei auf einer horizontalen Ebene gleitet. Über reibungsfreie Gelenke verbinden wir zwei Stäbe (m_1, I_1, l_1) und (m_2, I_2, l_2) mit der Masse m_m. Da wir fordern, dass diese Stäbe in der vertikalen Position gehalten werden, muss zur Stabilisierung des Systems eine geeignete Kraft F als von aussen eingeprägte Kraft auf die Masse einwir-

ken.

Zur Formulierung der Bewegungsgleichungen des mechanischen Systems be-

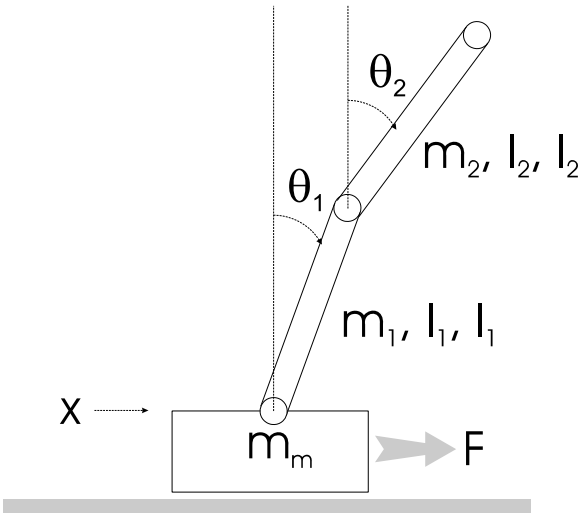


Abbildung 6.4: Mechanisches Modell eines stehenden Doppelpendels

trachten wir die drei Freiheitsgrade x (Bewegung von m_m), θ_1 (Winkel von Stab 1) und θ_2 (Winkel von Stab 2). Als Ergebnis erhalten wir ein System von drei nichtlinearen Differentialgleichungen in der Form

$$M \begin{pmatrix} \ddot{x} \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} = \vec{b}$$

Mit M bezeichnen wir die symmetrische Massenmatrix

$$M(\theta_1, \theta_2) = \begin{pmatrix} m_m + m_1 + m_2 & (\frac{m_1}{2} + m_2) l_1 \cos\theta_1 & \frac{m_2}{2} l_2 \cos\theta_2 \\ (\frac{m_1}{2} + m_2) l_1 \cos\theta_1 & (\frac{m_1}{3} + m_2) l_1^2 & \frac{m_2}{2} l_1 l_2 \cos(\theta_2 - \theta_1) \\ \frac{m_2}{2} l_2 \cos\theta_2 & \frac{m_2}{2} l_1 l_2 \cos(\theta_2 - \theta_1) & \frac{m_2}{3} l_2^2 \end{pmatrix}$$

mit $\vec{b}$ die rechte Seite

$$\vec{b} = \begin{pmatrix} F + (\frac{m_1}{2} + m_2) l_1 \sin\theta_1 \dot{\theta}_1^2 + \frac{m_2}{2} l_2 \sin\theta_2 \dot{\theta}_2^2 \\ \frac{m_2}{2} l_1 l_2 \sin(\theta_2 - \theta_1) \dot{\theta}_2^2 (\frac{m_1}{2} + m_2) g l_1 \sin\theta_1 \\ \frac{m_2}{2} l_2 (g \sin\theta_2 - l_1 \sin(\theta_2 - \theta_1) \dot{\theta}_2^2) \end{pmatrix}$$

Wir können nun den Zustandsvektor des Systems anschreiben

$$\vec{x} = (x, \dot{x}, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2)^T$$

Wir modellieren den Regler, der erforderlich ist, die Masse von x_0 nach x_c zu bewegen, als Kraft F in der Form

$$F = -k_x e + k_{\dot{x}}\dot{x} + k_{i_n} i_n + k_{\theta_1}\theta_1 + k_{\theta_2}\theta_2 + k_{\dot{\theta}_1}\dot{\theta}_1 + k_{\dot{\theta}_2}\dot{\theta}_2$$

wobei

$$e = x_c - c, \qquad \dot{e} = -\dot{x}, \qquad i_n = \int_{t_0}^{t_{end}} e dt$$

gilt.

Mit diesen Grundlagen wollen wir die Tupeldarstellung des Modells herleiten. Die Parameter des Reglers nehmen wir als bereits von den weiteren Parametern abhängig berechnet an. Aufgrund der Anwendung eines Regelungsmechanismus liegt es nahe, diesen separat zu modellieren und mittels der inneren Modellverknüpfung zu verbinden. Wir schreiben (gekürzt) die Tupel, wobei wir in weiterer Folge unter k_α die Gesamtheit der Parameter des Reglers verstehen, an:

$$\begin{aligned}
atm_{dpnd} &= (\theta_1, \dot{\theta}_1, \ddot{\theta}_1, \theta_2, \dot{\theta}_2, \ddot{\theta}_2, x, \dot{x}, \ddot{x}, \\
&\quad m, m_m, m_1, m_2, l_1, l_2, I_1, I_2, F, d, b) \\
mdl_{dpnd} &= (dpnd, [atm_key]_{dpnd}) \\
atm \bowtie mgl_{dpnd} &= \{(m, \dim(1,1), m_m, m_1, m_2, +, +), \ldots, (m, \dim(3,3), m_2, 2, /, l_2, l_2, *, *), \\
&\quad (d, \dim(1), \ddot{x}), (d, \dim(2), \ddot{\theta}_1), (d, \dim(3), \ddot{\theta}_2), \\
&\quad (b, \dim(1), F, m_1, 2, /, m_1, +, l_1, *, \theta_1, \dot{\theta}_1, \dot{\theta}_1, *, *, \sin, *, m_2, 2, \\
&\quad /, l_2, \theta_2, \dot{\theta}_2, \dot{\theta}_2, *, *, \sin, *, *, +, +), \ldots, \\
&\quad (m, d, *, b) \\
&\quad (\ddot{x}, d, \dim(1)), (\dot{x}, \mathrm{drv}(1), \ddot{x}), (x, \mathrm{drv}(1), \dot{x}) \\
&\quad (\ddot{\theta}_1, d, \dim(2)), (\dot{\theta}_1, \mathrm{drv}(1), \ddot{\theta}_1), (\theta_1, \mathrm{drv}(1), \dot{\theta}_1) \\
&\quad (\ddot{\theta}_2, d, \dim(2)), (\dot{\theta}_2, \mathrm{drv}(1), \ddot{\theta}_2), (\theta_2, \mathrm{drv}(1), \dot{\theta}_2)\} \\
atm_{rglr} &= (F, x, \dot{x}, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2, k_\alpha)
\end{aligned}$$

$$
\begin{aligned}
mdl_{rglr} &= (rglr, [atm\_key]_{rglr}) \\
atm \bowtie mgl_{rglr} &= \{e, x_c, x, -), (e, \mathrm{drv}(1), \dot{x}, -), (i_n, \mathrm{drv}(1), e) \\
&\quad (F, k_x, e, *, -, k_{\dot{x}}, \dot{x}, *, +, k_{i_n}, i_n, *, +, k_{\theta_1}, \theta_1, *, +, k_{\theta_2}, \theta_2, \\
&\quad *, +, k_{\dot{\theta}_1}, \dot{\theta}_1, *, +, k_{\dot{\theta}_2}, \dot{\theta}_2, *, +), \\
&\quad (b, b, F, 0, 0, \mathrm{vec}(3), +)\}
\end{aligned}
$$

Um das Grundmodell und die Regelung zu verbinden, formulieren wir das Modell *dpnd_rglr* als gekoppelte äußere Vereinigung der Teilmodelle *dpnd* und *rglr* folgendermaßen:

$$
dpnd\_rglr\colon atm_{dpnd} \cup atm_{rglr} \rightarrow \bar{Z}_{dpnd\_rglr} = dpnd \bowtie_{b=b+(F,0,0)^T} rglr
$$

Wollen wir diese innere Modellverknüpfung mittels der Beschreibungssprache *MQL* formulieren, so schreiben wir:

```
SELECT M1.X, M1.THETA1, M1.THETA2, M1.DTHETA1, M2.DTHETA2,
       M1.DDTHETA1, M1.DDTHETA2, M1.B, M2.F
FROM DPND M1, RGLR M2 WHERE
M2.X = M1.X AND M2.THETA1 = M1.THETA1 AND
M1.B = M2.B
PARAMETERS K
KX = <k(x)>, KDX = <k(x')>, KIN = <k(i(n))>, KTHETA1 = <k(theta(1))>,
KTHETA2 = <k(theta(2))>, KDTHETA1 = <k(theta'(1)>,
KDTHETA2 = <k(theta'(2))>
```

Kapitel 7

Referenz zu DABEQS

7.1 Referenz zu MQL

7.1.1 Syntax von MQL in EBNF Form

```
/* MQL ------------------------------------------------------------------*/
MqlStatement            :       CreateStatement
                        |       SelectStatement
                        |       ExportStatement
                        ;
/* CREATE STATEMENT -----------------------------------------------------*/
CreateStatement         :       ModelCreation
/*                        |       MacroCreation                          */
/*                        |       ExperimentCreation                     */
                        ;

ModelCreation           :       CREATE MODEL IDENTIFIER BRACKET_LEFT
                                Declaration COMMA
                                EquBlock InputOutputSpecification
                                BRACKET_RIGHT
                        ;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
Declaration             :       InitDeclarator
                        |       Declaration COMMA InitDeclarator
                        ;

InitDeclarator          :       Declarator Typ
                        |       Declarator Typ WITH DEFAULT Initialisator
                        ;

Declarator              :       IDENTIFIER
                        |       IDENTIFIER
                                BRACKET_LEFT Dimension BRACKET_RIGHT
                        ;

DclIdentifier           :       Declarator
                        |       IDENTIFIER DOT Declarator
                        ;

Dimension               :       DimensionSpec
                        |       Dimension COMMA DimensionSpec
```

```
                        ;

DimensionSpec           :       IDENTIFIER
                        |       NUMBER
                        ;

Initialisator           :       /* leer */
                        |       BRACKET_LEFT InitList BRACKET_RIGHT
                        ;

InitList                :       NUMBER
                        |       InitList COMMA NUMBER
                        ;

Typ                     :       TYP_FLOAT
                        |       TYP_STATE
                        |       TYP_INTEGER
                        ;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
EquBlock                :       EQU BRACKET_LEFT EquList BRACKET_RIGHT
                        ;

EquList                 :       EquExpression
                        |       EquList COMMA EquExpression
                        ;

EquExpression           :       FlxIOCaseStmt
                        |       MathExpression
                        ;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
FlxIOCaseStmt           :       CASE FlxIOWhenList END
                        ;

FlxIOWhenList           :       FlxIOWhenBlock
                        |       FlxIOWhenList FlxIOWhenBlock
                        ;

FlxIOWhenBlock          :       WHEN FlxIOLogExpression THEN EquExpression
                        ;

FlxIOLogExpression      :       FlxIOSpec
                        |       FlxIOLogExpression AND FlxIOSpec
                        ;

FlxIOSpec               :       INP BRACKET_LEFT DclIdentifier BRACKET_RIGHT
                        |       OUTP BRACKET_LEFT DclIdentifier BRACKET_RIGHT
                        ;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
InputOutputSpecification:       /* leer */
                        |       COMMA InputOutputSpec
                        ;

InputOutputSpec         :       InputOutputSpecBlock
                        |       InputOutputSpec COMMA InputOutputSpecBlock
                        ;

InputOutputSpecBlock    :       INP BRACKET_LEFT IdentifierList BRACKET_RIGHT
                        |       OUTP BRACKET_LEFT IdentifierList BRACKET_RIGHT
                        ;

IdentifierList          :       DclIdentifier
                        |       IdentifierList COMMA DclIdentifier
```

```
                        ;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
EquExpression           :       TargetExpression ASSIGN EquCalcExpression
                        ;

TargetExpression        :       DRV BRACKET_LEFT DclIdentifier BRACKET_RIGHT
                        |       DRV BRACKET_LEFT DclIdentifier COMMA
                                    NUMBER BRACKET_RIGHT
                        |       Declarator
                        ;

EquCalcExpression       :       EquCalcExpression PLUS EquCalcTerm
                        |       EquCalcExpression MINUS EquCalcTerm
                        |       EquCalcTerm
                        ;

EquCalcTerm             :       EquCalcTerm TIMES EquCalcFactor
                        |       EquCalcTerm DIV EquCalcFactor
                        |       EquCalcTerm POWER EquCalcFactor
                        |       EquCalcFactor
                        ;

EquCalcFactor           :       FunctionCall
                        |       BRACKET_LEFT EquCalcExpression BRACKET_RIGHT
                        |       DclIdentifier
                        |       NUMBER
                        |       UnaryOperator EquCalcFactor
                        ;

UnaryOperator           :       PLUS
                        |       MINUS
                        ;

FunctionCall            :       FunctionName BRACKET_LEFT EquCalcExpression
                                             BRACKET_RIGHT
                        ;

FunctionName            :       SIN
                        |       COS
                        |       TAN
                        ;

MathExpression          :       EquCalcExpression
                        ;
/* SELECT STATEMENT ----------------------------------------------------*/
SelectStatement         :       FullSelect
                        ;

FullSelect              :       FullSelectBlock
                        |       FullSelect UnionStatement FullSelectBlock
                        ;

FullSelectBlock         :       SubSelect
                        |       BRACKET_LEFT FullSelect BRACKET_RIGHT
                        ;

UnionStatement          :       UNION
                        |       UNION COUPLED
                        |       UNION UNCOUPLED
                        ;

SubSelect               :       SelectClause FromClause
```

```
                        |       SelectClause FromClause WhereClause
                        |       SelectClause FromClause ParmClause
                        |       SelectClause FromClause WhereClause
                                ParmClause
                        ;

SelectClause            :       SELECT SelectList
                        ;

SelectList              :       SelectExpressionBlock
                        |       SelectList COMMA SelectExpressionBlock
                        ;

SelectExpressionBlock   :       DclIdentifier
                        |       DclIdentifier IDENTIFIER
                        |       DclIdentifier AS IDENTIFIER
                        ;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
FromClause              :       FROM    ModelList
                        ;

ModelList               :       ModelSpec
                        |       ModelList COMMA ModelSpec
                        ;

ModelSpec               :       IDENTIFIER
                        |       IDENTIFIER IDENTIFIER
                        |       IDENTIFIER AS IDENTIFIER
                        |       BRACKET_LEFT SubSelect BRACKET_RIGHT
                        |       BRACKET_LEFT SubSelect BRACKET_RIGHT
                                IDENTIFIER
                        |       BRACKET_LEFT SubSelect BRACKET_RIGHT
                                AS IDENTIFIER
                        |       JoinedModel
                        ;

JoinedModel             :       ModelSpec JoinStatement ModelSpec
                                JoinCondition
                        |       BRACKET_LEFT JoinedModel BRACKET_RIGHT
                        ;

JoinStatement           :       MODEL JOIN
                        |       LEFT MODEL JOIN
                        |       RIGHT MODEL JOIN
                        ;

JoinCondition           :       ON JoinConditionList
                        ;

JoinConditionList       :       JoinCondListBlock
                        |       JoinConditionList AND JoinCondListBlock
                        ;

JoinCondListBlock       :       DclIdentifier ASSIGN DclIdentifier
                        ;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
WhereClause             :       WHERE SearchCondition
                        ;

SearchCondition         :       SearchCondBlock
                        |       SearchCondition SearchOp SearchCondBlock
                        ;
```

```
SearchCondBlock         :       Predicate
                        |       NOT Predicate
                        |       BRACKET_LEFT SearchCondition BRACKET_RIGHT
                        |       NOT BRACKET_LEFT SearchCondition BRACKET_RIGHT
                        ;

SearchOp                :       AND
                        |       OR
                        ;

Predicate               :       PredExpression PredOp PredExpression
                        |       DclIdentifier PredJoinOp DclIdentifier
                        ;

PredOp                  :       ASSIGN
                        |       NE
                        |       GT
                        |       GE
                        |       LT
                        |       LE
                        ;

PredJoinOp              :       JOIN_RIGHT
                        |       JOIN_LEFT
                        ;

PredExpression          :       MathExpression
                        ;
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
ParmClause              :       PARAMETERS ParmList
                        ;

ParmList                :       ParmListBlock
                        |       ParmList COMMA ParmListBlock
                        ;

ParmListBlock           :       DclIdentifier ASSIGN MathExpression
                        |       DclIdentifier FROM MathExpression TO
                                MathExpression
                        ;
/* EXPORT STATEMENT ----------------------------------------------------*/
ExportStatement         :       EXPORT  ExportSpec TO IDENTIFIER
                        ;

ExportSpec              :       MOSIS
                        |       ACSL
                        |       MATLAB
                        ;
```

7.1.2 Fehlercodes von MQL

Folgende Aufstellung listet alle vorgesehenen Meldungen des Systems *DABEQS* auf. Wir bezeichnen diese mit n/α, wobei n die Fehlernummer und α den Fehlertyp (Fehler oder Warnung) angibt.

1/F *Model already defined.*
Mittels einer `CREATE` Anweisung wurde ein Modell angeführt, das bereits existiert.

2/F *Variable with incompatible type.*
In der Datenbank der atomaren Mengen wurde bei der Überprüfung der Variablen in einer `CREATE` Anweisung.

3/F *Error in variable declaration.*

4/F *Error in equation declaration.*

5/F *Error in case statement.*

6/F *Error in input/output specification.*
Sämtliche obige Meldungen zeigen auf einen syntaktischen Fehler in den angegeben Sprachelementen.

7/F *INP/OUTP Block not possible with CASE.* Wurde eine `CASE` Klausel im `EQU` Block angegeben zusätzlich auch noch eine `INP/OUTP` Anweisung, so ist dies nicht möglich.

50/W *Input/Output specification not given.*
Bei einer `CREATE` Anweisung wurde keine Spezifikation der ein- und ausgehenden Größen angegeben; bei einer allfälligen Verknüpfung kann es zu fehlerhaften Ergebnismodellen kommen.

51/W *No state variable given.*
In einer `CREATE` Klausel wurde keine Variable des Typs `STATE` angegeben. Eine Integration über den Zeitbereich liefert dann nur konstante Werte.

100/F *Model not found.*

In einer `SELECT` Anweisung wurde auf ein Modell verwiesen, das nicht existiert.

101/F *Unknown variable specified.*

Eine unbekannte Variable wurde angegeben.

102/F *Error in parameter list.*

Ein Syntaxfehler in der `PARAMETERS` Klausel

103/F *Unknown variable in parameter list.*

Eine angegebene Variable existiert nicht in den Modellen, die in der angegebenen `SELECT` Anweisung angesprochen werden, oder ist keine Systemvariable.

104/F *Given value not valid for system variable.*

Einer Systemvariable wird ein laut Wertevorrat nicht zulässiger Wert zugewiesen.

200/F *Incompatible types joined.*

Bei einer inneren Modellverknüpfung wurden Variablen verbunden, deren Kopplung nicht zulässig ist.

201/F *Reference to variable ambiguous.*

In den Modellen, die durch die angegebene `SELECT` Anweisung verknüpft werden, existieren gleichnamige Variablen. Hier ist eine Qualifizierung erforderlich.

250/W *Input/Output variable used as an output/input.*

Die Art einer Variablen wurde bei der Erstellung des Modells anders definiert als nun durch Joining Operatoren angegeben.

300/F *Number of selected variables in* `UNION` *statements not equal.*

Bei der Vorschrift zur Erstellung einer äußeren Verknüpfung von Modellen muss die Anzahl der in der `SELECT` Liste angegebenen Variablen übereinstimmen.

301/F *Incompatible type of variables in* `UNION` *statement.*

Die an der gleichen Position stehenden Variablen müssen gleichen Typs sein.

400/F *Simulation algorithm does not converge.*

Der angewandte Simulationsalgorithmus konvergiert bei der näherungsweisen Berechnung der Jakobi-Matrix oder der Anwendung des Korrektors nicht.

500/F *Selected accuracy not possible - run stopped.*

Die angegebene Genauigkeit konnte nicht erfüllt werden. Es unterbleibt die Berechnung des interpolierten Ergebnis.

550/W *No simulation data - integration algorithm started.*

In der `PARAMETERS` Klausel wurde angegeben, dass die Anwendung von *DBI/DBC* zur Lösung der angegeben Simulationsaufgabe erfolgen soll. In der Experimentsdatenbank (Entität *EXD*) sind aber keine Daten gespeichert. Es erfolgt daher eine direkte Berechnung.

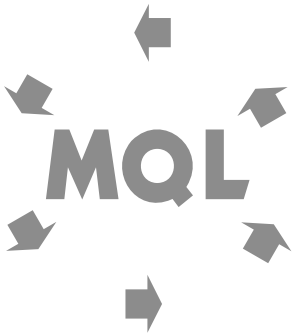
601/F *No preceeding SELECT command. Nothing exported.*

Um ein Modell exportieren zu können, muss vorhergehend durch eine `SELECT` Klausel angegeben werden, welches Modell werden soll.

7.1.3 Systemdaten

Folgende Tabelle gibt die Systemvariablen an, denen innerhalb einer `SELECT` Anweisung mittels der `PARAMETERS` Klausel Werte zugewiesen werden können.

Name	Wertevorrat	Beschreibung
T0	–	Beginnwert der Zeitintegration
TEND	–	Endewert der Zeitintegration
ABSERROR	–	Absolut Fehler für Integrationsverfahren
RELERROR	–	Relativer Fehler für Zeitintegration
NSTEPS	–	Anzahl der Integrationspunkte im Intervall T0 - TEND
DBIDBC	0,1	Wenn = 0, dann erfolgt keine Anwendung von *DBI/DBC*, Wenn = 1, dann erfolgt dessen Anwendung



Literaturverzeichnis

[1] –: *ACSL Reference Manual, Edition 11.* Mitchell & Gauthier Associates Inc., 1995.

[2] –: *Matlab Online Help Version 5.2.* The Mathworks Inc., 1998.

[3] F. Breitenecker: *Comparison 7 - ACSL.* EUROSIM - Simulation News Europe, Nr. 8, Juli 1993, S.30.

[4] F. Breitenecker, H. Ecker, I. Bausch-Gall: *Simulation mit ACSL.* Vieweg, 1993.

[5] F. Breitenecker, D. Solar, I.Husinsky: *Implizite Modelle im Simulationssystem HYBSYS.* Fortschritte in der Simulationstechnik, Tagungsband 6. Symposium September 1990, 332-333.

[6] F. Breitenecker: *Models, Methods and Experiments – a new structure for simulation systems.* Mathematics and Computers in Simulation Bd. 34, 231-260.

[7] F. Breitenecker: *Simulationssprachen für kontinuierliche Systeme.* Skriptum zu Vorlesungen und Übungen. 1989.

[8] F. Breitenecker, H. Ecker, I. Bausch-Gall: *Simulieren mit ACSL.* Forschritte in der Simulationtechnik Band 2, Vieweg 1993.

[9] K.E. Brenan, S.L. Campbell, L.R. Petzold: *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* North Holland 1989.

[10] I.N. Bronstein, K.A. Semendjajew: *Taschenbuch der Mathmatik.* B.G. Teubner, 1991.

[11] F. Cellier: *Continuous System Modeling.* Springer 1991.

[12] P.P.-S. Chen: *The Entity-Relationship Model – Toward a Unified View of Data.* ACM Transactions on Database Systems 1, 9-36, 1970.

[13] E.F. Codd: *A Relational Model of Data for Large Shared Data Banks.* Communications of the ACM 13, 377-387, 1970.

[14] A. Dubois: *Knowledge-Based Component Model Management System For Building Energy Modelling and Simulation.* Proceedings of the 1990 European Simulation Multiconference, 759-763.

[15] H. Elmqvist, D. Brück: *Composite Constructs for Object-oriented Modeling.* Eurosim '95 simulation congress 1995, 99-104.

[16] P. Eschenbacher: *Numerische Integration an Unstetigkeitsstellen in der Modellbeschreibungssprache SIMPLEX-MDL.* Fortschritte in der Simulationstechnik, Tagungsband 6. Symposium September 1990, 339-343.

[17] M. Frank: *Metamodelling of Simulation Tools.* Proceedings Session „Software Tools and Products" Argesim Report No. 2, 1995.

[18] S. Gass, K. Hoffman, R. Jackson, L. Joel, P. Saunders: *Documentation for a Model: a Hierarchical Approach.* Communications of the ACM, Vol. 24/12 1981, 728-734.

[19] N.Gaus: *Kopplung von Dialog und Datenhaltung für die regelungstechnische Simulation.* Fortschritte in der Simulationstechnik, Tagungsband 6. Symposium September 1990, 402-406.

[20] C. Gerthsen: *Physik : ein Lehrbuch zum Gebrauch neben Vorlesungen.* Springer, 1993.

[21] H. Herold: *Lex und Yacc: lexikalische und syntaktische Analyse.* Addison-Wesley, 1992.

[22] P. Heywood, G. MacKechnie, R. Pooley, P. Thanisch: *Object-Oriented Database Technology Applied to Distributed Simulation.* Eurosim '95 simulation congress 1995, 291-296.

[23] M. Koh, J. Seybold, R. Rühle: *Data models and simulation of mechanical systems.* Simulation Practice and Theory 4, 1996, S.319-334.

[24] P.R. Innocent: *Knowledge based simulation as an aid to product testing for usability.* Simulation Practice and Theory 6, 1996, S.383-398.

[25] K-H. Koh, R. de Souza, N.-C.Ho: *Database driven simulation/simulation based scheduling of a job-shop.* Simulation Practice and Theory 4, 1996, S.31-46.

[26] J.D. Lambert: *Numerical Methods for Ordinary Differential Systems.* John Wiley & Sons, 1990.

[27] M. Lingl: *Hybrid Modelling and Simulation.* Seminarunterlagen Matlab Seminar 1999, 173-176.

[28] M. Lingl: *Hybrid Modelling Aproach in Discrete, Continous and Combined Simulation.* in Vorbereitung.

[29] P. Lugner, W. Bub: *Systematik der konzeptionellen Modellbildung.* Fortschritte in der Simulationstechnik, Tagungsband 6. Symposium September 1990, 62-66.

[30] C.H. Luk, M.A. Jette: *Maintenance Simulation: Software Issues.* Eurosim '95 simulation congress 1995, 159-164.

[31] N. Messina, F. Boeri, J. Demartini: *A functional approach for modelling and simulation.* Eurosim '95 simulation congress 1995, 111-116.

[32] Y. Monsef: *Modelling and Simulation of Complex Systems.* Society for Computer Simulation Int., 1997.

[33] G. Engeln-Müllges, F. Reutter: *Formelsammlung zur Numerischen Mathematik mit C-Programmen.* Wissenschaftsverlag, 1987.

[34] T.I. Ören: *Concepts and Criteria to Assess Acceptability of Simulation Studies: A Frame of Reference.* Communications of the ACM, Vol. 24/4 1981, 180-189.

[35] T.I. Ören: *Model-Based Activities: A paradigm shift.* Simulation and Model-Based Methodologies: An integrative view, Springer 1984, 1-40.

[36] T.I. Ören, B.P. Zeigler: *Concepts for Advanced Simulation Methodologies.* Simulation Vol. 32/3 1979, 69-82.

[37] J. Plank: *State Events in Continuous Modelling and Simulation.* ARGESIM/ASIM Verlag, 1997.

[38] G. Pomberger: *Softwaretechnik und Modula-2.* Springer 1984.

[39] H. Praehofer: *Neue Konzepte für die Simulation von kombiniert diskret-kontinuierlichen Systemen – Ein systemtheoretischer Ansatz –.* Fortschritte in der Simulationstechnik, Tagungsband 6. Symposium September 1990, 112-116.

[40] H. Praehofer, G. Reisinger: *Object Oriented Realization of a Parallel Discrete Event Simulator.* Eurosim '95 simulation congress 1995, 327-332.

[41] T. Preiß: *Modellbildung und Integration.* Diplomarbeit, TU-Wien, 1992.

[42] J.W. Rozenblit, B.P. Zeigler: *Entity-based Structures for Modelling and Experimental Frame Construction.* Modelling and Simulation Methodology in Artificial Intelligence Era, North Holland 1986, 79-99.

[43] F.J. Schmitt: *Praxis des Compilerbaus.* Hanser, 1992.

[44] A. Schoisswohl, G. Schuster, F. Breitenecker: *bgm – Bond Graph Modeler.* Poster Book, Argesim Report No. 3, 1995, 81.

[45] G. Schuster: *Definition and Implementation of a Model Interconnection Concept in Continuous Simulation.* Dissertation, TU-Wien, 1994.

[46] C. Standridge: *The Simulation Data Language: Applications and Examples.* Simulation Vol.34 1981, 119-130.

[47] C. Standridge: *Using Simulation Data Language.* Simulation Vol.34 1981, 73-81.

[48] C. Standridge, J.R. Philips: *Using SLAM and SDL to assess space shuttle experiments.* Simulation Vol.36/6 1983, 25-35.

[49] M. Vetter: *Aufbau betrieblicher Informationssysteme mittels objektorientierter, konzeptioneller Datenmodellierung.* B.G. Teubner, 1991.

[50] G. Vossen: *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme.* Addison-Wesley 1987.

[51] N. Wirth: *Programmieren mit Modula-2.* Springer, 1985.

[52] B.P. Zeigler: *Theory of Modelling and Simulation.* John Wiley & sons, 1976.

[53] B.P. Zeigler: *Object-Oriented Simulation with Hierarchical, Modular Models.* Academic Press London, 1990.

[54] B.P. Zeigler, S. Chi, F. Cellier: *Model-Based Architecture for High Autonomy Systems.* AI-Simulation Group University of Arizona, 1991.

[55] B.P. Zeigler: *Multifaceted Modelling and Discrete Event Simulation.* Academic Press London, 1984.

Über den Autor …

Thomas Preiß, geboren 1968 in Wien, studierte 1986 bis 1992 Technische Mathematik, Studienzweig Computerwissenschaften, an der Technischen Universität Wien. Seine Diplomarbeit befasste sich schon mit der Erstellung von formalen Sprachen zur Unterstützung spezieller Gebiete der kontinuierlichen Simulation. Als Mitarbeiter der ARGESIM an der TU Wien (1990 - 1995) vertiefte er seine Kenntnisse in Simulationstechnik und Datenbanken und publizierte Ergebnisse seiner Arbeiten bei ASIM-Tagungen.
Seit 1992 arbeitet Thomas Preiß als Systemanalytiker in der Niederösterreichischen Landesregierung, wo er für große Datenbanksysteme zuständig ist. Wissenschaftlich widmete er sich weiterhin der Untersuchung von Datenbankmangagementsystemen in Verbindung mit kontinuierlicher Simulation. Ergebnisse dieser Arbeit sind in dieser Dissertation zusammengefasst.

Über diesen Band …

In dieser Arbeit soll gezeigt werden, wie relationale Strukturen für kontinuierliche Simulation nutzbar zu machen sind. Die Darstellung erstreckt sich von der Darstellung der grundlegenden Theorien bis zu den Detailanalysen tatsächlicher Implementierungskonzepte. Die dafür erforderlichen Überlegungen bauen zunächst die Theorie eines Tupelkalküls auf, das zur Darstellung von Modellbildung, Verknüpfung von Modellen und deren numerischer Auswertung, der Simulation an sich, geeignet ist. Diese findet ihren Niederschlag in der Definition einer formalen Sprache (Model Query Language), die die Anwendung des Tupelkalküls ermöglicht. Schließlich wird eine Detailanalyse für eine Implementierung unter Berücksichtigung der ausgewählten Menge an Elementen der Sprache MQL angegeben. Den Abschluss bilden praktische Anwendungen, die derart ausgewählt sind, dass sämtliche Aspekte des neuen Konzepts beleuchtet werden können.

Über diese Reihe …

Die Bände dieser neuen ASIM - Reihe Fortschrittsberichte Simulation konzentrieren sich auf neueste Lösungsansätze, Methoden und Anwendungen der Simulationstechnik (Ingenieurwissenschaften, Naturwissenschaften, Medizin, Ökonomie, Ökologie, Soziologie, etc.).
ASIM, die deutschsprachige Simulationsvereinigung (Fachausschuss 4.5 der GI - Gesellschaft für Informatik) hat diese Reihe ins Leben gerufen, um ein rasches und kostengünstiges Publikationsmedium für derartige neue Entwicklungen in der Simulationstechnik anbieten zu können.
Die Fortschrittsberichte Simulation veröffentlichen daher: * Monographien mit speziellem Charakter, wie z. B. Dissertationen und Habilitationen * Berichte zu Workshops (mit referierten Beiträgen) * Berichte von Forschungsprojekten * Handbücher zu Simulationswerkzeugen (User Guides, Vergleiche, Benchmarks), und Ähnliches.
Die Kooperation mit den ARGESIM Reports der ARGESIM vermittelt dabei zum europäischen Umfeld und zur internationalen Publikation.