Aktive DBMS zur Konsistenzsicherung in CAD-Umgebungen

Joachim Reinert
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern

Zusammenfassung

In der Entwicklung von Modellen und sie realisierende „Systeme“ für Entwurfsanwendungen sind in den vergangenen Jahren verschiedene Problembereiche lokalisiert und bearbeitet worden. Dazu zählen komplexe Objektmodelle, die Organisation langer Verarbeitungszyklen, die Notwendigkeit des verteilten Entwurfs und Probleme bei der Beschreibung „gültiger“ Entwurfsobjekte (Konsistenzbegriff). In dieser Arbeit wird das Konzept von *ECA*-Regeln, das in vielen Ansätzen für aktive Datenbanken verwendet wird, bezüglich der Anwendung als Modell zur Sicherung der Konsistenz in solchen Umgebungen analysiert. Die Ergebnisse dieser Analyse zeigen, daß die allgemein üblichen Vorschläge für solche Regelsysteme nicht *direkt* zur Konsistenzsicherung eingesetzt werden können, sondern höchstens „Zielsprache“ eines Übersetzungsprozesses darstellen können, da sonst das System keine *Garantien* für die Erhaltung der Konsistenz übernehmen kann.

1 Einführung

1.1 Problembereiche in CAD-Umgebungen

Die Forschungs- und Entwicklungsanstrengungen in den letzten Jahre zur Entwicklung adäquater Systeme zur Unterstützung von Entwurfsanwendungen haben die zentralen Problembereich solcher Anwendungen offengelegt [HNSB90, RS92]:

- Komplexe Objekte
  Bei der Erfassung der abzubildenden Miniwelt wurde sehr schnell deutlich, daß klassische Informations- und Datenmodelle (z.B. ER-Modell) eine genaue Abbildung nur sehr unvollständig leisten können, da die Objekte in Entwurfsumgebungen heterogen und komplex strukturiert sind. Daher bestand die vordringliche Aufgabe in der Entwicklung neuer Informations- und Datenmodelle zur Erfassung der vorliegenden Objektstrukturen [Sch93].

- Lange Verarbeitungszyklen
  Neben der Modellierungsproblematik der Entwurfsobjekte stellt auch der Verarbeitungsvorgang hohe Anforderungen, da hier lange Verarbeitungszyklen realisiert werden müssen. Stunden, Tage oder auch noch längere Zeiträume sind nicht ungewöhnlich. Das Problem liegt darin, daß klassische Datenhaltungssysteme (z.B. relationale Datenbankmanagementsysteme) bei der Realisierung ihrer Garantien (ACID-Paradigma) von kurzen Verarbeitungsschritten ausgehen. Die dort entwickelten Konzepte (z.B. Isolationsebenen) sind für Entwurfsanwendungen ungeeignet [RMH+94].

- Verteilter Entwurf
  Während die klassischen Datenhaltungssysteme darauf ausgelegt waren, viele Benutzer *gleichzeitig* mit *geringen* Überlappungsbereichen im Bezug auf die Daten zu verwalten, sind es in CAD-Umgebungen vergleichsweise wenige Benutzer, deren Arbeitsbereiche sich bezüglich der Daten jedoch erheblich überlappen. Die Überlappung entsteht vor allem auch dadurch, daß die Objekte, die verarbeitet werden, den oben angeführten sehr langen Verarbeitungszeiten unterliegen. Darüberhinaus kann in Entwurfsumgebungen nicht von einer vollständig zentralisierten Datenhaltung ausgegangen werden, sondern die Entwurfsobjekte sind zu einem Zeitpunkt sowohl innerhalb *einer* organisatorischen Einheit als auch über *verschiedene* organisatorische Einheiten verteilt (diese Verteilung impliziert auch die Verteilung über verschiedene Datenhaltungssysteme).

- Spezifikation und Kontrolle von Integritätsbedingungen[1]
  Eine große Herausforderung bei jeder Modellbildung ist die Sicherstellung der Übereinstimmung zwischen der abgebildeten „Realität“ und dem davon abgeleiteten Modell, insbesondere dann, wenn nach der Modellbildung alle Aktionen innerhalb des Modells durchgeführt werden, denn damit besteht die Gefahr, daß sich „Realität“ und Modell auseinanderentwickeln. Aus diesem Grund sind sprachliche Ausdrucksmittel, die es erlauben, bestimmte Eigenschaften der Objekte des Modells zu spezifizieren, für *jedes* Modell außerordentlich wichtig. Werden solche Modelle durch ein

[1] Wir benutzten in dieser Arbeit die Begriffe der Konsistenz und der Integrität weitgehend synonym.

Datenhaltungssystem realisiert, sind die spezifizierten Integritätsbedingungen (IBs) nur dann von Interesse, wenn durch das System auch Garantien für deren Einhaltung übernommen werden.
Im CAD-Bereich ist die Erfassung solcher Integritätsbedingungen aus folgenden Gründen sehr schwierig:

- Komplexität der Objekte
  Da, wie oben ausgeführt, die Objekte innerhalb des CADs sehr komplex strukturiert sind, ist diese Komplexität auch in den IBs enthalten. Das bedeutet, daß für die Erfassung der IBs die gesamten Formulierungsmöglichkeiten dieser komplexen Datenmodelle zur Verfügung stehen muß. Beispielsweise ist es daher in der Regel erforderlich, die *Rekursion* innerhalb der Definition von Integritätsbedingungen zuzulassen. Dies ist eine ganz erhebliche Erweiterung im Vergleich zum klassischen relationalen Datenmodell.
- Unterschiedliche Ebenen der Konsistenz
  Während klassische Integritätsbedingungen auf dem Zustand des Datenbestandes (Zustandsbedingungen) oder auf höchstens zwei aufeinanderfolgenden Zuständen (Übergangsbedingungen) ausgewertet werden konnten, ist es in CAD-Systemen sinnvoll, den Konsistenzbegriff auch auf die benutzte Entwurfsmethodologie auszudehnen, d.h., der *Entwurfsprozeß* wird neben den *Entwurfsobjekten* Gegenstand von Integritätsbedingungen. Daneben wird versucht, auch das *Entwurfsziel*, d.h. den Gegenstand, der am Ende des Entwurfsprozesses vorliegen soll, mit Hilfe solcher Integritätsbedingungen zu beschreiben. Gerade letzteres macht deutlich, daß vor Abschluß des gesamten Entwurfsprozesses nicht damit gerechnet werden kann, daß die Entwurfsobjekte allen definierten Integritätsbedingungen genügen, sondern daß *bestenfalls* ein monotones Anwachsen der Menge der erfüllten Integritätsbedingungen erwartet werden kann.

Im Rahmen der aufgezeigten Problemschwerpunkte beschäftigt sich die vorliegende Arbeit mit dem Problem der Konsistenzsicherung. Dazu wird untersucht, inwieweit aktive DBMS dazu einen sinnvollen Beitrag leisten können. Solche DBMS werden im folgenden Abschnitt kurz charakterisiert.

1.2 (Re-)aktive Datenbankmanagementsysteme (aDBMS)

In den letzten Jahren war die Entwicklung (re-)aktiver DBMS ein Forschungsschwerpunkt. Das bedeutet, die bisher passiven DBMS, deren Datenbestand sich nur durch explizite Benutzeroperationen änderte, so zu erweitern, daß bestimmte, vorher spezifizierte Situationen erkannt werden und in der Folge eine Reaktion darauf ausgeführt wird. Als Beschreibungsmittel solcher (re-)aktiver Systeme wurden von verschiedenen Autoren sogenannte *E*vent-*C*ondition-*A*ction-Regeln vorgeschlagen [Day88, SHP88, WF90, GJS92, GD93].
Im Event-Teil einer Regel wird das Ereignis beschrieben, auf das reagiert werden soll. Dabei reicht die Bandbreite von einfachen, operational definierten Ereignissen (Einfügen, Ändern, Löschen von Objekten) bis hin zu komplexen, in einer Event-Algebra spezifizierten Ereignissen. Die Problematik der Event-Begriffe im Rahmen von DBMS wurde auch in [Rei94] dargestellt; hier wurde aufgezeigt, daß sie für *klassische* Transaktionssysteme nach dem ACID-Paradigma *nicht* sinnvoll sind. Die für CAD-Umgebungen identifizierte Aufgabe der Verwaltung von langen Verarbeitungszyklen eröffnet jedoch eine andere Ebene, da dabei die ACID-Eigenschaften der Verarbeitung nicht für den gesamten Verarbeitungsprozeß gelten können, sondern bestimmte Ergebnisse früher freigeben werden müssen. Daher ist auf einer höheren Ebene ein Zeitbegriff (der wesentlicher Bestandteil eines Event-Begriffes ist) für diese Umgebung denkbar. Die generelle Problematik einer solchen Verarbeitung wird im folgenden Abschnitt diskutiert.
Der Condition-Teil einer Regel dient zur Beschreibung der Situation, die zum Zeitpunkt des Ereignisses (oder gegebenenfalls zu einem späteren Zeitpunkt) vorliegen muß, damit die Aktionen, die im Action-Teil spezifiziert werden, eine adäquate Reaktion auf das Ereignis darstellen. Die Ausführung solcher *ECA*-Regeln können natürlich weitere Regeln auslösen, so daß auf natürliche Art und Weise eine Vorwärtsverkettung der Regeln entsteht, wie sie bereits aus Produktionsregelsystemen [For81] bekannt ist.
Ein in der Literatur immer wieder vorgeschlagenes Anwendungsgebiet (re-) aktiver DBMS, und damit von *ECA*-Regeln, ist die Sicherung von Integritätsbedingungen. Derartige Überlegungen haben schon eine gewisse Tradition, da schon sehr frühe Vorschläge zur Integritätssicherung in relationalen Systemen sich ähnlicher Ausdrucksmittel bedienten [HM75].
Die allgemeine Einsetzbarkeit von *ECA*-Regeln in Entwurfsumgebungen wird in [RR] diskutiert. In dieser Arbeit gehen wir der speziellen Frage nach, wo der Einsatz von *ECA*-Regeln zur Sicherung der Integrität in CAD-Umgebungen sinnvoll ist bzw. welche Schwierigkeiten bei einem solchen Einsatz zu erwarten sind. Dazu werden wir folgende sehr intuitive Beschreibung der *ECA*-Regeln benutzen:

`Event:` <*Beschreibung des Ereignisses*>
`Condition:`<*Beschreibung der Bedingung*>
`Action:` <*Beschreibung der Reaktion*>

Bei der Beschreibung der einzelnen Elemente werden wir uns in diesem Aufsatz einer intuitiven Sprache bedienen, die sich auf den Sprachmitteln von SQL abstützt.
In einigen Bereichen der Integritätssicherung werden wir nur Teile solcher Regeln benötigen: *EA*-Regeln, in denen nur der Event-Teil und der Action-Teil definiert ist, können sehr einfach für Signalisierungsprozesse verwendet werden, während *CA*-Regeln (nur der Condition-Teil und der Action-Teil sind definiert) den *Zustand* des Datenbestandes überwachen. Bei letzteren gehen wir davon aus, daß implizit alle Events, die die Condition erfüllen können, mit ODER verknüpft werden und den Event-Teil der *ECA*-Regel darstellen.

Die vorliegende Arbeit stellt nun einige Bereiche der Integritätssicherung in CAD-Umgebungen dar und zeigt die Problematiken bei der Lösung mittels *ECA*-Regeln auf. Zunächst wird anhand der Rekursion als fundamentaler Erweiterung in Komplex-Objekt-Datenmodellen und der Problemen der direkten Nutzung des Event-Konzeptes aufgezeigt, daß die Nutzung von *ECA*-Regeln zur direkten Modellierung der Objekte und Prozesse in CAD-Umgebungen nicht geeignet sind. Danach wird die Existenz des bekannten Problems der Mehrdeutigkeit von Regelmengen in CAD-Umgebungen dargestellt und damit auch der Einsatz von *ECA*-Regeln zur Einhaltung bestimmter Verarbeitungsvorgaben in Frage gestellt. Ein weiterer Kritikpunkt an *ECA*-Regeln im Einsatz zur Konsistenzsicherung ist die Tatsache, daß das Konzept keinerlei Garantien für die Einhaltung solcher

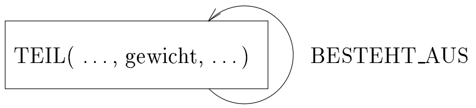
Bedingungen leisten kann. Dies wird in einem weiteren Abschnitt vertieft. Insgesamt zeigt sich, daß ein Einsatz solcher Regeln für die komplexen Aufgaben im CAD-Bereich nicht sinnvoll ist.

2 Datenmodell und Integrität

Im Einführungsabschnitt haben wir bereits darauf hingewiesen, daß in CAD-Umgebungen die Integritätsbedingungen unterschiedlichste Bereiche betreffen können. Bei der Erfassung der Entwurfsobjekte und deren Beziehungen zueinander sowie bei der Beschreibung der Entwurfsmethodologie müssen die *zulässigen* Situationen gegenüber den im gewählten Modellierungssystem *möglichen* abgegrenzt werden können. Modellierungskonstrukte, die diesem Zweck dienen, werden Integritätsbedingungen (IB) genannt. Man kann dabei zwischen modellinhärenten und expliziten IBs unterscheiden: Wird eine Miniwelt mit Hilfe eines Modellierungssystems $\mathcal{A}$ beschrieben, so sind auf Grund der Benutzung von $\mathcal{A}$ automatisch dessen inhärente IBs erfüllt (beispielsweise die Schlüsselbedingung des relationalen Datenmodells). Die Datenmodelle für komplexe Objekte weisen darüberhinaus weitere inhärente IBs auf, so z.B. die referentielle Integrität zwischen Objekten.
Zur Spezifikation expliziter IBs sind separate Konstrukte notwendig, die (nur) den Zweck der Beschreibung zulässiger Situationen haben. In vielen Modellierungssystemen hat sich dazu die eine oder andere Art einer logischen Beschreibung etabliert. Oft findet man dabei eine Prädikatenlogik 1. Stufe. In CAD-Umgebungen wird für die Modellierung der Entwurfsobjekte ein Komplex-Objekt-Datenmodell benötigt. Die Auswirkungen auf die dann zu definierenden IBs werden im folgenden Abschnitt an Hand der Rekursion beispielhaft dargestellt.

2.1 Erfassung rekursiver Integritätsbedingungen

Die komplexe Struktur der Entwurfsobjekte und des Entwurfsprozesses erfordert die Definitionsmöglichkeit von IBs mit rekursiver Struktur. Dies stellt im Vergleich mit den klassischen Formulierungsmöglichkeiten in Logiken 1. Stufe eine wesentliche Erweiterung dar. Bei der Benutzung von Regeln kann nun diese Rekursion entweder *explizit* in die Sprache für die Formulierung des Condition-Teils (also die Anfragesprache des DBMS)



TEIL::
BESTEHT_AUS: SET OF TEIL;
gewicht: REAL;
...

Abbildung 1: Einfaches Schema der Stückliste

eingeführt werden oder *implizit* durch die Folgerungsmöglichkeit der Regelverarbeitung (Vorwärtsverkettung) erreicht werden.

Beispiel: Abbildung 1 stellt die Struktur der bekannten Stückliste dar. Soll nun die IB beschrieben werden, daß kein Teil schwerer als eine bestimmte Grenze g ist, so kann man dies in einer *EA*-Regel, die eine Anfragesprache mit Rekursion benutzt, wie folgt spezifizieren. Dabei ist die Rekursion hier intuitiv mit `RECURSIVE CONNECTED` ausgedrückt:

```
Condition:IF (SELECT *
              FROM TEIL OBT
              WHERE g > TEIL.gewicht +
                         (SELECT SUM(TEIL.gewicht)
                          FROM TEIL UNT
                          WHERE OBT RECURSIVE CONNECTED UNT
                                THROUGH BESTEHT_AUS
             )           )
          THEN
Action:   INFORM ENGINEER
```

Es ist zu beachten, daß die Integritätsbedingung im Condition-Teil *negiert* formuliert wird, d.h., es werden diejenigen Teile ermittelt, die diese Bedingung verletzten.
Soll dieselbe Integritätsbedingung (genauer, ihr *rekursiver* Charakter) mit Hilfe von *ECA*-Regel ausgedrückt werden, so kann dies sinnvollerweise nur dadurch geschehen, daß das Schema des Objektes TEIL um ein Attribut, `gewichtssumme`, erweitert wird und dann folgende Regeln definiert werden:

```
Event:    ON UPDATE TEIL.gewicht
Condition:IF TEIL.gewicht + TEIL.gewichtssumme < g
          THEN
Action:   UPDATE TEIL T
          SET T.gewichtssumme = T.gewichtssumme -
                                OLD.gewicht + NEW.gewicht
          WHERE TEIL = T

Event:    ON UPDATE TEIL.gewichtssumme
Condition:IF TEIL.gewichtssumme < g
          THEN
Action:   UPDATE TEIL OBT
          SET OBT.gewichtssumme = OBT.gewichtssumme -
                                  OLD.gewichtssumme +
                                  NEW.gewichtssumme
          WHERE TEIL IN OBT.BESTEHT_AUS

Event:    ON UPDATE TEIL.gewichtssumme
Condition:IF TEIL.gewichtssumme >= g
          THEN
Action:   INFORM ENGINEER
```

Bei dieser Realisierung wird schon deutlich, daß die Erfassung komplexer Objekte *durch ECA*-Regeln nur schwer möglich ist, da selbst die Beschreibung einfacher Bedingungen erheblichen Aufwand erfordert.

2.2 Erfassung von Ablaufbedingungen

Neben den komplex strukturierten Entwurfsobjekten ist auch die Entwurfsmethodologie in CAD-Umgebungen Gegenstand von IBs. Dabei soll z.B. garantiert werden, daß der Entwurfsprozeß bestimmten Anforderungen genügt, wie sie durch Normen festgelegt sind (ISO 9000). Hierbei scheint das Event-Konzept von *ECA*-Regeln besonders attraktiv, insbesondere wenn eine komplexe Event-Algebra zur Definition der erwarteten (oder notwendigen) Ereignisse vorgesehen wird. Allerdings erweist sich die Einbindung solcher *ECA*-Regeln in den Verarbeitungsprozeß als äußerst problematisch: In vielen vorgeschlagenen Systemen kann sich der Zustand der Daten zwischen der Signalisierung eines Events und der Auswertung der Bedingung bzw. zwischen Bedingungsüberprüfung und der Ausführung der Aktion *unkontrollierbar* verändern [BBWZ95][2]. Ein Grund hierfür sind die komplexen Event-Spezifikationen, wie sie durch Event-Algebren für *ECA*-Regeln möglich werden (z.B. [GD93]), denn dadurch wird der Erkennungsprozeß sehr aufwendig. Außerdem kann beim Einsatz von Regeln, die durch den lesenden Zugriff, d.h., gerade *während* der Prüfung der Integritätsbedingung durch den Condition-Teil einer Regel, ausgelöst werden, der Zustand der Daten wieder verändert werden[3].

Beispiel: In einem Objekttyp O soll der Benutzer, der den letzten Zugriff durchgeführt hat, für Dokumentationszwecke gespeichert werden. Dazu könnte man intuitiv folgende Regel benutzen:

```
Event:    ON ACCESS OF O
Action:   UPDATE O
          SET O.letzte_benutzung = actual_user
```

Das Problem einer solchen Regel besteht nun darin, daß bei jeder Abfrage das Attribut `letzte_benutzung` geändert wird, also insbesondere auch dann, falls das Attribut `letzte_benutzung` abgefragt wird. Wird die Regel nun *vor* dem Zugriff ausgewertet, so erhält der Benutzer *immer* seine eigene Kennung als Ergebnis. Wird die Regel nach dem Zugriff ausgewertet, entspricht der zurückgegebene Wert nicht dem letzten Zugriff, sondern dem vorletzten.

Dieses einfache Beispiel macht schon deutlich, wie stark die Events auch in den Prozeß der Anfrage- und Operationsverarbeitung eingreifen können. Werden die Spezifikationsmöglichkeiten der bereits erwähnten komplexen Event-Algebren nun tatsächlich zur Erfassung von Ablaufbedingungen in CAD-Umgebungen genutzt, können (oder müssen, denn dies ist gerade der Anspruch einer ereignisgesteuerten Verarbeitung) dadurch sehr unübersichtliche Abhängigkeiten entstehen. Neben dem Problem der generellen Semantik der entstehenden Regelmenge (siehe auch im folgenden Abschnitt) ergibt sich für die Integritätssicherung in solchen Umgebungen verstärkt das Problem der Zusicherung bestimmter Bedingungen, denn im Rahmen des Entwurfs einer Regel entsteht auch die Frage, inwieweit man sich bei der Realisierung des Action-Teils A auf die Gültigkeit der Bedingung C verlassen kann, diese also tatsächlich so etwas wie eine Vorbedingung zur Ausführung darstellt. Es kann nicht mehr davon ausgegangen werden, daß das Objekt, das als Ergebnis einer Anfrage von einem System geliefert wird, die Selektionsbedingung noch erfüllt. Damit muß für eine Zusicherung von Bedingungen solange eine iterative Auswertung derselben stattfinden, bis sich der Wert nicht mehr ändern kann. Das bedeutet, daß eine Fixpunktsemantik realisiert werden muß!

[2] In vielen Modellen zu *ECA*-Regeln finden sich auch *explizite* Verzögerungsmechanismen zwischen den Teilen einer Regel. Diese sind hier jedoch nicht angesprochen.

[3] Derartige Ereignisse sind in vielen Vorschlägen, besonders bei objektorientierten DBMS, explizit oder implizit enthalten.

Die bisherige Diskussion hat deutlich gemacht, daß der Einsatz von *ECA*-Regeln in CAD-Umgebungen zur *direkten* Erfassung der Strukturen einer CAD-Umgebung schwierig ist. Darüber hinaus führen jedoch auch scheinbar für *ECA*-Regeln passende Probleme in CAD-Umgebungen, wie z.B. die Erfassung von Vorgängen innerhalb von CAD-Umgebungen, mit deren Hilfe dann vom System die Einhaltung bestimmter Standards in der Verarbeitung automatisiert und garantiert werden könnten, zur semantischen Problemen wie der nun folgende Abschnitt zeigt.

3 Semantik einer Regelmenge

Beschreibt man bestimmtes Verarbeitungswissen mit Hilfe von *ECA*-Regeln, so besteht prinzipiell die Möglichkeit, daß zu einem Zeitpunkt mehrere Regeln ausgelöst sein können, wie folgendes Beispiel verdeutlicht.

Beispiel: Auf einem Chip sollen verschiedene Logikfunktionen realisiert werden. Dabei kann es passieren, daß die verfügbare Chip-Fläche nicht ausreicht. Es soll nun beschrieben werden, daß in einem solchen Fall entweder die Länge oder die Breite des Chips angepaßt werden soll. Wir gehen davon aus, daß der Flächenbedarf einer Funktion über die Attribute `laenge` und `breite` im Objekt FUNKTION enthalten ist sowie daß der zugehörige Chip über `FUNKTION.CHIP` erreichbar ist. Weiterhin soll der CHIP die größte zusammenhängende freie Fläche über das Attribut `restflaeche` zur Verfügung stellen. Außerdem sollen im Chip ebenfalls die Attribute `breite` und `laenge` verfügbar sein.

```
Event:    ON CREATE OF FUNKTION
Condition:IF NEW.CHIP.restflaeche <=
             NEW.laenge * NEW.breite
          THEN
Action:   UPDATE CHIP
          SET CHIP.laenge = CHIP.laenge +
                            NEW.laenge -
                            (CHIP.restflaeche/ NEW.breite)
          WHERE CHIP = NEW.CHIP

Event:    ON CREATE OF FUNKTION
Condition:IF NEW.CHIP.restflaeche <=
             NEW.laenge * NEW.breite
          THEN
Action:   UPDATE CHIP
          SET CHIP.breite = CHIP.breite +
                            NEW.breite -
                            (CHIP.restflaeche/ NEW.laenge)
          WHERE CHIP = NEW.CHIP
```

```
Event:    ON UPDATE OF CHIP.(breite, laenge)
Action:   UPDATE CHIP
          CHIP.restflaeche = CHIP.restflaeche +
                             (NEW.breite - OLD.breite) *
                             NEW.laenge +
                             (NEW.laenge - OLD.laenge) *
                             NEW.breite
```

Die Alternative zwischen der Anpassung der Länge oder der Breite des Chips wird hierbei intuitiv über alternative Regeln erfaßt.

Durch solche Zusammenhänge, die hier natürlich offensichtlich sind, entsteht das Problem der Festlegung, *wie* eine solche Regelmenge abgearbeitet werden soll, denn das Regelparadigma selbst konzentriert sich ja gerade auf die Verarbeitung *einer* Regel. Bezüglich des *Ergebnisses* der Regelverarbeitung (also der operationalen Semantik) entstehen Probleme, wenn sich Teile verschiedener Regeln überlappen, z.B. dadurch, daß die Aktionen zweier Regeln (möglicherweise, siehe unten) dieselben Daten verändern oder die Bedingung einer Regel sich auf Daten bezieht, die von einer anderen Regel verändert werden. Daraus entstehen dann *Reihenfolgeabhängigkeiten.* In unserem Beispiel ist das Ergebnis offensichtlich davon abhängig, *wann* die `restflaeche` des betroffenen CHIPs geändert wird: Wird bei der Erzeugung einer neuen Funktion festgestellt, daß die `restflaeche` des zugehörigen CHIPs nicht mehr ausreicht und wird nach der Anpassung der `laenge` die `restflaeche` angepaßt, so ist die Bedingung der Regel, die die `breite` anpassen würde, nicht mehr erfüllt und die `breite` des CHIPs wird nicht verändert. Wird dagegen die `restflaeche` nicht direkt nach der Veränderung der `laenge` angepaßt, so wird auch die `breite` verändert. Das bedeutet, daß es ausgehend von einem konkreten Datenbankzustand eine Menge verschiedener Ergebniszustände gibt, die durch die Anwendung der Regeln erreichbar sind. Solche Mehrdeutigkeiten sind aus unserer Sicht im Einsatz zur Integritätssicherung nicht tolerierbar. Zur Lösung des Problems sind zwei verschiedene Ansätze denkbar: Entweder man erkennt die grundsätzliche Gefahr der Mehrdeutigkeiten und weist jeweils nach, daß es keine unterschiedlichen Ergebniszustände gibt, oder aber man erweitert die Sprache der Regeln um Kontrollstrukturen.

- Erkennung von Mehrdeutigkeiten
  Dieses Problem wird in den theoretischen Arbeiten zu Term- bzw. Wortersetzungssystemen im Begriff der Konfluenz erfaßt. Ein Regelsystem ist dann konfluent, wenn sichergestellt ist, daß trotz eventueller Überlappungen immer genau ein Ergebniszustand erreicht wird (Normalform). Der notwendige Begriffsrahmen wurde überblicksartig in [Hue80] dargestellt und beispielsweise in [AWH92, AHW95] auf den Bereich der (re-)aktiven DBMS zugeschnitten. Allerdings ist die Frage, ob eine gegebene Menge von Regeln konfluent ist, unentscheidbar. Daneben besteht im Bereich der *ECA*-Regeln noch ein weiteres Problem: Während bei der Betrachtung von Term-/Wortersetzungssystemen in der Regel davon ausgegangen wird, daß es entscheidbar ist, ob überhaupt eine Überlappung (Operationen auf gemeinsamen Daten) vorliegt, ist selbst dieses Problem bei *ECA*-Regeln deshalb nicht entscheidbar, da als Formalismus für die Bedingung C, wie schon gesagt, eine Logik 1. Ordnung bzw. noch mächtigere Konzepte eingesetzt werden. Ob sich die Aktionen zweier Regeln überlappen, ist unter anderem davon abhängig, ob die Condition-Teile der Regeln zu gleicher Zeit erfüllt sein können. Diese Frage ist jedoch schon für Logiken erster Ordnung auf Schemaebene nicht entscheidbar.
  Um die Konfluenz einer Menge von Regeln ohne die Einführung von Kontrollstrukturen (siehe unten) sicherzustellen, können im wesentlichen zwei verschiedene Ansätze verfolgt werden:

  1. Einschränkung der Regelsprache auf eine entscheidbare Teilmenge.
     Bei diesem Ansatz werden nur die Regeldefinitionen erlaubt, deren Konfluenz bezüglich der Bedingungen bzw. der Operationen entscheidbar ist. Damit werden beispielsweise viele Bedingungen, wie sie in SQL formuliert werden können, für die Formulierung des Condition-Teils ausgeschlossen. Insbesondere die komplexen Zusammenhänge in CAD-Umgebungen können in diesem Fall nicht mehr erfaßt werden.
  2. Einschränkung der Regelmengen auf entscheidbare Regelmengen.
     In diesem Ansatz wird die Regelsprache nicht generell auf einen entscheidbaren Anteil zurückgeschnitten, sondern man läßt zunächst alle Regeldefinitionen zu und entscheidet durch einen vorgegebenen Algorithmus, ob Konfluenz vorliegt. Dieser Algorithmus muß natürlich eine konservative Entscheidung treffen, kann also in dem Fall, daß keine Konfluenz nachweisbar ist, höchstens das Ergebnis „möglicherweise nicht konfluent" liefern. Entsprechende Regelmengen dürfen dann vom System nicht angenommen werden. Entsprechende Ansätze wurden z.B. in [Rei92, BW94] vorgestellt.

- Benutzung von Kontrollstrukturen
  Eine für *ECA*-Regeln in (re-)aktiven DBMS vorgeschlagene Möglichkeit zur generellen Vermeidung von Mehrdeutigkeiten sind Prioritäten, die für die Regeln vergeben werden. Eine Menge von Regeln, die zu einem Zeitpunkt ausgelöst sind, werden an Hand dieser Prioritäten geordnet und entsprechend abgearbeitet. Dadurch wird es möglich, eine eindeutige Reihenfolge der Auswertung und damit auch ein eindeutiges Ergebnis der Abarbeitung zu erreichen. Will ein Benutzer eine solche eindeutige Semantik erreichen, so erzwingt der Ansatz (im Extremfall) die vollständige Spezifikation aller Prioritäten. Dies entspricht einer komplexen IF-THEN-ELSE-Programmierung, was im Fall von Regelmengen noch dadurch erschwert wird, daß diese keine Struktur aufweisen, während ein imperatives Programm in der Regel mindestens die Sequenz zur Anordnung benutzt.

Die beschriebenen Alternativen zur Erreichung eines eindeutigen Ergebnisses der Regelverarbeitung sind aus unserer Sicht in CAD-Umgebungen nicht gangbar: Die Einschränkung der Formulierungsmöglichkeiten zur Erzwingung der Konfluenz machen das Konzept wegen der geringen Aussagekraft unbrauchbar während die Einführung von Kontrollstrukturen den (wesentlichen) Aspekt der „einfachen" Definition Regeln ad absurdum führen. Es wird daher fragwürdig ist, ob das *ECA*-Regel-Paradigma überhaupt geeignet ist, diese Aufgabe zu realisieren.
Während das eben angesprochene Problem im Bereich aller nichtmonotonen Regelsysteme besteht und durch die Zusammenfassung verschiedener Konzepte innerhalb von *ECA*-Regeln nur verstärkt wird, zeigt der folgende Abschnitt ein spezielles Problem von *ECA*-Regeln im Rahmen der Integritätssicherung in einem DBMS auf.

4 Sicherung der Integrität

Wie bereits bemerkt, sollte ein System *Garantien* für die Gültigkeit der definierten IBs zu vorgegebenen Zeitpunkten übernehmen. Unter der Nutzung einer geeigneten Sprache für die Teile einer Regel (z.B. Rekursion innerhalb der Anfragesprache) scheinen *ECA*-Regeln als Beschreibungsmittel für IBs äußerst attraktiv, da sie neben der Überprüfung einer Bedingung (C) auch die Angabe einer Reaktion auf die Verletzung dieser Bedingung erlauben (A). In diesem Action-Teil kann dann neben dem vollständigen Abbruch der laufenden Transaktion auch versucht werden, durch Korrekturoperationen bereits geleistete Arbeit zu erhalten und die verletzte Bedingung wieder in Kraft zu setzen. Daneben ist durch die häufige Benutzung von regelähnlichen Formulierungen in der Umgangssprache ein intuitives Verständnis vorgegeben.

In klassischen DBMS sind die Zeitpunkte, an denen die IBs gelten müssen, in der Regel das Ende von ACID-Transaktionen. In CAD-Umgebungen werden in der Regel entwurfsspezifische Ablaufeinheiten definiert, wobei die ACID Eigenschaften an unterschiedlichen Verarbeitungszeitpunkten und in unterschiedlicher Komplexität realisiert werden. Beispielsweise wird von *jedem* Verarbeitungsschritt gefordert, daß er alle dem Datenmodell inhärenten Integritätsbedingungen erfüllt, während erst nach Abschluß des gesamten Prozesses die Integritätsbedingungen, die das Entwurfsziel erfassen sollen, vollständig erfüllt sein können. Dennoch wird *jede* Entwurfsmethodologie Zeitpunkte definieren, zu denen bestimmte Bedingungen gelten müssen bzw. ab denen, im Sinne einer schrittweisen Verbesserung, bestimmte Bedingungen nicht mehr zurückgenommen werden dürfen, sondern für alle dann erreichten Zustände gelten müssen.

Werden *ECA*-Regeln zur Erfassung der Integritätsbedingungen eingesetzt, d.h., daß mit der Bedingung C die entsprechende IB in negierter Form beschrieben (die dann erfüllt ist, wenn es Objekte gibt, die die IB verletzen) und mit der Aktion A die Korrekturoperationen spezifiziert werden, so *garantiert* dieses Konzept jedoch in keiner Weise die *Einhaltung* der Integritätsbedingung: Die Aktion A einer Regel ist nur *syntaktisch* an die Bedingung C gebunden. Aus diesem Grunde garantiert die Ausführung einer Regel in keiner Weise, daß das erwünschte *Ziel*, die Einhaltung einer bestimmten Bedingung, erreicht wird. Bietet ein System echte *CA*-Regeln an seiner Schnittstelle an, so kann das Problem gelöst werden, da damit *jede* Veränderung des DB-Zustandes erkannt werden kann und man nicht daran gebunden ist, *alle* Events die zur Verletzung der Bedingung führen könnten, explizit zu formulieren. Werden in letzterem Fall nämlich bei der Formulierung der Regeln nicht alle Events bzw. Verbindungen von Events und Bedingungen erfaßt, so können Integritätsverletzungen unentdeckt bleiben, wie folgendes Beispiel aufzeigt:

Beispiel: Die in einer Firma laufenden Projekte sind der jeweiligen Abteilung zugeordnet. Wird eine Abteilung aufgelöst, so werden die noch laufenden Projekte anderen Abteilungen zugeordnet. Beispielsweise werden Projekte im Bereich „Informationsverarbeitung" bis zu einer weiteren Entscheidung der Stabsabteilung „Informationsplanung" zugeordnet. Intuitiv läßt sich dies mit folgender Regel spezifizieren:

```
Event:    ON DELETE OF ABTEILUNG
Condition:IF (SELECT *
                FROM PROJEKT
                WHERE PROJEKT.abteilung = OLD
               )
Action:   UPDATE PROJEKT
          SET PROJEKT.abteilung = ``Informationsplanung''
          WHERE PROJEKT.abteilung = OLD
          ⋮
```

Diese Regel garantiert die Einhaltung der definierten Integritätsbedingung solange, bis die Abteilung „Informationsplanung" gelöscht wird.

Das Beispiel macht deutlich, daß alleine *ECA*-Regeln die Integrität des DB-Zustandes oder des Ablaufes nicht garantieren können. Dies bedeutet, daß die entsprechende Anforderung bei der Integritätssicherung explizit durch zusätzliche Maßnahmen erreicht werden muß. Einige Systeme (z.B. ODE [GJS92]) erweitern daher die Abarbeitungsstrategie so, daß nach dem Ende der Aktion die Bedingung C noch einmal getestet wird und die Transaktion abgebrochen wird, wenn es immer noch Objekte gibt, die die Bedingung verletzen. In [Rei95] wird die Erweiterung des *ECA*-Regelmodells zu einem *ECAC*-Regelmodell vorgeschlagen, wobei die zweite Bedingung das *Ziel* der Regel erfaßt und dadurch eine gewisse Unabhängigkeit zwischen der Vorbedingung einer Regel (ausgedrückt durch die erste Bedingung) und deren Nachbedingung (ausgedrückt durch die zweite Bedingung) erreicht. Die Umsetzung solcher Vorschläge erweitert die Möglichkeit der *garantierten* Einhaltung von IBs durch das System, allerdings können sie nicht vollständig überzeugen (vor allem im Zusammenspiel mit anderen Regeln oder Regelkomponenten). Es bleibt die Frage, *warum* zur „Implementierung" von Integritätsbedingungen *ECA*-Regeln benutzt werden sollen und nicht besser spezialisierte Strukturen, z.B. speziell generierte Zugriffsfunktionen.

5 Diskussion und Zusammenfassung

Ein Unterstützungssystem für Entwurfsanwendungen wird nur dann sinnvoll, wenn es für die *Qualität* der entworfenen Objekte gewisse *Garantien* abgeben kann. Dazu zählt unter anderem auch die Einhaltung definierter Konsistenzbedingungen. Im vorliegenden Aufsatz haben wir die Frage aufgegriffen, ob aktive DBMS bzw. das ihnen zugrundeliegende Konzept der *ECA*-Regeln diese Aufgabe direkt übernehmen können. Aus unserer Sicht ergeben sich dabei erhebliche Probleme. Der Versuch, die den Regeln inhärente Folgerungsmöglichkeit zu nutzen und dadurch Eigenschaften der komplexen Objekte in Integritätsbedingungen zu erfassen (rekursive Strukturen), führt zu einer unübersichtlichen Menge von Regeln, die der Entwerfer sicher nicht direkt für komplexe Situationen entwerfen kann. Daher scheint es für Bedingungen, die sich auf den Zustand von Objekten beziehen, sinnvoller, eine geeignete Anfragesprache, die das komplexe Objektmodell voll unterstützt, zur Verfügung zu stellen.

Auch das Event-Konzept von *ECA*-Regeln führt weitere Probleme bei der Übernahme von Konsistenzgarantien durch ein aktives DBMS ein. Falls der Zugriff auf ein Objekt als Event in der Regeldefinition benutzt werden kann, sind Aussagen über den Zustand *nach* Abschluß der Überprüfung einer Konsistenzbedingung ohne weitere Maßnahmen nicht mehr möglich. Aus diesem Grund sind in solchen Fällen aufwendige Auswertungsprozeduren notwendig (Fixpunktsemantik).

Die Problematik der Erfassung der Semantik großer Mengen von Produktionsregeln ist nicht neu, jedoch trägt auch die Technologie aktiver DBMS dazu keine neuen Erkenntnisse bei, so daß es bei einer *direkten* Nutzung der Regeln zur Modellierung nur die Möglichkeiten der starken Einschränkung der Regelsprache oder aber die Einführung von Kontrollstrukturen für Regelmengen bleiben. Während ersteres gerade für die komplexen Zusammenhänge in CAD-Umgebungen sicher nicht geeignet ist, muß die Frage gestellt werden, ob die Einführung von Kontrollstrukturen nicht die gesamte Attraktivität des regelbasierten Ansatzes zur Konsistenzsicherung vernichtet. Daher werden immer wieder Entwurfswerkzeuge für Regelmengen gefordert, wobei es dann zu einer *indirekten* Nutzung von Regeln kommt, d.h., das Modell der Konsistenzbedingungen des Entwurfswerkzeuges wird dann in Regeln übersetzt. Damit liegt die Verantwortung für die Semantik ganz im Entwurfswerkzeug bzw. im Übersetzer. Hier ist die Frage, ob das Ziel dieser Übersetzung *ECA*-Regeln sein müssen. Denn diese werden eventuell nicht nur noch als Code-Schablone benutzt. Es könnten jedoch ebensogut (vielleicht sogar besser) direkt spezielle Zugriffsfunktionen generiert werden und über ein verbindliches Bibliothekskonzept für die Manipulationsfunktionen von Objekten genutzt werden kann.

Der Einsatz von Entwurfswerkzeugen scheint nicht nur erforderlich um einer Menge von *ECA*-Regeln überhaupt eine abstrakte Semantik zuordnen zu können, sondern gerade auch im Bezug auf den Einsatzzweck der *Konsistenzsicherung* müssen solche Werkzeuge die geforderten Garantien realisieren, die das Konzept der *ECA*-Regeln selbst nicht leisten kann.

Als Ergebnis dieser Untersuchung ist festzuhalten, daß *ECA*-Regeln ohne weitere Maßnahmen (z.B. Entwurfswerkzeuge) keinen Beitrag zur Konsistenzsicherung in CAD-Umgebungen leisten können. In [Rei95] wird darüberhinaus ein Regelsystem entwickelt, das die aufgezeigten Problematiken in Betracht zieht und Lösungen anbietet. Allerdings zeigt sich dabei, daß die Komplexität der Regelverarbeitung erheblich wächst, so daß ein Einsatz zur Sicherung der Konsistenz nicht in Betracht zu kommen scheint.

Literatur

[AHW95] A. Aiken, J. M. Hellerstein und J. Widom. Static Analysis Techniques for Predicting the Behavior of Active Database Rules. *ACM TODS*, 20(1):3–41, März 1995.

[AWH92] A. Aiken, J. Widom und J. M. Hellerstein. Behavior of Database Production Rules: Termination, Confluence, and Observable Determinism. *ACM SIGMOD Record*, 21(2):59–68, Juni 1992.

[BBWZ95] J.A. Blakeley, A.P. Buchmann, D.L. Wells und J. Zimmermann. Building an Integrated Active OODBMS: Requirements, Architecture, and Design Decisions. In *Proc. of the Data Engineering'95*, 1995.

[BW94] E. Baralis and J. Widom. An Algebraic Approach to Rule Analysis in Expert Database Systems. In *Proceedings of the VLDB'94, Santiago, Chile*, S. 475–486, September 1994.

[Day88] U. Dayal. Active Database Management Systems. In C. Beeri, J.W. Schmidt und U. Dayal (Herausgeber), *Proceedings of the 3rd International Conference on Data and Knowledge Bases: Improving Usability and Responsiveness*, S. 150–169. Morgan Kaufmann Publishers, Inc., Juni 1988.

[For81] C.L. Forgy. OPS5 Users's Manual. Technical Report CMU-CS-81-135, Canergie-Mellon University, 1981.

[GD93] S. Gatziu and K.R. Dittrich. An Event language for the active object-oriented Database System SAMOS. In W. Stucky and A. Oberweis (Herausgeber), *Tagungsband der 5. BTW*, S. 94–103. Gesellschaft für Informatik, Springer-Verlag, März 1993.

[GJS92] H.H. Gehahi, H.V Jagadish und O. Shmueli. Event Specification in an Active Object-Oriented Database. *ACM SIGMOD Record*, 21(2):81–90, Juni 1992.

[HM75] M.M. Hammer and D.J. McLeod. Semantic Integrity in a Relational Data Base System. In *Proceedings of the VLDB'75, Framingham, Mass.*, S. 25–47, September 1975.

[HNSB90] D. Harrison, R. Newton, R. Spickelmier und T. Barnes. Electronic CAD Frameworks. *Proc. of the IEEE*, 78(2):393–417, Februar 1990.

[Hue80] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 24(4):797–821, Oktober 1980.

[Rei92] J. Reinert. Sicherung der Referentiellen Integrität - Optimierung durch Schemaanalyse. ZRI-Bericht 5, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, 1992.

[Rei94] J. Reinert. Eventbegriffe in Datenbanksystemen. *Datenbankrundbrief*, (14):11–14, 1994.

[Rei95] J. Reinert. *Ein Regelsystem zur Integritätssicherung in aktiven relationalen Datenbanksystemen.* Dissertation, Universiät Kaiserslautern, 1995. In Vorbereitung.

[RMH+94] N. Ritter, B. Mitschang, T. Häder, M. Gesmann und H. Schöning. Capturing Design Dynamics - The CONCORD Approach. In *Proc. 10th Int. IEEE Data Engineering Conference, Houston, Texas*, S. 440–451, 1994.

[RR] J. Reinert and N. Ritter. Applying ECA-Rules in DB-based Design Environments. zur Veröffentlichung eingereicht.

[RS92] F. J. Rammig and B. Steinmüller. Frameworks und Entwurfsumgebungen. *Informatik Spektrum*, 15(1):33–43, 1992.

[Sch93] H. Schöning. *Anfrageverarbeitung in Komplexobjekt-Datenbanksystemen.* Deutscher Universitätsverlag, 1993.

[SHP88] M.R. Stonebraker, E.N. Hanson und S. Potamianos. The POSTGRES Rule Manager. *IEEE TOSE*, 14(7):897–907, 1988.

[WF90] J. Widom and S. J. Finkelstein. Set-Oriented Production Rules in Relational Database Systems. *ACM SIGMOD Record*, 19(2):259–270, Juni 1990.