communications in mathematical
match
and in computer chemistry

no. 37, March 1998

ISSN 0340-6253

MATCDY (37) 7-156 (1998)

Objektorientierte Darstellung und Algorithmen zur Klassifizierung endlicher bewerteter Strukturen†

Christof Benecke‡
Lehrstuhl II für Mathematik, Universität Bayreuth
D-95440 Bayreuth

1. Überblick

Gegenstand dieser Arbeit ist eine neue Methode zur Klassifizierung endlicher bewerteter Strukturen nach Ähnlichkeit (incl. Isomorphie). Insbesondere werden dabei auch unscharfe oder Fuzzy - Strukturen berücksichtigt, wie beispielsweise Moleküle in räumlicher Darstellung.

Die vorgestellten Methoden wurden bereits erfolgreich zur Lösung der folgenden Fragestellungen bei unterschiedlichen Strukturtypen eingesetzt:

Graphen	*Isomorphietest, kanonische Numerierung, Teil-/ Subgraphensuche*
Designs	*Isomorphietest, Berechnung der Automorphismengruppe*
Topologische molekulare Graphen	*Isomorphietest, kanonische Numerierung, Teil-/ Subgraphen-/Ringsuche*
Dreidimensionale molekulare Graphen	*Isomorphietest, Ähnlichkeitserkennung, kanonische Numerierung, Teil-/Subgraphensuche*

Während bei den ersten drei Strukturtypen ein exakter Isomorphiebegriff existiert, mußte vor allem bei dreidimensionalen molekularen Graphen ein Ähnlichkeitsbegriff definiert werden, der eine Identifizierung ermöglicht. Insbesondere sollten dabei auch geringfügige Toleranzen - beispielsweise in der geometrischen Darstellung - als zulässig vorgegeben werden können.

---

†Diese Arbeit wurde von der Fakultät für Mathematik und Physik der Universität Bayreuth als Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigt. 1. Gutachter: Prof. Dr. A. Kerber, 2. Gutachter: Prof. Dr. R. Laue. Tag der Einreichung: 5. März 1997, Tag des Kolloquiums: 22. Mai 1997

Gefördert durch das **BMBF** Projekt *„Entwicklung eines stereochemischen Strukturgenerators zur Verwendung im Informationssystem Spektroskopie / Spec-Info"* und das **DFG** Projekt *„Erkennung, Beschreibung und Visualisierung molekularer Strukturen"* (03KE7BAY9).

‡Jetzige Anschrift CSC PLOENZKE AG, Bayerstraße 36, D-80335 München, E-mail: cbenecke@csc.com

Ausgangspunkt der Arbeit war das Problem der Klassifizierung dreidimensionaler molekularer Strukturen. Die räumliche Plazierung der Strukturen wird durch eine Kraftfeldoptimierung erreicht. Da zum einen das zugrundeliegende numerische Verfahren bei unterschiedlichen Anfangsbedingungen unterschiedliche Ergebnisse liefert, aber auch mehrere räumliche Plazierungen eines topologisch identischen Moleküls existieren können, war es notwendig, ein Hilfsmittel zur *automatischen* Unterscheidung der erzeugten Moleküle zur Verfügung zu stellen.

1.1. Ziele

Zur Lösung dieser Aufgabe wird in der vorliegenden Arbeit ein geeignetes objektorientiertes Modell für eine große Anzahl von Strukturklassen geschaffen, das durch seinen allgemeinen Charakter sehr einfach auch an neue Strukturklassen angepaßt werden kann.

Die objektorientierte Darstellung erlaubt eine sehr übersichtliche Formulierung von Strukturen und ermöglicht dadurch auch eine leichte Anpassung an neue Struktureigenschaften.

In einem nächsten Schritt wird die Begriffe der Ähnlichkeit und Isomorphie für Strukturen, die diesem Modell entsprechen, definiert.

Das primäre Ergebnis dieser Arbeit ist ein *diskreter* Algorithmus, der aufgrund des objektorientierten Strukturmodells und des dafür vorgegebenen Ähnlichkeitsbegriffes eine Identifizierung von Strukturen ermöglicht, also eine sog. *Ähnlichkeitsabbildung* zwischen Strukturen findet.

Um dies zu erreichen, wird eine neue Methode entwickelt, die es prinzipiell ermöglicht, ohne Änderung der vorgestellten Algorithmen oder des daraus entwickelten Programmes, verschiedenste Strukturen zu verarbeiten.

Das bedeutet, daß bei einem Wechsel der Klasse der Strukturen (z. Bsp. molekulare Graphen) zu einer anderen Klasse (beispielsweise Designs) kein neuer Algorithmus zur Klassifizierung erstellt werden muß. Zudem ist es möglich - unter Verwendung entsprechender Abbildungen - Strukturen aus unterschiedlichen Klassen zu vergleichen.

Dieses Ziel wird mit Hilfe von abstrakten Datentypen erreicht, deren Schnittstellen an die jeweils vorliegenden Eigenschaften der Struktur, angepaßt werden können.

Setzt der Anwender dieser Theorie die vorgestellten Algorithmen ein, oder entwickelt er neue Verfahren, die ausschließlich mit diesen Methoden arbeiten, muß er also lediglich die Schnittstelle der betrachteten Klasse von Strukturen zur Verfügung stellen.

Ausgehend von der allgemeinen objektorientierten Strukturdefinition werden nachfolgend verschiedene Algorithmen zur Klassifizierung vorgestellt. Neben der Isomorphie bzw. Ähnlichkeit und Teilstrukturisomorphie ist es u. a. möglich, anhand der gegebenen Strukturbeschreibung eine kanonische Darstellung der Strukturen zu berechnen, also auch das wichtige Normalformenproblem zu lösen.

1.2. Aufbau

Die Arbeit ist in mehrere Abschnitte gegliedert. Der erste Teil beinhaltet wichtige, in den nachfolgenden Kapiteln verwendete Definitionen, vor allem aus dem Bereich der objektorientierten Analyse. Ein Bereich, der in der Mathematik einen wesentliche natürlicheren Ansatz bei der Modellbildung erlaubt, jedoch noch nicht sehr verbreitet ist.

Im zweiten Abschnitt werden die notwendigen Hilfsmittel zur objektorientierten Darstellung von endlichen bewerteten Strukturen spezifiziert. Dabei werden auch einige herkömmlichen Darstellungsarten vorgestellt und mit der neuen, erweiterten Darstellung verglichen. Daneben wird der Begriff der Ähnlichkeit und insbes. der Isomorphie von Strukturen definiert.

Der dritte Teil beschäftigt sich mit Algorithmen zur Isomorphie- und Ähnlichkeitserkennung sowie zur Teilstrukturerkennung und kanonischen Numerierung von Strukturen. Die Algorithmen beruhen auf der zuvor definierten objektorientierten Strukturdarstellung, und sind ebenfalls in objektorientierter Form implementiert. Anhand von Beispielen wird erklärt, welchen Einfluß der Austausch der Informationen zwischen dieser neuartigen Darstellung und den Algorithmen auf die Effizienz der Verfahren hat.

Die Anwendung dieser Theorien erfolgt im vierten Teil der Arbeit. Die zuvor definierte Darstellung der Strukturen wird auf einige der eingangs erwähnten Strukturtypen übertragen und anhand von einfachen Beispielen erläutert. Besondere Beachtung finden vor allem die molekularen Graphen, da hier insbesondere der Ähnlichkeitsbegriff von endlichen Strukturen benötigt wird. Mögliche Erweiterungen der verwendeten Modelle und weitere Einsatzgebiete sind ebenfalls in diesem Abschnitt zu finden.

Inhaltsverzeichnis

1. Überblick 7
1.1. Ziele 8
1.2. Aufbau 8

2. Objektorientierte Analyse 11

3. Strukturen und ihre Darstellungen 13
3.1. Klassische Darstellung von Strukturen 13
3.2. Objektorientierte Darstellung 17
3.3. Ähnlichkeit 35

4. Algorithmen 41
4.1. Ähnlichkeitserkennung 41
4.2. Kanonische Numerierung 65
4.3. Teilstrukturisomorphie 78
4.4. Berücksichtigung von Isomorphieeigenschaften 88
4.5. Komplexität 91

5. Graphen 93
5.1. Schnittstelle für Graphenisomorphie 93
5.2. Schnittstelle für Teil-/Subgraphenisomorphie 94
5.3. Schnittstellen zur kanonischen Numerierung 99

6. Moleküle 101
6.1. Chemische Strukturen 101
6.2. Diskretisierungen für topologische Identität 103
6.3. Diskretisierungen zur Erkennung räumlicher Ähnlichkeit 120
6.4. Einsatz bei der Clusteranalyse/Datenbanken 143
6.5. Einsatz bei der Konformationsanalyse 144
6.6. Bestimmung der Ähnlichkeit von Molekülen 145
6.7. Einsatz bei der Spektrenvorhersage 145
6.8. Dockingprobleme in der Pharmazie 147
6.9. Einsatz in der Kombinatorischen Chemie 150

7. Andere Verfahren 151
7.1. 2D - Substruktursuche bei Molekülen 151
7.2. 3D - Klassifizierung von Molekülen 151
7.3. Kanonische Numerierung von Molekülen 152

8. Bibliographie 153

2. Objektorientierte Analyse

Die objektorientierte Analyse ist ein mächtiges Hilfsmittel zur Modellierung realer Probleme. In der Mathematik wird sie jedoch bisher selten eingesetzt. Sie erlaubt durch das Prinzip der Kapselung eine sehr einfache, modulare Formulierung komplexer Zusammenhängen.

Ihre Anwendung ist mit dem Einsatz einiger neuer Begriffe verbunden, die in diesem Abschnitt kurz erläutert werden sollen. Für eine Beschreibung in einem etwas ausführlicheren Kontext sind unter anderem [AH], [PK], [WWW]oder [CL] zu empfehlen.

Mit SYMMETRICA (vgl. [KK]) wurde bereits 1986 ein objektorientiertes Computer - Algebra System vorgestellt, das die Vorteile dieses Ansatzes nutzt. Inzwischen sind die Methoden zur Realisierung solcher Systeme natürlich weiter verbessert worden (C++, CLOS). Der Grundgedanke der objektorientierten Darstellung, wie er nachfolgend beschrieben wird, ist jedoch gleich.

Im Gegensatz zum funktionalen Ansatz betrachtet man beim objektorientierten Ansatz sogenannte *abstrakte Datentypen*. Sie dienen primär zur *Vereinfachung* des Entwurfs und der Implementierung von Programmsystemen, indem Daten und darauf operierende Funktionen zu einer Einheit zusammengesetzt werden. Das bedeutet, daß ein abstrakter Datentyp bestimmte Informationen enthält und in der Lage ist, spezifische Operationen auf diesen Informationen durchzuführen. In diesem Zusammenhang spricht man auch von der *Kapselung*, die weiter unten definiert wird.

Abstrakte Datentypen zeichnen sich dadurch aus, daß sie:

- eine Typdefinition exportieren
- Operationen auf dem Typ definieren und exportieren, mit denen Instanzen dieses Typs manipuliert werden können,
- interne Daten dadurch geschützt sind, daß nur durch die vorherigen Operationen darauf zugegriffen werden kann,
- mehrfache Instantiierung des Typs erlauben und
- Axiome und Vorbedingungen ihren Einsatz beschreiben.

Eine *Instanz* ist eine zur Laufzeit eines Programmes existierende Einheit eines abstrakten Datentyps. Der Export der Typdefinition erlaubt es, mehrere Einheiten eines abstrakten Datentyps gleichzeitig zu erzeugen. Durch Axiome und Vorbedingungen wird beispielsweise die Plausibilität der in einem abstrakten Datentyp enthaltenen Daten gesichert.

Die auf einem abstrakten Datentyp realisierten Operationen bezeichnet man auch als *Methoden*. Die Daten auch als Attribute.

Kapselung ist das Prinzip, nach dem die Schnittstelle einer Software - Komponente auf notwendige Informationen beschränkt wird. Relevant ist, wie eine Komponente *verwendet* werden kann, nicht wie sie *realisiert* ist.

Dies bedeutet nicht, daß eine effektive Realisierung von Verfahren ohne Bedeutung ist, sondern daß ihre Komplexität dem Anwender verborgen bleibt.

Übertragen auf das Problem der Darstellung von Strukturen ist die Kapselung damit ein wesentliches Hilfsmittel zur allgemeinen Beschreibung von Strukturen. Der große Vorteil gegenüber früheren Ansätzen ist, daß der Strukturaufbau nicht mehr von vorne herein bekannt sein muß, sondern Informationen darüber lediglich über eine exakt definierte Schnittstelle ausgetauscht werden.

Objektorientiertes Design von Anwendungen besteht in der Regel aus einer Sammlung von abstrakten Datentypen. *Objekte* sind die (in einer entsprechenden Programmiersprache realisierten) Instanzen der Implementierung abstrakter Datentypen. *Objektidentitäten* sind Referenzen für Objekte.

Die Implementierung eines abstrakten Datentyps bezeichnet man als *Klasse*. Sie ist die exakte Beschreibung aller Daten und Methoden einer Menge von gleichartigen Objekten.

Zusammenfassend kann man sagen:

- Klassen sind statische Beschreibungen und entsprechen dem Typ eines Objektes.
- Objekte sind Variablen bzw. Instanzen einer Klasse. Sie existieren nur zur Laufzeit eines Programmes.

Es bestehen Analogien zwischen den Begriffen Klasse und Typ und Objekt und Variable.

Die Fähigkeit, Klassen durch einen (oder mehrere) Typen zu parametrisieren, bezeichnet man als *Generizität*.

Beispielsweise ist es sinnvoll, bei der Definition eines abstrakten Datentyps STAPEL, einen generischen Parameter ART zu verwenden, der dann durch einen vorgegebenen Typ ersetzt werden kann. So ist es möglich, STAPEL von Textfeldern aber auch STAPEL von komplexeren Typen zu erzeugen, ohne explizit mehrere Definitionen von STAPEL zu verwenden.

Ein in der objektorientierten Analyse sehr oft verwendeter Begriff ist die *Vererbung*. Dabei handelt es sich um einen Mechanismus, der es dem Objekt einer Klasse erlaubt, auf Daten und Methoden einer bereits früher definierten Klasse zugreifen zu können. Eine Klasse *erbt* Daten und Methoden einer anderen Klasse. Sie sind dann Bestandteil der abgeleiteten Klasse und müssen dort nicht neu definiert werden.

Kann ein Objekt Eigenschaften von mehreren Elternklassen erben, so nennt man dies auch *Polymorphismus*.

Verwendet man eine Klasse und die darin definierten Daten und Methoden ausschließlich als Elternklasse für andere Klassen, so spricht man von *abstrakten Klassen*.

Eine abstrakte Klasse ist damit eine Klasse, von der keine Instanzen erzeugt werden. Sie dient ausschließlich als Elternklasse für andere Klassen, die ihre Eigenschaften erben.

Abstrakte Klassen sind ein wesentliches Element zur allgemeinen Beschreibung von Strukturen. Mit Hilfe einer abstrakten Klasse ist es möglich, Methoden und Daten zu verwenden, denen erst in einer abgeleiteten Klasse spezifischer Inhalt gegeben wird. Allgemein nennt man Methoden, die in einer Elternklasse definiert und in einer abgeleiteten Klassen mit gleichem Namen und Argumenten, jedoch unterschiedlichen Inhalt verwendet werden, auch *virtuelle* Methoden.

Um die in Objekten zur Laufzeit enthaltenen Werte auch ablegen zu können (beispielsweise in einer Datenbank) und sie damit über die Programmausführung hinaus zu erhalten, benötigt man entsprechende Methoden zum Laden und Speichern von Objekten.

Die Eigenschaft von Objekten, über die Laufzeit des Programmes hinaus ihre Identität, also den Zustand ihrer Beschreibung zu erhalten, nennt man auch *Persistenz*.

Abstrakte Datentypen können miteinander in unterschiedlichen *Beziehungen* stehen. Sie können zum einen voneinander abgeleitet werden, es ist jedoch auch möglich, daß ein abstrakter Datentyp ein Attribut oder eine Attributliste eines anderen Typs darstellt.

Verwendet man beispielsweise zur Darstellung des abstrakten Datentyps Graph die weiteren Typen Knoten und Kante, dann wäre Kante eine mögliche Realisierung einer Beziehung zwischen zwei Knoten. Der abstrakte Datentyp Kante enthielte dann zwei Objekte von Typ Knoten.

Die Darstellung logischer Beziehungen zwischen Objekten kann also durch Objekte eines anderen abstrakten Datentyps geschehen, wobei dieser wiederum entsprechende Attribute und Methoden zur Verarbeitung zur Verfügung stellt.

Damit wurden die wichtigsten im folgenden verwendeten Begriffe aus der Objekttechnologie erklärt.

Die darüber hinaus in dieser Arbeit benötigten Grundlagen aus der Gruppen- und Graphentheorie werden als bekannt vorausgesetzt und an dieser Stelle nicht weiter definiert. Literatur dazu findet man bei [KE], [HUPP], [HA] (Gruppentheorie) bzw. [VL], [DJ] oder [GT] (Graphentheorie).

3. Strukturen und ihre Darstellungen

Dieses Kapitel befaßt sich mit der Frage, auf welche Art man diskrete Strukturen darstellen bzw. repräsentieren kann. Der erste Abschnitt befaßt sich mit der „klassischen", also bisherigen Darstellung von Strukturen. Im zweiten Teil betrachten wir dann die neue, objektorientierte Darstellung von Strukturen.

Die eigentliche Definition von Strukturen lautet (vgl. [VK] oder [HM]):

Sei S eine Menge. S wird auch als Trägermenge oder Grundmenge bezeichnet. Seien

- $\{\omega_i \mid i \in I\}$ *eine Menge von Operationen auf S* (ω_i: $S^{q_i} \rightarrow S$ *sei* q_i *- stellig),*
- $\{\rho_j \mid j \in J\}$ *eine Menge von Relationen auf S* ($\rho_j \subseteq S^{r_j}$ *sei* r_j *- stellig).*

Dann nennt man $(S, \{\omega_i \mid i \in I\}, \{\rho_j \mid j \in J\})$ *eine Struktur. Sind* $I = \{1, \ldots, m\}$ *und* $J = \{1, \ldots, n\}$, *so schreibt man auch:* $(S, \omega_1, \ldots, \omega_m, \rho_1, \ldots, \rho_n)$ *und bezeichnet diese als Struktur vom Typ* $(q_1, \ldots, q_m; r_1, \ldots, r_n)$

Viele mathematische Strukturen lassen sich mit Hilfe dieser Definition beschreiben. Für die Modellierung von komplexen Problemstellungen benötigt man jedoch eine Darstellung, die einen umfassenden praktischen Einsatz erlaubt. Praktisch bedeutet in diesem Fall „auf einem Rechner realisierbar". So ist in der obigen Definition beispielsweise keine Schachtelung von Relationen erlaubt, die jedoch zur Darstellung bei realen Strukturen, gerade aus dem Bereich der Datenbanken, heute üblich sind. Auch Bewertungen der Trägermenge oder von Relationen, wie sie bereits bei einfachen, bewerteten Graphen benötigt werden, sind darin nicht vorgesehen. Es wird also eine Darstellung benötigt, welche die tatsächliche Realisierung von Strukturen im Rechner besser berücksichtigt. Die Allgemeinheit der Darstellung sollte dabei möglichst groß sein, um neue Strukturen oder Modelle schnell und einfach implementieren zu können.

Zunächst wird ein Überblick über einige herkömmliche Darstellungsarten gegeben, bevor der neue, objektorientierte Ansatz erörtert wird.

Im objektorientierten Ansatz werden Informationen über eine diskrete Struktur - im Gegensatz zum klassischen Ansatz - durch eine genau definierte Schnittstelle geliefert. Diese Schnittstelle wird durch Methoden repräsentiert, die bei jeder neuen Beschreibung einer Struktur vorgegeben werden müssen.

3.1. Klassische Darstellung von Strukturen

Der klassische Ansatz beruht auf einem hierarchischen Modell. Man gibt einen bestimmten Satz von Operationen, die auf einer Trägermenge erlaubt sind, vor. Mit deren Hilfe bildet man neue Mengen „höherer Ordnung". Auf diesen Mengen höherer Ordnung sind ebenfalls bestimmte Operationen erlaubt, die wiederum Strukturen noch höherer Ordnung schaffen. Die diskrete Struktur wird also durch eine Baumstruktur beschrieben, wobei die Kanten zulässige Operationen darstellen, und die Knoten die Ergebnisse der Operationen auf derart iterativ aufgebauten Mengen bzw. der Trägermenge, die Wurzel in diesem Baum ist, sind.

Diese Art der Beschreibung entspricht dem aktuellen Stand der Forschung. In der Praxis werden diese Konstruktionen unter anderem auch in der Datenbanktechnologie zur Beschreibung komplexer diskreter Strukturen, und dort insbesondere von Datenstrukturen verwandt:

3.1.1. Stand der Forschung

In heutigen Datenbanksystemen werden nicht mehr ausschließlich „flache" Datenobjekte gespeichert

sondern nicht zuletzt durch den Einsatz multimedialer Elemente verstärkt auch komplexe Daten. Die Anforderungen, die an die Funktionalität der Abfragesprachen (SQL als klassisches Beispiel, siehe [DB2]) gestellt werden, wachsen in gleichem Maße. Grumbach und Vianu beschäftigen sich deshalb in [GRVI] mit der Komplexität und möglichen Optimierungsansätzen derartiger Abfragesprachen. Zur Darstellung der komplexen Datenbankobjekte greifen sie auf die Definitionen von [AH87] zurück. Danach sind komplexe Datenobjekte solche Objekte, die rekursiv aus atomaren Objekten unter Verwendung von Tupel- und Mengenkonstruktoren gebildet werden. „{}" sei der Mengenkonstruktor, unabhängig von der Kardinalität und „[]" der Tupelkonstruktor. U sei ein atomarer Typ und dom (U) eine endliche Menge von atomaren Attributen. Komplexe Typen und ihre Attribute werden dann ausgehend von einer gegebenen Teilmenge D von dom (U) definiert:

1. U sei ein atomarer Typ und dom (U, D) = D
2. Ist T ein Typ, dann ist {T} ein Mengentyp mit den Attributen: $\mathrm{dom}(\{T\}, D) = \zeta_{fin}(\mathrm{dom}(T, D))$

   wobei $\zeta_{fin}(S)$ eine Menge endlicher Teilmengen von S ist.
3. Seien $T_1,...,T_n$ atomare oder Mengentypen. Dann ist $[T_1,...,T_n]$ ein Tupeltyp mit der Attributmenge

   $\mathrm{dom}([T_1, \ldots, T_n], D) = \{[a_1, \ldots, a_n] \mid a_i \in \mathrm{dom}(T_i, D),\ i \in \{1, \ldots, n\}\}$

   Tupeltypen können nicht direkt ineinander verschachtelt werden.

Damit läßt sich jedes Objekt auch sehr einfach mit Hilfe eines beschrifteten Baumes darstellen:

3.1.1.1. Beispiel

Das komplexe Objekt { [U ,{ [U, U] }] } wird durch den folgenden Graphen beschrieben. (⊗ stehen für den Tupelkonstruktor, ⊛ für einen Mengenkonstruktor. □ wird für die Blätter verwendet.)

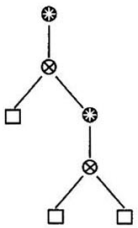


Abbildung 1: *Der Typ { [U ,{ [U, U] }]*

3.1.2. Erweiterungen

Eine Erweiterung dieser Darstellung wird in der Arbeit von Schimmel vorgenommen ([SCH]). Das Grundprinzip beruht wie zuvor darauf, diskrete Strukturen durch ineinander verschachtelte komplexe Objekte darzustellen. Schimmel definiert hierfür Grundobjekte, die durch Anwendung spezieller Konstruktoren zu komplexeren Objekten kombiniert werden können, welche wiederum Grundstrukturen für weitere, noch komplexere Objekte sind. Die diskrete Struktur wird also analog zu 3.1.1. aufgebaut. Im Gegensatz zu den bereits zuvor eingeführten Tupel- und Mengenkonstruktoren werden jedoch allgemeine Konstruktoren, die eine variablere Kombination der Grundelemente zulassen, verwendet. Der für die Klassifizierung dieser Strukturen verwendete Algorithmus stützt sich dabei auf [MK].

Die Grundobjekte bezeichnet Schimmel als **ATOME**. Um **ATOME** unterscheiden zu können, führt er die Funktion **Vergleichstest** ein. **ATOME** sind bei ihm typischerweise ganze Zahlen, Zeichenketten

oder reelle Zahlen.

Für die Konstruktion komplexerer Objekte definiert er die folgende Operation:

3.1.2.1. Definition

Sei Ω eine endliche Menge, $n \in \mathbb{N}$, $U \leq S_n$ eine Permutationsgruppe, die auf den n - Tupeln $(x_1, \ldots, x_n) \in \Omega^n$ operiert:

$$\circ : U \times \Omega^n \to \Omega^n$$

$$(u, (x_1, \ldots, x_n)) \to (x_{u^{-1}(1)}, \ldots, x_{u^{-1}(n)})$$

3.1.2.2. Bezeichnung

$[(x_1, \ldots, x_n)]_U := \mathrm{Bahn}_U((x_1, \ldots, x_n))$

3.1.2.3. Beispiel

Sei $\Omega = \{1, 2, 3, 4\}$, $S_4 \geq U = S_1 \oplus Z_3$, dann ist

$$[(1, 2, 3, 4)]_U = \{(1, 2, 3, 4), (1, 3, 4, 2), (1, 4, 2, 3)\}$$

Um eine vollständige Beschreibung der diskreten Struktur zu erhalten, ist ein nicht unerheblicher Aufwand erforderlich. Schimmel definiert im folgenden noch **Typen**, **Objekte**, **Bausteine** und **Träger** von Strukturen. Da lediglich der prinzipiellen Aufbau verdeutlicht werden soll, seien einige Vereinfachungen an dieser Stelle gestattet.

3.1.2.4. Definition

Es werden nun rekursiv *Typen* der Stufe $i \in \mathbb{N}$ definiert. $\mathcal{T}_i$ sei die Menge der Typen der Stufe i. Es gilt $St(T) = i \;\; \forall\, T \in \mathcal{T}_i$.

- $\mathcal{T}_0 := \{\mathrm{Atom}\}$. Atom ist ein beliebig gewählter Name.
- Für $\mathcal{T}_{i+1}$ gilt: $\mathcal{T}_{i+1} := \{[(T_1, \ldots, T_n)]_U\}$ mit

  1. $n \in \mathbb{N}$
  2. $U \leq S_n$
  3. $T_j \in \mathcal{T}_k \;\; \forall\, j \in \{1, \ldots, n\}$ und $k \leq i$
  4. $\exists j \in \mathbb{N}$ mit $(T_j \in \mathcal{T}_i)$

Diese Definition erlaubt nun, Typen zu erzeugen, die äquivalent zu den mit den in 3.1.1. vorgestellten Tupel- und Mengenkonstruktoren konstruierten komplexen Objekten sind.

Am Beispiel des gefärbten Würfels wird ein Eindruck vermittelt, wie die obigen Definitionen praktisch verwendet werden.

3.1.2.5. Beispiel

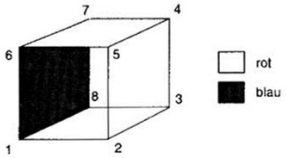


Abbildung 2: *„Primitive" diskrete Struktur - der gefärbte Würfel*

Die Grundmenge der diskreten Struktur, die **ATOME** sind in diesem Beispiel die Nummern der Ecken und die Farben der Seiten. Man beachte, daß die Farben hier nicht etwa als Attribute der Ecken verstanden werden.

ATOME ={ 1, ..., 8, rot, blau, weiß}

Die Elemente aus **ATOME** lassen sich nicht weiter in kleinere Einheiten zerlegen.

Stufe 0: Die Menge der Typen ist $\mathcal{T}_0$:= { Atom}.

Stufe 1: In dieser Stufe werden die Kanten des Würfels als neuer Typ definiert. Man setzt $\mathcal{T}_1$:= { Kante }, wobei Kante := { Atom, Atom }. Eine Instanz dieses Typs ist beispielsweise $k_1 = \{(1),(2)\}$.

Stufe 2: Die Flächen des Würfels werden als neuer Typ definiert. Es gilt: $\mathcal{T}_2$:= {Fläche} mit Fläche := {Kante, Kante, Kante, Kante, Atom}. Eine Flächeninstanz ist also z. Bsp. $Fl_1 = \{k_1, k_2, k_3, k_4, \text{blau}\}$ mit den Kanten $(k_1, k_2, k_3, k_4) = (\{(1),(8)\}, \{(8),(7)\}, \{(7),(6)\}, \{(6),(1)\})$.

Stufe 3: Letztendlich wird der Würfel selbst als Typ eingeführt:

$\mathcal{T}_3$:= { Würfel } mit Würfel := { Fläche, Fläche, Fläche, Fläche }.

An dieser Stelle wird eine ganz zentrale Problematik deutlich, die im weiteren Verlauf der Arbeit immer wieder angesprochen wird. Es handelt sich dabei um die Frage, wie eine tatsächliche diskrete Struktur in das zugrundeliegende Modell transformiert werden kann, in diesem Fall der gefärbte Würfel in das durch die zuvor angegebenen Definitionen beschriebene Modell. Die Wahl von Schimmel, die Farben in die Grundmenge mit aufzunehmen, ist aus der Definition des Modells verständlich. Es stellt sich jedoch die Frage, ob diese *Eigenschaft* eines Bestandteils der diskreten Struktur - nämlich der Flächen - nicht besser als Eigenschaft im Sinne der in 3.1.1. vorgestellten Bewertung (dom) der Typen verstanden würde.

Wie auch im vorherigen Abschnitt lassen sich die Typzusammenhänge auch hier graphisch darstellen. Für den gefärbten Würfel ergibt sich damit folgender *„Typgraph"*:

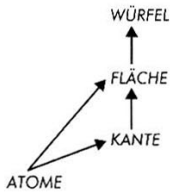


Abbildung 3: *Typgraph des gefärbten Würfels*

Weiterhin zeigt der sogenannte Objektgraph die Abhängigkeiten der Typinstanzen untereinander:

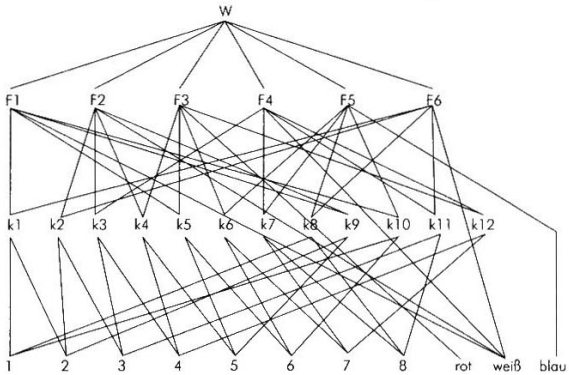


Abbildung 4: *Objektgraph des gefärbten Würfels*

Wertebereiche oder Attribute der unterschiedlichen Typen werden an dieser Stelle noch nicht definiert. Diese Aufgabe übernehmen Klassifikationsfunktionen, die einem Objekt anhand der Typen, aus denen es aufgebaut ist, Klassennummern zuordnen. Implizit werden darin bestimmte Attribute der verschiedenen Type des betrachteten Objekts berücksichtigt. Für die Darstellung des Ansatzes wäre eine explizite Definition entsprechend 3.1.1. jedoch wünschenswert.

Da diese Strukturbeschreibung keine standardisierte Schnittstelle aufweist, die Informationen über die Struktur liefert, verwenden die in [SCH] vorgestellten Algorithmen zur Berechnung der Automorphismengruppe der diskreten Strukturen immer auch Informationen über den genauen Aufbau der unterschiedlichen Typen der diskreten Struktur. Dies bedeutet jedoch auch, daß der Algorithmus bei einer Änderung der Darstellung der diskreten Struktur entsprechend angepaßt werden muß.

Die Implementation der Algorithmen bei Schimmel läßt zudem keine flexible Repräsentation der Objekte zu, was nicht zuletzt die Ausführungsgeschwindigkeit beeinträchtigt.

3.2. Objektorientierte Darstellung

Die hier neu entwickelte, objektorientierte Darstellung bringt vor allem den Vorteil einer exakt definierten Schnittstelle für den Zugriff auf Eigenschaften diskreter Strukturen. Sie erlaubt eine Trennung zwischen der betrachteten Struktur und den Algorithmen, die darauf angewandt werden. So benötigt der Algorithmus selbst keine Informationen mehr über den internen Aufbau der Struktur. Die vorgegebenen Schnittstellen erlauben, falls gewünscht, eine flexible Steuerung der Algorithmen, da an allen wichtigen Stellen im Algorithmus die Möglichkeit besteht, die Entscheidungen mit den Informationen aus der Struktur abzugleichen und somit einen in Abhängigkeit von den vorgegebenen Eigenschaften optimalen Verlauf zu erzielen.

Objektorientierung ist hierbei äquivalent zu einer konsequenten Durchführung eines modularen Konzepts zur Beschreibung von Problemen und dessen Umsetzung bei der Implementation. Moderne Hilfsmittel, wie beispielsweise C++ erleichtern dabei die Umsetzung der für die Lösung des Problems benötigten Algorithmen. Der Vorteil dieser modernen Hilfsmittel ist vor allem eine einfache Formulierung von Zusammenhängen zwischen Daten und den darauf angewandten Funktionen, den Methoden.

Die objektorientierte Darstellung von Strukturen erleichtert damit die Beschreibung der benötigten

Schnittstellen, indem Informationen und die zu ihrer Verarbeitung benötigten Funktionen gemeinsam dargestellt werden können.

Die Aufgabe der Schnittstelle wird in der Implementation von einer abstrakten Klasse übernommen. Dies bedeutet, daß dem Benutzer durch diese Klasse bereits eine vorgefertigte Strukturdefinition zur Verfügung steht. Seine Aufgabe ist nun, in einer davon abgeleiteten, eigenen Klasse, die vorgegebenen Daten und Methoden so zu verändern, daß sie die von ihm gewünschte Funktionalität der betrachteten Struktur wiederspiegeln.

Die folgenden Grafiken zeigen schematisch den Unterschied der beiden Ansätze auf.

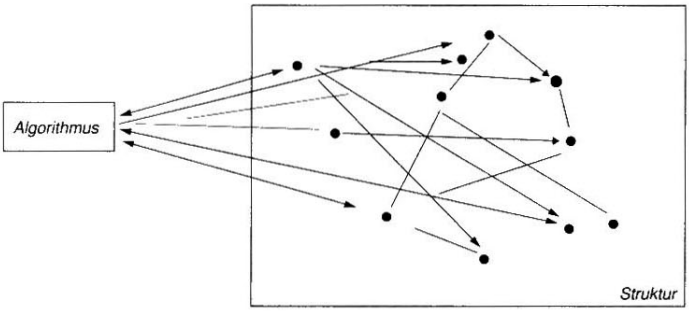


Abbildung 5: *Klassische Darstellung*

In der klassischen Darstellung existiert keine konsequente Trennung zwischen Algorithmus und Datenstruktur. Dies erfordert, daß der Algorithmus sehr speziell an die betrachtete Struktur gebunden ist. Im Beispiel des gefärbten Würfels kann der Algorithmus also ausschließlich auf ein aus Farben, Kanten und Flächen aufgebautes Objekt angewandt werden, wobei die Anordnung derjenigen aus dem Typgraphen entsprechen muß.

Der Zugriff auf die entsprechenden Informationen erfolgt zudem ausschließlich über den Typ- und Objektgraph (vgl. Abbildung 3 und Abbildung 4), wohingegen beim objektorientierten Ansatz Strukturinformationen zu jeder Zeit im Algorithmus vollständig zur Verfügung stehen, also schnell und direkt abgefragt werden können, ohne eine spezielle Typhierarchie berücksichtigen zu müssen. Das nachfolgende objektorientierte Modell besitzt eine entsprechenden Schnittstelle zur Struktur, die damit nicht nur die Formulierung der Algorithmen, sondern auch eine effizientere Verwaltung der Daten im

Rechner ermöglicht:

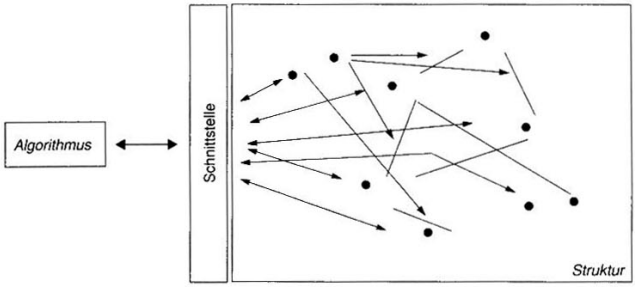


Abbildung 6: *Objektorientierte Darstellung*

Was sich hinter der Schnittstelle aus Abbildung 6, also in der Struktur selbst, verbirgt, bleibt prinzipiell dem Benutzer überlassen. Zwar läßt sich auch der klassische Ansatz weiterhin einbinden, es ist jedoch von Vorteil, eine objektorientierte Modellierung der zu behandelnden Struktur zu verwenden, zumal sie zu einem schlüssigeren Gesamtkonzept führt und die Modellierung der diskreten Struktur unter Berücksichtigung der zur Verfügung stehenden Schnittstellen wesentlich erleichtert.

Literatur zur allgemeinen Theorie der objektorientierten Problemanalyse ist beispielsweise bei [CY], [AH] oder [HS] zu finden.

3.2.1. Strukturaufbau

Die in dieser Arbeit vorgestellte objektorientierte Strukturdarstellung basiert auf einer Definition der Struktur als abstrakte Klasse (vgl. 2.), die bestimmte Methoden (Funktionen) und Attribute (Eigenschaften) vorgibt, die als Schnittstelle zur betrachteten Struktur verwendet werden.

Abstrakte Strukturen besitzen für sich, wie bereits erwähnt, zunächst keine konkreten Eigenschaften oder Funktionalität.

Möchte nun der Leser eine Struktur definieren, die seine reale Problemstellung wiedergibt, so muß er von der vorgegebenen abstrakten Struktur eine neue Struktur ableiten, und die bereits vorgegebenen Schnittstellen so modifizieren, daß sie dem Problem entsprechende, für die betrachteten Fragestellungen (Isomorphie/Ähnlichkeit, ...) relevante Informationen liefern.

Die nachfolgenden Algorithmen basieren auf den Methoden und Attributen, die von der abstrakten Struktur vorgegeben werden, unabhängig von den dort gelieferten Ergebnissen. Diese können vom Benutzer frei gewählt werden und sind in der Regel wiederum abstrakte Datentypen, implementiert als eigene Klasse. Es müssen jedoch auch dort bestimmte Methoden (in der Regel Vergleichsoperatoren) definiert sein.

Legt man diese Schnittstellen zugrunde, so ist für einen Algorithmus nicht mehr notwendigerweise der Aufbau der Struktur selbst von Bedeutung. Die Struktur ist vollständig durch die in der abstrakten Basisstruktur vorgegebenen Schnittstellen definiert.

Welche Methoden und Attribute werden nun zur Definition der Struktur benötigt?

Betrachtet man die Frage der Isomorphie von Strukturen, so benötigen wir zunächst Informationen über diejenigen Elemente der Struktur, die unter Erhaltung der Eigenschaften der Gesamtstruktur aufeinander abgebildet werden sollen.

Diese Elemente bezeichnen wir im folgenden (in Anlehnung an die klassische Strukturdefinition) als

TRÄGER der Struktur.

Da es sich hier um eine objektorientierte Strukturbeschreibung handelt, sind diese Träger natürlich wiederum selbst in einer Klasse realisiert.

Auch für die Träger existiert eine abstrakte Basisklasse, in der die für Träger der Struktur zulässigen Methoden vorgegeben sind. Im wesentlichen ist dies ein Vergleichsoperator, der einen Vergleich unterschiedlicher Trägerinstanzen erlaubt.

Die Struktur selbst liefert demnach Informationen über die *Anzahl* der für das Problem relevanten Trägerinstanzen, und ermöglicht zusätzlich über den Zugriffsoperator „[]" den *Zugriff* auf die einzelnen Trägerinstanzen bzw. deren Objektidentitäten. Der Zugriff erfolgt dabei immer über die Objektidentität der Trägerinstanz, die für den späteren Ablauf der Algorithmen angeordnet werden. Die hier betrachteten Strukturen besitzen eine endliche Anzahl von Trägerinstanzen.
Damit kann man eine Numerierung der Trägerinstanzen vornehmen, so daß alle Trägerinstanzen der Struktur eindeutig durch ihre Numerierung identifiziert werden können, was für die späteren Algorithmen von Bedeutung ist.

Die objektorientierte Darstellung einer Struktur besteht jedoch nicht nur aus Trägern, sondern ist vor allem durch die Beziehungen der Trägerinstanzen untereinander definiert.

Bei molekularen Graphen beispielsweise sind neben Knoten (Trägerinstanzen) auch Kanten auch Entfernungen zwischen Knoten und Winkel zwischen inzidenten Kanten (covalente Bindungen) interessant, also Bindungslängen und auch Bindungswinkel. Diese Eigenschaften sind Bewertungen von komplexen Zusammenhängen, die zwischen Trägerinstanzen bestehen.

Der Benutzer wird in der Regel eigene Objekte verwenden, die diese Beziehungen repräsentieren. Diese Objekte sind wiederum Bestandteil der benutzerdefinierten gesamten Struktur und sind absichtlich darin gekapselt, um dem Benutzer in der Auswahl der von ihm definierten Objekte entsprechende Freiheiten zu geben. Es wird also nicht wie zuvor durch spezielle Konstruktionsoperatoren die interne Organisation der Struktureigenschaften, Beziehungen und Komponenten vorgegeben, sondern die abstrakte Struktur - Basisklasse enthält zusätzliche Methoden, um benötigte Informationen über die Beziehung der Trägerinstanzen untereinander abzufragen. Wie solche Beziehungen im Detail aussehen können, wird im folgenden anhand von Beispielen gezeigt.

Wichtig für das Verständnis dieser Methoden ist, daß in den nachfolgenden Algorithmen immer Abbildungen von Trägerinstanzen aufeinander gesucht werden, wobei die internen Struktureigenschaften unter dieser Abbildung natürlich erhalten bleiben müssen.

Diese objektorientierte Vorstellung einer Struktur legt zugrunde, daß zwei Strukturen identisch sind, wenn alle Beziehungen aller Trägerinstanzen untereinander identisch sind.

In diesem Sinne ist es also möglich, eine Klassifizierung von Trägerinstanzen zu erreichen, indem man ihre Beziehungen bezüglich der vorgegebenen Menge von Trägerinstanzen vergleicht.

Zu diesem Zweck existiert in der abstrakten Basisklasse der Struktur eine Methode *ref*, die zwei Argumente besitzt. Das erste Argument ist eine Liste von Trägerinstanzen, bezüglich derer das zweite Argument, das eine einzelne Trägerinstanz ist, klassifiziert werden soll.

Betrachten wir einen Graphen auf 10 Knoten und die Knotenliste (1,2,3,4,5,7,8,9,10).

Die „Beziehung" des Knoten 6 zu dieser fest vorgegebenen Liste von Knoten in der Gesamtstruktur Graph läßt sich eindeutig durch die sechste Zeile (bzw. Spalte) der Adjazenzmatrix des Graphen darstellen. (Ohne Pos. 6)

In einem dazu isomorphen Graphen müßte das Bild von Knoten 6 zu der Liste von Bildknoten der obigen Liste die gleiche Darstellung bezüglich der Adjazenzmatrix besitzen.

Die Bedeutung dieser *ref* - Methode der Struktur wird später in den entsprechenden Beispielen deutlicher.

Die Beziehungen der Trägerinstanzen von Strukturen unterscheiden sich in Abhängigkeit von den

Struktureigenschaften deutlich.

Betrachtet man molekulare Graphen und darauf die abstrakte Beziehung (hier: Relation) „Kante" zwischen zwei Trägerinstanzen, so kann ein mögliches Attribut die Bindungslänge sein mit dem Wertebereich **R** . Eine anderes Attribut der gleichen Relation ist beispielsweise die Bindungsvielfachheit mit dem Wertebereich **N** .

Da die Algorithmen jedoch unabhängig vom Wertebereich und dessen Ordnung arbeiten sollen, definiert man eine weitere Klasse, welches das Ergebnis dieser Beziehungsabfrage charakterisiert. Einzige Methoden dieser Klasse sind Vergleichsoperatoren, die auch im Algorithmus verwendet werden. Die Vergleichsfunktionen selbst sind also vom Benutzer vorzugeben.

Diese „*rel* - Ergebnis" Basisklasse ist in der programmtechnischen Umsetzung nicht wie die Struktur als abstrakte Basisklasse, sondern als generischer Parameter der abstrakten Basisklasse der Struktur realisiert.

Neben *rel* beinhaltet die Strukturdefinition noch einige weitere Methoden, die jedoch an späterer Stelle erklärt werden.

Das folgende Diagramm zeigt nochmals die eben vorgestellten, für eine Strukturdefinition notwendigen Klassen:

Abbildung 7: *Objektorientierte Strukturdarstellung*

Der Vorteil dieser Darstellung besteht, wie bereits erwähnt, in der allgemeinen Anwendbarkeit und der Unabhängigkeit vom internen Aufbau der Struktur.

Die Algorithmen arbeiten ausschließlich mit obigen Objekten.

Der Benutzer verwendet dann für sein spezifisches Problem angepaßte, von den obigen abstrakten Klasse abgeleitete Klassen bzw. eigene generische Klassen.

Diese Benutzerobjekte enthalten dann natürlich eine Vielzahl von weiteren, problemspezifischen Attributen und Methoden, die jedoch von den hier vorgestellten, darauf angewandten Algorithmen selbst nicht aufgerufen werden. Dies geschieht in der Regel in den Schnittstellen der Trägerinstanzen und *rel*.

Dadurch, daß die Träger und das Ergebnis der *rel* - Methode wiederum abstrakte Datentypen sind, ergibt sich ein maximaler Grad an Kapselung. Der Algorithmus verwendet lediglich die auf dem abstrakten Datentyp TRÄGER und dem Ergebnis der Methode *rel* definierten Vergleichsoperatoren. Die Routinen sind - entsprechend dem objektorientierten Konzept dieses Ansatzes - Bestandteil der Information selbst. Dies ermöglicht auch die Verwendung von „unscharfen" Ergebnissen, was bedeutet, daß auch die Ähnlichkeit von Strukturen geprüft werden kann, wenn die Vergleichsoperatoren auf den Informationen Toleranzen zulassen. Dieser Vorgang bleibt jedoch dem Algorithmus selbst verborgen.

Möchte man beispielsweise diese Strukturen auch in einer Datenbank verwenden, so kann man dies über entsprechende Polymorhismen erreichen, indem das Benutzerobjekt nicht nur von der abstrakten Strukturklasse, sondern zusätzlich von einer abstrakten Datenbankklasse abgeleitet wird, die beispielsweise virtuelle Methoden enthält, durch die eine Persistenz der Objekte gewährleistet werden kann.

3.2.2. Schnittstellendefinition

Im wesentlichen handelt es sich bei den Methoden um die in der Trägerklasse implementierten Vergleichsoperatoren und die vorgestellte Methode *rel* mit dem dazugehörigen rel - Ergebnis und die zusätzlichen, seltener benötigten Methoden Γ und κl, die Informationen über die betrachtete Struktur liefern. Der folgende Abschnitt gibt neben der exakten Definition dieser Methoden auch eine genaue Beschreibung des Wirkungsbereiches bzw. der Aufgaben, die sie erfüllen müssen.

Diese Schnittstelle dient als Grundlage zur Formulierung der Algorithmen zur Ähnlichkeits- und Isomorphieerkennung, also der Beantwortung der Frage, ob Ähnlichkeitsabbildungen oder Isomorphismen existieren.

Um Strukturen in objektorientierter Darstellung zu klassifizieren, ist es notwendig, möglichst viele Eigenschaften der Struktur zu kennen. Dabei unterscheiden wir hier zwischen Eigenschaften, die man aus der isolierten Betrachtung eines einzelnen Trägers erhält und solchen, die einem Träger durch seinen Platz in der Gesamtstruktur zugeschrieben werden können. Die Eigenschaften ergeben sich aus den diese Beziehungen repräsentierenden abstrakten Datentypen und ihren Attributen. Bei einfachen, nicht geschachtelten Strukturen sind dies - analog zur anfangs gegebenen Strukturdefinition - die auf der Trägermenge definierten Relationen und Operationen, aber auch Bewertungen ebendieser.

Die Isomorphie- oder Ähnlichkeitserkennung für Strukturen mit in objektorientierter Darstellung besteht nun daraus, eine bijektive Abbildung φ auf den Objektidentitäten der Trägerinstanzen zweier Strukturen zu ermitteln, so daß alle Eigenschaften der beiden Strukturen unter f übereinstimmen.

Die Vergleichsoperatoren der Trägerinstanzen ermöglichen einen Vergleich der lokalen Attribute. Da eine Trägerinstanz ein Objekt ist, und damit bestimmte Attribute besitzt, können Trägerinstanzen so aufgrund ihrer Attribute angeordnet werden. Repräsentiert das Trägerobjekt - wie beispielsweise bei Molekülen - ein Atom, so ist eine Anordnung von Trägerinstanzen aufgrund der Informationen Atomname und Valenz, die ja Bestandteil des Objektes Atom selbst sind, möglich.

Lokale Eigenschaften beschränken sich dabei durchaus nicht nur auf Eigenschaften wie beispielsweise die Farbe oder den Eckengrad von Knoten. Sie können sich aus der direkten Umgebung ergeben (berechnet werden), so beispielsweise durch eine Liste der Farben oder Knotengrade der benachbarten Atome. Wichtig ist dabei nur, daß eine derartige Liste unabhängig von den übrigen Trägerinstanzen, beispielsweise von der Reihenfolge der benachbarten Knoten, dargestellt wird.

Alle übrigen Eigenschaften, wie die Bindungswinkel bei Molekülen, werden über die Funktion *rel* zur Verfügung gestellt.

Trägerinstanzen können also mit ihren Vergleichsoperatoren isoliert betrachtet werden, wohingegen in der Methode *rel* die Informationen über die übrigen Eigenschaften der Struktur ermittelt werden. Wie diese Umsetzung konkret geschieht, wird im folgenden an verschiedenen Beispielen dargelegt.

Es ist klar, daß die Vergleichsoperatoren und *rel* für unterschiedliche Strukturen, aber auch bei unterschiedlichen Anforderungen an die Isomorphie oder Ähnlichkeit gleicher Strukturen, neu anhand der vorhandenen und relevanten Informationen definiert werden müssen.

Für die Isomorphie- oder Ähnlichkeitserkennung, wie sie in 3.3. für die objektorientierte Darstellung von Strukturen exakt definiert ist, bedeutet dies, daß die Eigenschaften der Trägerinstanzen und das Ergebnis der Methode *rel* für eine Isomorphie- bzw. Ähnlichkeitsabbildung unter den betrachteten Objektidentitäten der Trägerinstanzen übereinstimmen müssen.

Um die in den Algorithmen jeweils aktuelle Aufteilung der Trägerinstanzen darzustellen, wird der Begriff der Klassifizierung eingeführt. In ihr werden Trägerinstanzen (bzw. deren Objektidentitäten), die gleiche Eigenschaften besitzen, wiederum in Klassen zusammengefaßt. Diese Klassen sind jedoch nicht mit der Implementierung eines abstrakten Datentyps zu verwechseln.

Das Ziel aller nachfolgenden Algorithmen ist es, jede Trägerinstanz in einer eigenen Klasse zu speichern und somit eine eindeutige Zuordnung aller Klassen der jeweiligen Klassifizierung der betrachteten Struktur zu ermöglichen.

3.2.2.1. Bezeichnung

Da es sich bei den hier betrachteten Strukturen um *endliche* Strukturen handelt, werden die Trägerinstanzen bzw. deren Objektidentitäten eindeutig numeriert. Die Trägerinstanzen $A_1,...,A_n$ sind deshalb eindeutig durch ihre Indizes 1,...,n identifizierbar und werden im folgenden auch in beiden Formen verwendet.

Wir gehen also im folgenden davon aus, daß die Trägerinstanzen dadurch bereits eine beliebige, aber feste Ausgangsnumerierung besitzen.

Durch den zu verwirklichenden objektorientierten Ansatz bei der Darstellung endlicher Strukturen, sind die Trägerinstanzen nicht isoliert, sondern immer in Verbindung mit der gesamten Struktur zu sehen, deren Bestandteil sie sind. Um dies auch optisch sichtbar zu machen, wird an Stellen, wo die Zugehörigkeit eines Trägerinstanzen zu einer bestimmten Struktur $\mathcal{S}$ mit der Menge S aller Träger von Bedeutung ist, für $A_i \in S$ das Tupel $(\mathcal{S}, A_i)$, oder die kürzere Schreibweise $\mathcal{S}.A_i$ verwandt. Analoges gilt für die noch zu definierenden Schnittstellen der Struktur.

Der objektorientierte Ansatz ermöglicht dadurch eine sehr übersichtliche Darstellung von Objektelementen und Objektfunktionalitäten.

Die abstrakte Klasse, die zur Implementation des abstrakten Datentyps Struktur verwendet wird, wird im folgenden kurz mit $\mathcal{S}$ bezeichnet und auch stellvertretend diskrete Struktur genannt.

3.2.2.2. Definition

Sei $\mathcal{S}$ eine diskrete Struktur mit der Menge S der Trägerinstanzen. Eine *Klassifizierung* $\mathcal{K}$ von $\mathcal{S}$ ist eine geordnete Partition von S. Das heißt

$\mathcal{K} = (\mathcal{K}_1, ..., \mathcal{K}_j)$ mit

$$\mathcal{K}_i \subseteq S\,,\ \mathcal{K}_m \cap \mathcal{K}_n = \emptyset\,,\ \bigcup_{o=1}^{j} \mathcal{K}_o = S \qquad 1 \le i, m, n \le j\,,\ m \ne n$$

Die Elemente der Klassifizierung bezeichnet man als *Klassen*. Mit $|\mathcal{K}|$ bezeichnet man die Anzahl j der Klassen, aus denen die Klassifizierung besteht.

3.2.2.3. Bezeichnung

- Gilt $|\mathcal{K}| = 1$, d. h. $\mathcal{K}_1 = S$, so wird die Klassifizierung als *Ausgangsklassifizierung* bezeichnet.
- Gilt $|\mathcal{K}| = |S|$, müssen alle Klassen einelementig sein und damit liegt jede Trägerinstanz in einer eigenen Klasse. In diesem Fall spricht man von einer *vollständigen* Klassifizierung.

3.2.2.4. Bezeichnung

Sei TRÄGER die abstrakte Basisklasse für einen TRÄGER der Struktur $\mathcal{S}$. In der Klasse Träger müssen Operatoren der Form „<„ und „= =“ definiert sein, die eine Anordnung von Trägerinstanzen erlauben. „= =“ wird hier (in Anlehnung an den C - Syntax) als Gleichheitsoperator für abstrakte Datentypen verwendet. Sind zwei Trägerinstanzen A und B gegeben, so können sie mit Hilfe dieser Methoden verglichen werden. Um zu verdeutlichen, daß sich hinter diesen Vergleichsoperatoren in der Regel komplexe Berechnung verbergen, schreiben wir im folgenden auch $A <_S B$ bzw. $A =_S B$.

3.2.2.5. Bemerkungen

- Die Methoden $<_S$ und $=_S$ werden ausschließlich für die Erzeugung der nachfolgend beschriebenen Vorklassifizierung verwendet. Sie werden im weiteren Verlauf der Algorithmen nicht mehr benötigt.
- Die Attributmenge, welche die Trägerklassen beschreibt, sollte so gewählt werden, daß eine möglichst gute Trennung der Trägerinstanzen bezüglich ihrer Eigenschaften erzielt werden kann.
- Die durch $<_S$ und $=_S$ definierte Ordnung wird für die Bildung einer sogenannten Vorklassifizie-

rung, wie in der nächsten Definition beschrieben, ausgenutzt. Trägerinstanzen mit gleichen Attributen können damit in einer Klassifizierung in gleichen Klassen zusammengefaßt werden.

3.2.2.6. Definition

Sei $\mathcal{S}$ eine diskrete Struktur mit der Menge S der Trägerinstanzen. Unter einer *Vorklassifizierung* $\mathcal{K}^0$ von $\mathcal{S}$ versteht man eine mit Hilfe der obigen Vergleichsoperatoren $<_S$ und $=_S$ erreichte Klassifizierung $\mathcal{K}$ mit den folgenden Eigenschaften:

- $\forall\ 1 \le i \le |\mathcal{K}^0|$ gilt $A_m =_S A_n \qquad \forall\ A_m, A_n \in \mathcal{K}_i^0$
- $\forall\ 1 \le i \le |\mathcal{K}^0| - 1$ gilt $A <_S B \qquad \forall\ A \in \mathcal{K}_i^0, B \in \mathcal{K}_{i+1}^0$

Zur Verdeutlichung dieser Definitionen erläutern wir sie am Beispiel des gefärbten Würfels aus Abbildung 2.

3.2.2.7. Beispiel

Im Gegensatz zu 3.1.2.5. sind die Farben hier nicht Bestandteil der Trägermenge sondern Attribute der Träger. Für die objektorientierte Darstellung der Struktur „gefärbter Würfel" benötigen wir zunächst eine Beschreibung der Träger. Da bei einer Isomorphiebetrachtung eine Abbildung der Knoten des Würfels aufeinander gesucht wird, ist es sinnvoll diese als Träger zu wählen.
Da der objektorientierte Ansatz Attribute und Operationen darauf zusammenfaßt, benötigen wir noch eine Menge von Eigenschaften, die der Träger - also die Knoten - besitzen sollen. Im Hinblick auf die Vorklassifizierung sollten solche Attribute gewählt werden, welche eine möglichst gute Unterscheidung von Objekten des Typs TRÄGER ermöglichen.
Diese Frage ist nicht immer eindeutig zu beantworten. Da in diesem Beispiel eine bei Graphen allgemein übliche Unterscheidung nach dem Attribut Knotengrad zu keinem Ergebnis führen würde (alle Knoten besitzen Grad 3), läßt sich eine Differenzierung der Knoten lediglich durch die Verwendung der Farbinformationen erreichen.
Betrachtet man die klassische Strukturdarstellung, so sind die Farben den Flächen zugeordnet. In der objektorientierten Darstellung besitzt man jedoch die Freiheit, dem Typ TRÄGER Klassifikationsmerkmale von komplexeren Typen zuzuordnen. Es werden beispielsweise später Atome eines Moleküls durch die Eigenschaften der angrenzenden Bindungen klassifiziert (Hybridisierungen). Im vorliegenden Fall bedeutet dies, daß einem Knoten die Farben aller angrenzenden Flächen als Klassifikationsmerkmal zugeordnet werden können. Im nachfolgenden Ansatz wird jeder so entstehenden Farbkombination ein vorgegebener Wert zugeordnet, so daß eine möglichst starke Unterscheidung der Knoten erreicht wird. Es ist jedoch auch denkbar, eine Farbfolge entsprechend den im Uhrzeigersinn angrenzenden Flächen zu definieren, falls die Orientierung des Würfels von Belang ist.

Die Klasse, die in diesem Fall den Träger repräsentiert besitzt also folgende Gestalt:

KNOTEN : TRÄGER
Attribute: Farbwert Methoden: Farbenklasse()Liefert Farbwert zurück) Operator < (KNOTEN, KNOTEN) Operator == (KNOTEN, KNOTEN)

Der Farbwert sei dann wie folgt definiert:

$$\text{Farbwert} := \begin{cases} 0 & \text{nur weiße Flächen grenzen an} \\ 1 & \text{nur weiße und blaue Flächen grenzen an} \\ 2 & \text{nur weiße und rote Flächen grenzen an} \\ 3 & \text{weiße, rote und blaue Flächen grenzen an} \end{cases}$$

Daraus läßt sich die folgende Wertetabelle ableiten:

Knoten	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Farbwert	1	0	0	2	2	3	3	1

Die Methode Farbenklasse() ermöglicht lediglich den Zugriff auf den aktuellen Farbwert einer Knoteninstanz. Die Implementation der Methoden zum Vergleich der Knoten sind dann:

```
KNOTEN :: Operator < ( K1, K2 )
Falls K1.Farbenklasse() < K2.Farbenklasse()
    Gib „WAHR" zurück
Sonst
    Gib „FALSCH" zurück
```

```
KNOTEN :: Operator == ( K1, K2 )
Falls K1.Farbenklasse() = K2.Farbenklasse()
    Gib „WAHR" zurück
Sonst
    Gib „FALSCH" zurück
```

Faßt man nun die Knoten mit gleichen Attributen zusammen, so ergibt sich unter Verwendung von Definition 3.2.2.6. für den gefärbten Würfel die folgende Vorklassifizierung:

$$\mathcal{K}^0 = (\{2,3\},\{1,8\},\{4,5\},\{6,7\})$$

Durch eine Anordnung der Knoten bezüglich der Farben ihrer angrenzenden Flächen kann somit eine nichttriviale Vorklassifizierung erstellt werden.

Andererseits könnte man die Farben jedoch auch als Kantenbewertung definieren, stünden jedoch damit nicht mehr zur Unterscheidung der Trägerinstanzen zur Verfügung, wodurch keine Aufteilung (Verfeinerung) der Ausgangsklassifizierung mehr erreicht werden kann.

Prinzipiell ist bei der Strukturbeschreibung darauf zu achten, daß die Struktur durch die gewählten Klassifikationsmerkmale wie gewünscht identifiziert werden kann. Beispielsweise ist beim gefärbten Würfel die Reihenfolge der benachbarten Farbflächen nicht ausschlaggebend für die Strukturdarstellung. Eine etwas komplexere Struktur könnte diese jedoch zur eindeutigen Unterscheidung benötigen.

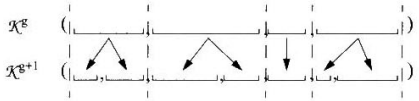
3.2.2.8. Definition

Sei $\mathcal{S}$ eine beliebige diskrete Struktur. Sei $\mathcal{K}^g$ ($g \in \mathbf{N}^+$) eine Klassifizierung von $\mathcal{S}$, wobei $|\mathcal{K}^g| = j$. Unter einer *Verfeinerung* von $\mathcal{K}^g$ versteht man eine Klassifizierung $\mathcal{K}^{g+1}$ mit den folgenden Eigenschaften:

- $\mathcal{K}^{g+1} = (\mathcal{K}_1^{g+1}, \ldots, \mathcal{K}_k^{g+1})$, also $|\mathcal{K}^{g+1}| = k$
- $A_m =_S A_n \qquad \forall\, A_m, A_n \in \mathcal{K}_i^{g+1},\ 1 \le i \le k$
- $A \le_S B \qquad \forall\, A \in \mathcal{K}_i^{g+1}, B \in \mathcal{K}_{i+1}^{g+1},\ 1 \le i \le k-1$
- $k > j$
- $\forall\, \mathcal{K}_i^{g+1}\ \exists!\, \mathcal{K}_j^g : \mathcal{K}_i^{g+1} \subseteq \mathcal{K}_j^g$

Die Verfeinerung stellt also - wie der Name schon sagt - eine weitere, feinere Unterteilung einer bestehenden Klassifizierung dar. Dabei bleibt die durch die Vergleichsoperatoren der Trägerinstanzen auf die Vorklassifizierung ausgeübte Ordnung erhalten. Eine verfeinerte Klassifizierung muß immer mehr Klassen enthalten als die vorherige Klassifizierung.

Die folgende Abbildung verdeutlicht den Zusammenhang zwischen einer Klassifizierung und ihrer Verfeinerung:



Wichtig für die nachfolgende Algorithmen ist vor allem, daß bei einer Verfeinerung immer nur bereits vorhandene Klassen *aufgeteilt* werden, jedoch keine neuen Klassen durch die *Vereinigung* vorhandener Klassen entstehen.

Im Bild wird dies durch die gestrichelten senkrechten Linien verdeutlicht, über deren Grenzen hinweg sich keine Klasse, auch nicht bei weiteren Verfeinerungen, erstreckt.

3.2.2.9. Beispiel

Eine Verfeinerung von $\mathcal{K}^0$ aus Beispiel 3.2.2.7. wäre beispielsweise

$$\mathcal{K}^1 = (\{2,3\},\{1\},\{8\},\{4,5\},\{6,7\})$$

Ziel der späteren Algorithmen zur Isomorphie-/Ähnlichkeitserkennung, kanonischen Numerierung und der Teilstrukturerkennung wird es sein, ausgehend von einer Vorklassifizierung, durch möglichst geschicktes, iteratives Verfeinern der aktuellen Klassifizierung, zu einer vollständigen Klassifizierung zu gelangen.

Dabei liefern die bei der Vorklassifizierung oder vorherigen Verfeinerungen erzeugten einelementigen Klassen wertvolle Informationen für weitere Verfeinerungen.

Wie bereits erwähnt, besitzen die Elemente der Trägermenge eine beliebige, aber feste Ausgangsnumerierung.

Mit Hilfe der Algorithmen zur Isomorphie- oder Ähnlichkeitserkennung wird nun eine Abbildung φ gesucht, welche die Trägerinstanzen zweier Strukturen so aufeinander abbildet, daß ihre Werte bezüglich der lokalen Vergleichsoperatoren der Träger und *rel* übereinstimmen. Übereinstimmen ist dabei immer im Bezug auf die Realisation der Vergleichsoperatoren zu sehen.

Die beschriebenen Algorithmen ermitteln jedoch nicht direkt φ, sondern numerieren die Trägerinstanzen neu. Die Numerierung erfolgt derart, daß, falls beide Strukturen isomorph sind, Element Nr. 1 von Struktur A auf Element Nr. 1 von Struktur B abgebildet werden, Nr. 2 auf Nr. 2 usw., also eine Isomorphie bzw. Ähnlichkeit bezüglich der identischen Abbildung vorhanden ist.

Numeriert man also die Trägerinstanzen S von Struktur $\mathcal{S}$ mit σ_1 und die Trägerinstanzen T von Struktur $\mathcal{T}$ mit σ_2 neu, so ergibt sich anschaulich folgendes Diagramm:

$$\begin{tikzcd} S \arrow[r, "\sigma_1"] \arrow[d, "\varphi"'] & S \arrow[d, "id"] \\ T \arrow[r] & T \end{tikzcd}$$

Die Abbildung φ läßt sich dann nachträglich einfach durch die Komposition $\sigma_1 \cdot \mathrm{id} \cdot \sigma_2^{-1}$ darstellen.

Eine mögliche Neunumerierung σ der Elemente der Trägermenge entsteht durch die Reihenfolge, in der ein Atom als einziges Element einer Klasse der aktuellen Klassifizierung auftaucht. Besteht die

erste im Verlauf der nachfolgend beschriebenen Algorithmen auftretende einelementige Klasse beispielsweise aus Element Nr. 5 der Trägermenge, so wird dadurch an dieser Stelle die Abbildung $\sigma(5) = 1$ festgelegt.

In jedem Schritt der nachfolgenden Algorithmen versucht man nun, so viele einelementige Klassen wie möglich durch geschicktes Ausnutzen der Struktureigenschaften zu erzeugen, um damit die partielle Permutation (Numerierung) σ zu einer vollständigen Numerierung fortzusetzen.

Wurden bereits p Elemente durch σ neu numeriert, so definieren wir im folgenden: $|\sigma| := p$.

Eine durch den oben beschriebenen Vorgang aus einer partiellen Numerierung iterativ erzeugte vollständige Neunumerierung der Trägerinstanzen der betrachteten Struktur bezeichnet man als *Einbettung*. Die Menge alle möglichen Einbettungen wird im folgenden mit $\mathcal{E}$ bezeichnet.

Analog zu den eingangs beschriebenen partiellen Permutationen existieren also auch hier partielle Einbettungen und Fortsetzungen ebensolcher.

Es stellt sich nun die Frage, wie eine weitere Verfeinerung der aktuellen Klassifizierung erreicht werden kann, zumal alle bisherigen Informationen über die betrachtete Struktur aus den Attributen der TRÄGERinstanzen bereits zur Konstruktion der Vorklassifizierung genutzt wurden.

Es muß also eine weitere Schnittstelle geschaffen werden, die zusätzliche Informationen liefert. Da die Informationen, die über eine isolierte Trägerinstanz erhalten werden können, offensichtlich nicht mehr ausreichen, wendet man sich der Gesamtstruktur zu, um - ganz allgemein - Aussagen zu gewinnen, die eine Trägerinstanz als Bestandteil einer komplexeren Umgebung charakterisieren.

Hierfür sind die diese Beziehungen in der Gesamtstruktur beschreibenden Objekte mit ihren Attributen von Bedeutung.

In dem Algorithmus von McKay [MK], der speziell auf Graphen ausgerichtet ist, wurden zu diesem Zweck die Bindungseigenschaften von Knoten herangezogen, also die Bindungsvielfachheit. Mehr zu diesem klassischen Ansatz in Abschnitt 4.1.7.

Schimmel führt zu diesem Zwecke die Klassifikationsfunktionen ein, die in Kombination mit den sog. Auf- und Abwärtschritten von unten durch die gesamte Typhierarchie schrittweise komplexere Informationen aus höheren Typebenen zur Klasseneinteilung einzelner Atome ausnutzen.

Beim objektorientierten Ansatz ist dies prinzipiell ähnlich. Was bei Schimmel die Klassifikationsfunktionen für eine vorgegebene Typhierarchie leisten, ist jedoch hier völlig unabhängig vom Aufbau der betrachteten Struktur.

Dies bedeutet, daß die Elemente einer Klasse der aktuellen Klassifizierung anhand der bisher gefundenen, einelementigen Klassen bewertet werden. Dabei spielt es keine Rolle, ob diese Bewertung nur aufgrund der zuletzt gefundenen Trägerinstanz oder aufgrund der Menge aller bisher festgelegten Trägerinstanzen geschieht.

Betrachtet man die zu Beginn dieses Kapitels gegebene, rein mathematische Strukturdefinition so sind dort analog sowohl ein- als auch mehrstelligen Relationen zugelassen.

Für eine Verfeinerung der einzelnen Klassen liefern die Elementen der zu verfeinernden Klasse, also einzelnen Trägerinstanzen, mit Hilfe der im folgenden beschriebenen *rel* - Funktion in Abhängigkeit von der bisherigen Festlegung einelementiger Klassen eine komplexe Beschreibung aller Zusammenhangskomponenten innerhalb der definierten Struktur.

Dies können sowohl komplexe Datentypen sein, oder deren Objektidentitäten, die Zusammenhänge zwischen den festgelegten Trägerinstanzen darstellen, aber auch einfache Werte wie beispielsweise bei Graphen die Vielfachheit der Bindung einer oder mehrerer bereits festgelegter Knoten.

Dieses breite Einsatzgebiet legt die Realisation des Ergebnisses der Methode *rel* als abstrakten Datentyp nahe. Dadurch können wiederum die entsprechenden Informationen gekapselt werden, so daß der Algorithmus lediglich - wie bei den TRÄGERinstanzen über entsprechende vorgegebene Methoden (Vergleichsoperatoren) darauf zugreifen kann.

Damit können unterschiedliche Trägerinstanzen aufgrund des von der Methode *rel* gelieferten Werte weiter klassifiziert werden. Trägerinstanzen mit gleichem Ergebnis der Methode *rel* bilden dann wiederum neue Klassen, wodurch die gewünschte Verfeinerung der Ausgangsklasse erzielt wird. Für eine genaue Beschreibung dieser Aufteilung sei wiederum auf Kapitel 4. verwiesen.

Die Reduktion der Methode *rel* auf die Betrachtung der Trägerinstanzen ist letztendlich darin begründet, daß auf der untersten Ebene der Isomorphiebetrachtung diskreter Strukturen immer eine Bijektion auf den Trägerinstanzen, unter Berücksichtigung der Struktureigenschaften, gesucht wird.

Dennoch sind die zu klassifizierenden Trägerinstanzen gerade bei der Definition der Methode *rel* nicht losgelöst von der Gesamtstruktur zu sehen. Um es im Sinne der klassischen Strukturdefinition auszudrücken heißt dies, es werden nicht nur die Informationen berücksichtigt, die auf Schachtelungstiefe 0 zur Verfügung stehen, sondern auch solche aus höheren Ebenen. Dadurch erhält man ein detailliertes Bild über die Eigenschaften des betrachteten Atoms in der komplexen Gesamtstruktur.

3.2.2.10. Bezeichnung

Sei REL - ERGEBNIS die abstrakte Basisklasse für den Rückgabewert der nachfolgenden Methode *rel* der Struktur $\mathcal{S}$.

Wie auch in der Klasse TRÄGER müssen in REL - ERGEBNIS Operatoren der Form „<„ und „==“ definiert sein, die einen Vergleich verschiedener Instanzen von REL - ERGEBNIS erlauben. Da diese Vergleichsoperatoren normalerweise komplexe Berechnung kapseln, verwenden wir sie für zwei REL - ERGEBNIS - Instanzen (*rel* - Ergebnisse) R_1 und R_2 nachfolgend immer in der Form $R_1 <_R R_2$ bzw. $R_1 =_R R_2$.

3.2.2.11. Bezeichnung

Sei $\mathcal{S}$ eine diskrete Struktur mit der Menge S der Trägerinstanzen. Sei $\mathcal{E}$ die Menge aller Numerierungen von S. Die Methode *rel* ist Bestandteil der Schnittstelle des abstrakten Datentyps Struktur. Die Methode *rel* besitzt zwei Argumente. Das erste Argument ist eine (partielle) Numerierung der Trägerinstanzen. Das zweite Argument ist eine weitere Trägerinstanz. Das Ergebnis der Methode *rel* wird in einer Instanz der Klasse REL - ERGEBNIS hinterlegt. Dieses Objekt muß die oben erwähnten Vergleichsoperatoren $<_R$ und $=_R$ aufweisen.

Die Methode *rel* liefert durch ihren Aufruf mit der partiellen Numerierung $\sigma \in \mathcal{E}$ und der Trägerinstanz $A \in \mathcal{S}.S$ durch $R = \mathcal{S}.rel(\sigma, A)$ eine Instanz R des REL - ERGEBNISses und damit ein Klassifikationsmerkmal für die Trägerinstanz A unter Berücksichtigung der aktuellen (evtl. partiellen) Einbettung σ.

3.2.2.12. Bemerkung

- Zur Berechnung der *rel* - Funktion können alle Strukturmerkmale herangezogen werden. Je mehr typische Eigenschaften genutzt und in einer Instanz des REL - ERGEBNISses gespeichert werden, desto größer ist der „Verfeinerungseffekt“.
- Die Methode *rel* ist invariant bezüglich Umnumerierungen der Trägerinstanzen.

Wie die *rel* - Funktion zur Verfeinerung der aktuellen Klassifizierung beiträgt, wird gleich anhand eines kurzen Beispiels erklärt. Zunächst folgen jedoch noch die übrigen Schnittstellendefinitionen.

Zusätzlich zu den in den Trägern und durch die *rel* - Methode definierten Klassifikationsmerkmale gibt es noch weitere Methoden, durch welche man den Ablauf der später beschriebenen Algorithmen beeinflussen kann. Durch sie können in den Informationen über die Gesamtstruktur, unter Verwendung der aktuellen Einbettung, berücksichtigt werden. Sie werden jedoch nicht direkt für die Verfeinerung der Klassifizierung verwendet, sondern liefern mögliche Abbruchkriterien für die Algorithmen.

3.2.2.13. Bezeichnung

Sei Γ - ERGEBNIS die abstrakte Basisklasse für den Rückgabewert der nachfolgenden Methode Γ der Struktur $\mathcal{S}$.

Wie zuvor müssen in Γ - ERGEBNIS Operatoren der Form „<" und „==" definiert sein, die einen Vergleich verschiedener Instanzen von Γ - ERGEBNIS erlauben. Auch hier verwenden wir für zwei Γ - ERGEBNIS - Instanzen (Γ - Ergebnisse) G_1 und G_2 nachfolgend immer in der Form $G_1 <_\Gamma G_2$ bzw. $G_1 =_\Gamma G_2$.

3.2.2.14. Bezeichnung

Sei $\mathcal{S}$ eine diskrete Struktur mit der Menge S der Trägerinstanzen. Sei $\mathcal{E}$ die Menge aller Numerierungen von S. Die Methode Γ ist Bestandteil der Schnittstelle des abstrakten Datentyps Struktur. Die Methode Γ besitzt *ein* Argument, nämlich eine (partielle) Numerierung der Trägerinstanzen. Das Ergebnis der Methode Γ wird in einer Instanz der Klasse Γ - ERGEBNIS hinterlegt. Dieses Objekt muß die oben erwähnten Vergleichsoperatoren $<_\Gamma$ und $=_\Gamma$ aufweisen.

Die Methode Γ liefert durch ihren Aufruf mit der partiellen Numerierung $\sigma \in \mathcal{E}$ durch $G = \mathcal{S}.\Gamma(\sigma)$ eine Instanz G des Γ - ERGEBNISses und damit ein Klassifikationsmerkmal für die aktuelle (evtl. partielle) Einbettung σ.

Die Methode Γ ist zusätzlich bezüglich der Numerierung $\sigma \in \mathcal{E}$ fortsetzbar. Dies bedeutet, seien zwei partielle Numerierungen σ_1 und σ_2, sowie zwei entsprechenden Fortsetzungen σ_1^+ und σ_2^+ gegeben, dann muß gelten:

1. $\mathcal{S}.\Gamma(\sigma_1) <_\Gamma \mathcal{S}.\Gamma(\sigma_2) \Rightarrow \mathcal{S}.\Gamma(\sigma_1^+) <_\Gamma \mathcal{S}.\Gamma(\sigma_2^+)$
2. $\mathcal{S}.\Gamma(\sigma_1) >_\Gamma \mathcal{S}.\Gamma(\sigma_2) \Rightarrow \mathcal{S}.\Gamma(\sigma_1^+) >_\Gamma \mathcal{S}.\Gamma(\sigma_2^+)$
3. $\mathcal{S}.\Gamma(\sigma_1^+) =_\Gamma \mathcal{S}.\Gamma(\sigma_2^+) \Rightarrow \mathcal{S}.\Gamma(\sigma_1) =_\Gamma \mathcal{S}.\Gamma(\sigma_2)$

Damit besteht die Möglichkeit, zu jedem Zeitpunkt im Algorithmus eine Bewertung der aktuellen Einbettung aus globaler Sicht vorzunehmen. Die Methode Γ kann - wie übrigens alle anderen Schnittstellen auch - nur optional mit einer Funktionalität hinterlegt werden. Ist eine Struktur ausschließlich durch ihre lokalen Eigenschaften definiert, wie beispielsweise Graphen beim Modell von McKay, kann Γ auch einen beliebiges, für alle Numerierungen konstantes Γ - ERGEBNIS liefern und damit nicht zu einer Verfeinerung der Klassifizierung beitragen.

Andererseits ist es möglich, mit Γ Struktureigenschaften zu beschreiben, die durch lokale Eigenschaften nicht beschrieben werden können.

Beispielsweise können bei dreidimensionalen Strukturen wie Molekülen in räumlicher Darstellung Spiegelungen der Gesamtstruktur oft nicht ausschließlich aufgrund lokaler Winkelbetrachtungen erkannt werden.

Auch die Methode Γ ist invariant bezüglich Umnumerierungen der Trägerinstanzen.

Eine ähnlich definierte Schnittstelle wird zur Berechnung einer kanonischen Numerierung benötigt. Um eine kanonische Form für eine bestimmte Struktur festlegen zu können, muß man beim Vorliegen zweier möglicher partieller Anordnungen der Trägerinstanzen entscheiden, welche Anordnung zu einem kanonischen Repräsentanten führt oder führen kann. Dies geschieht durch die Funktion $\mathcal{KL}$:

3.2.2.15. Bezeichnung

Sei KL - ERGEBNIS die abstrakte Basisklasse für den Rückgabewert der nachfolgenden Methode $\mathcal{KL}$ der Struktur $\mathcal{S}$.

Wie zuvor müssen in KL - ERGEBNIS Operatoren der Form „<„ und „==" definiert sein, die einen Ver-

gleich verschiedener Instanzen von KL - ERGEBNIS erlauben. Auch hier verwenden wir für zwei KL - ERGEBNIS - Instanzen ($\mathcal{kl}$ - Ergebnisse) K_1 und K_2 nachfolgend immer in der Form $K_1 <_K K_2$ bzw. $K_1 =_K K_2$.

3.2.2.16. Bezeichnung

Sei $\mathcal{S}$ eine diskrete Struktur mit der Menge S der Trägerinstanzen. Sei $\mathcal{E}$ die Menge aller Numerierungen von S. Die Methode $\mathcal{kl}$ ist Bestandteil der Schnittstelle des abstrakten Datentyps Struktur. Die Methode $\mathcal{kl}$ besitzt *ein* Argument, nämlich eine (partielle) Numerierung der Trägerinstanzen. Das Ergebnis der Methode $\mathcal{kl}$ wird in einer Instanz der Klasse KL - ERGEBNIS hinterlegt. Dieses Objekt muß die oben erwähnten Vergleichsoperatoren $<_K$ und $=_K$ aufweisen.

Die Methode $\mathcal{kl}$ liefert durch ihren Aufruf mit der partiellen Numerierung $\sigma \in \mathcal{E}$ durch $K = \mathcal{S}.\mathcal{kl}(\sigma)$ eine Instanz K des KL - ERGEBNISses und damit ein Bewertung für die aktuelle (evtl. partielle) Einbettung σ.

Die Methode $\mathcal{kl}$ ist (analog zu Γ) bezüglich der Numerierung $\sigma \in \mathcal{E}$ fortsetzbar und invariant bezüglich Umnumerierungen der Trägerinstanzen.

Durch die Funktion $\mathcal{kl}$ hat man die Möglichkeit, verschiedene Numerierungen der Atome unter Berücksichtigung der Struktureigenschaften zu bewerten.

An dieser Stelle wird aufgrund der unterschiedlichen Einsatzgebiete von Γ und $\mathcal{kl}$ eine Trennung der Funktionen vorgenommen. Bei der eigentlichen Implementierung kann darauf jedoch verzichtet und die Funktionalität der beiden Funktionen zusammengefaßt werden.

3.2.3. Beispiel

Seien die Definitionen aus Beispiel 3.2.2.7. gegeben. Für die Definition von *rel* und $\mathcal{kl}$ werden noch einige Zusatzfunktionen benötigt, die Informationen über Eigenschaften des Würfel liefern.

Ähnlich wie bei der Definition von Graphen lassen sich auch beim gefärbten Würfel Kanten definieren. Die Menge E aller Kanten ist also eine zweistellige Relation auf der Klasse KNOTEN mit

$$E = \{\{1,2\},\{1,8\},\{1,6\},\{2,3\},\{2,5\},\{3,8\},\{3,4\},\{4,5\},\{4,7\},\{5,6\},\{6,7\},\{7,8\}\}$$

Die dazu komplementäre Menge $(V \times V) \setminus E$ ist dann die Menge aller nicht verbundenen Knoten im gefärbten Würfel.

Zusätzliche Informationen über die Relation zweier Atome können noch die Zugehörigkeit zu einer gemeinsamen Fläche oder die folgende, dazu äquivalente Aussage liefern:

3.2.3.1. Definition

Sei G der durch Abbildung 2 beschriebene gefärbte Würfel. Sei S die Menge aller TRÄGERinstanzen, in diesem Fall also der KNOTEN. Die Funktion $\theta : S \times S \to \{0, 1\}$ sei definiert durch

$$\theta(A, B) = \begin{cases} 0 & \text{falls } d(A, B) = 3 \\ 1 & \text{sonst} \end{cases},$$

wobei d der (minimale) Abstand zwischen zwei Knoten ist.

3.2.3.2. Bemerkung

Im betrachteten Würfel liegen sich zwei Punkte, die voneinander den Abstand 3 haben, diagonal gegen-

über. Das heißt:

$$\theta(A, B) = 0 \Leftrightarrow \{A, B\} \in \{\{1, 4\}, \{2, 7\}, \{3, 6\}, \{5, 8\}\}$$

Da die Anzahl der zu einem Knoten benachbarten Kanten oder Flächen für alle Knoten aufgrund der Symmetrie des Würfels identische sind, werden weitere Informationen nicht berücksichtigt.

Darüber hinaus sind die Farbinformationen bereits erschöpfend in die der Vergleichsoperatoren der Klasse KNOTEN eingegangen (vgl. 3.2.2.7.), und würden als zusätzliche Bewertung der obigen Relationen keine Informationen liefern, die zu einer Verfeinerung führen.

3.2.3.3. Bezeichnung

Ein Multigraph G auf n Punkten wird durch eine $n \times n$ Matrix $M = (m_{ij})$ spezifiziert (*Adjazenzmatrix*): Sind die Knoten i und j im Graphen durch eine k-fache Kante verbunden, so gilt $m_{ij}=k$. Gilt $(i, j) \notin E$, so ist $m_{ij}=0$.

Sei A die Adjazenzmatrix des gefärbten Würfels aus Abbildung 2:

$$M := \begin{pmatrix} 0&1&0&0&0&1&0&1 \\ 1&0&1&0&1&0&0&0 \\ 0&1&0&1&0&0&0&1 \\ 0&0&1&0&1&0&1&0 \\ 0&1&0&1&0&1&0&0 \\ 1&0&0&0&1&0&0&1 \\ 0&0&0&1&0&0&0&1 \\ 1&0&1&0&0&1&1&0 \end{pmatrix}$$

Abbildung 8: *Adjazenzmatrix des gefärbten Würfels*

Da es sich hierbei um einen einfachen Graphen handelt, kommen keine Einträge größer zwei vor.

Wie bereits erwähnt, ist bei den nachfolgenden Algorithmen die Reihenfolge, in der die Elemente der Trägermenge einer betrachteten Struktur in einelementige Klassen aufgeteilt werden, von besonderer Bedeutung. Diese Reihenfolge wird in der (partiellen) Einbettung σ festgehalten. Das Element A der Trägermenge, das zuletzt in einer einelementigen Klasse auftauchte, wird in σ auf $|\sigma| + 1$ abgebildet. Mit $|\sigma|$ wird - wie bereits erwähnt - die Anzahl der Elemente der Trägermenge bezeichnet, die bisher eingebettet bzw. neu numeriert wurden.

Nachdem mit der Funktion θ und der Bindungsvielfachheit aus der Adjazenzmatrix nun alle Informationen, die im gefärbten Würfel über zwei beliebige Instanzen der Klasse KNOTEN verfügbar sind, zur Verfügung stehen, kann die Methode *ref* der Struktur, sowie ihr Ergebnis formuliert werden.

3.2.3.4. REL - Ergebnis

Das Ergebnis der *ref* - Methode wird in der Klasse REL - ERGEBNIS gespeichert, das folgende Form besitzt:

REL - ERGEBNIS
Attribute: Konnektivität 3 - Abstand Methoden: Operator < (REL - ERGEBNIS, REL - ERGEBNIS) Operator == (REL - ERGEBNIS, REL - ERGEBNIS)

Die Verwendung der beiden Attribute wird bei der nachfolgenden Definition der *ref* - Methode selbst deutlich. Es handelt sich dabei um nat. Zahlen, so daß die beiden Vergleichsoperatoren wie folgt defi-

niert werden können:

```
REL - ERGEBNIS :: Operator < ( R1, R2 )
---------------------------------------
Falls R1.Konnektivität < R2.Konnektivität
    Gib „WAHR" zurück
Falls R1.Konnektivität = R2.Konnektivität
    Falls R1.3 - Abstand < R2.3 - Abstand
        Gib „WAHR" zurück
Gib „FALSCH" zurück
```

```
REL - ERGEBNIS :: Operator == ( R1, R2 )
----------------------------------------
Falls R1.3 - Abstand = R2.3 - Abstand und
    R1.Konnektivität = R2.Konnektivität
    Gib „WAHR" zurück
Sonst
    Gib „FALSCH" zurück
```

3.2.3.5. rel - Methode

Die Methode *rel* erzeugt eine Instanz der obigen REL - ERGEBNIS - Klasse und belegt die Eigenschaften 3-Abstand und Konnektivität entsprechend den übergebenen Parametern. Sei also M die Adjanzenzmatrix des gefärbten Würfels und $\theta(A, B)$ wie zuvor definiert ($A, B \in S$). Dann ist die Methode *rel* definiert durch:

```
GEFÄRBTER WÜRFEL :: rel(σ, B)
-----------------------------
Erzeuge Instanz R vom Typ REL - ERGEBNIS
Belege R.3-Abstand durch θ(A, B)
Belege R.Konnektivität durch (m_AB)
Gib R zurück
```

wobei A der zuletzt durch die Permutation $A \rightarrow |\sigma| + 1$ in σ aufgenommene Knoten ist und $\sigma \in \mathfrak{T}$, $B \in V$ beliebig.

Mit Hilfe dieser Definition kann man eine Wertetabelle für die Ergebnisse von *rel*, definiert auf dem gefärbten Würfel angeben. In den Spalten sind die Instanzen A der Träger KNOTEN aufgelistet, die zuletzt durch $A \rightarrow |\sigma| + 1$ in σ aufgenommen wurden. In den Reihen stehen die Knoten, für die ein Attribut für die Verfeinerung berechnet werden soll (Knoten B aus obiger Bezeichnung). Die Tabelleneinträge spiegeln die - leicht nachzuprüfende - Symmetrie der *rel* - Funktion bezüglich der Elemente B und A wieder. Für die Darstellung des REL - ERGEBNISses wird ein Tupel (j,k) zur Darstellung der Attribute verwendet, wobei der Wert k dem 3 - Abstand entspricht und j der Konnektivität.

rel		A (aus $A \rightarrow \|\sigma\|$)							
		1	2	3	4	5	6	7	8
	1	(0,1)	(1,1)	(0,1)	(0,0)	(0,1)	(1,1)	(0,1)	(1,1)
	2	(1,1)	(0,1)	(1,1)	(0,1)	(1,1)	(0,1)	(0,0)	(0,1)
	3	(0,1)	(1,1)	(0,1)	(1,1)	(0,1)	(0,0)	(0,1)	(1,1)
B	4	(0,0)	(0,1)	(1,1)	(0,1)	(1,1)	(0,1)	(1,1)	(0,1)
	5	(0,1)	(1,1)	(0,1)	(1,1)	(0,1)	(1,1)	(0,1)	(0,0)
	6	(1,1)	(0,1)	(0,0)	(0,1)	(1,1)	(0,1)	(1,1)	(1,1)

7	(0,1)	(0,0)	(0,1)	(1,1)	(0,1)	(1,1)	(0,1)	(1,1)
8	(1,1)	(0,1)	(1,1)	(0,1)	(0,0)	(1,1)	(1,1)	(0,1)

Um mit Hilfe von *rel* eine Verfeinerung der Ausgangsklassifizierung $\mathcal{K}^0$ zu erzielen, geht man jetzt wie folgt vor: ($\sigma = //$, $|\sigma| = 0$)

1. Wähle eine KNOTENinstanz $A \in S$.
2. Ersetze die aktuelle Einbettung σ durch $\sigma := \sigma \cdot (A \to |\sigma| + 1)$.
3. Für alle Klassen $\mathcal{K} \in \mathcal{K}^0$ bestimme $rel(\sigma, B)$ für alle $B \in \mathcal{K}$ und spalte die Klassen so auf, daß in den neuen Klassen nur Knoten mit gleichem REL - ERGEBNIS liegen.

Ausgehend von der Vorklassifizierung $\mathcal{K}^0 = \{\{2,3\},\{1,8\},\{4,5\},\{6,7\}\}$ wählt man nun beispielsweise A = 8. Die zunächst willkürliche Auswahl dieses Knotens wird im folgenden noch genauer begründet. Sei $\sigma = //$. Ersetze nun σ gemäß 2. durch $\sigma = \begin{pmatrix} 8 & \\ 1 & * \end{pmatrix}$. Damit gilt jetzt $|\sigma| = 1$. Für die Klasse $\mathcal{K}_1^0 = \{2,3\}$ berechnet man dann $rel(\sigma, 2) = (a_{8,3}, \theta(8,3)) = (0,1)$ sowie $rel(\sigma, 3) = (a_{8,2}, \theta(8,2)) = (1,1)$. Diese Werte wurden aus der vorherigen Tabelle aus den grau unterlegten Feldern entnommen. Da $(0,1)$ lexikographisch kleiner als $(1,1)$ ist, wird die Klasse verfeinert, d.h. in zwei neue Klassen $\mathcal{K}_{1_1}^0 = \{2\}$ und $\mathcal{K}_{1_2}^0 = \{3\}$ zerlegt. Analog geht man nun mit den übrigen Klassen vor. Zum Schluß erhält man die Verfeinerung

$$\mathcal{K}^1 = \{\{2\},\{3\},\{1\},\{8\},\{4\},\{5\},\{6,7\}\}$$

6 und 7 haben bezüglich 8 ein identisches REL - ERGEBNIS, so daß hier keine Aufspaltung vorgenommen werden kann. Damit hat man also bereits nach *einem* Verfeinerungsschritt eine fast vollständige Klassifizierung der Knoteninstanzen erhalten. Wie die Wahl des Atoms geschieht, bezüglich dem die oben beschriebene Verfeinerung stattfindet, ist im folgenden Kapitel bei der Vorstellung der entsprechenden Algorithmen beschrieben.

Wurde eine Klasse mit Hilfe der Methode *rel* verfeinert, so kann sich eine weitere Verfeinerung nur dadurch ergeben, daß der aktuellen Einbettung eine neue Knoteninstanz hinzugefügt wird. Dies liegt in diesem Fall daran, daß nur der zuletzt aufgenommene Knoten relevant für eine Beurteilung des Knotens B ist.

Die Methode *rel* kann unter Umständen auch ein für alle Knoten konstantes Ergebnis liefern. Es wird dann keine Verfeinerung der aktuellen Klasse erreicht. Beispiele und Gründe hierfür sind in den Anwendungskapiteln zu finden. Die Algorithmen sind deshalb trotzdem durchführbar. Berechnet man jedoch beispielsweise beim Isomorphietest keinerlei Klassifikationsmerkmale für die betrachteten Atome, also weder unterscheidbare *rel*, noch unterscheidbare Attribute des Knotens selbst, so werden alle Elemente der $S_{|S}$ ausgegeben, da aufgrund der fehlenden Restriktionen offensichtlich alle möglichen Bijektionen zwischen den Trägerinstanzen erlaubt sind.

3.2.3.6. Bemerkung

Die Methode Γ wird für dieses Beispiel nicht benötigt.

Die Methode *kl* wird auf natürliche Weise aus der Adjazenzmatrix des Würfelgraphen abgeleitet.

3.2.3.6.1. Definition

Gegeben sei eine symmetrische $n \times n$ Matrix $M = (m_{ij})$ über $\mathbb{N}$. Die Abbildung Γ bilde die untere Drei-

ecksmatrix gemäß der folgenden Abbildung auf einen Vektor der Länge $\frac{n(n-1)}{2}$ ab:

$$M := \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Damit gilt für a_{ij} :

$$\Upsilon(M) = (1, 0, 1, 0, 0, 1, 0, 1, 0, 1, \ldots, 1)$$

Wird nur die linke untere Teilmatrix der Größe p auf diese Art abgebildet, dann benutzt man die Funktion $\Upsilon_p(M)$. Es gilt dann beispielsweise für Υ_4:

$$\Upsilon_4(M) = (1, 0, 1, 0, 0, 1)$$

3.2.3.6.2. Definition

Seien $V^1 \in \mathbf{N}^m$ und $V^2 \in \mathbf{N}^n$ gegeben. O.B.d.A sei $m \leq n$. Die *lexikographische Ordnung zweier Vektoren beliebiger Länge* ist definiert durch:

$$V^1 = V^2 \Leftrightarrow v_i^1 = v_i^2 \quad \forall\, i \leq m \text{ und } m = n$$

$$V^1 < V^2 \Leftrightarrow \begin{array}{l} 1.\ \exists\, j \leq m \text{ mit } v_i^1 = v_i^2 \ \ \forall\, i < j \text{ und } v_j^1 < v_j^2 \text{ oder} \\ 2.\ v_i^1 = v_i^2 \ \ \forall\, i \leq m \text{ und } m < n \end{array}$$

$$V^1 > V^2 \Leftrightarrow \exists\, j \leq m \text{ mit } v_i^1 = v_i^2 \ \ \forall\, i < j \text{ und } v_j^1 > v_j^2$$

Sei σ eine partielle Einbettung der Trägerinstanzen, wobei bereits p (= $|\sigma|$) Festlegungen bekannt sind. Die Anwendung von σ auf die Adjazenzmatrix M wird definiert durch:

$$\sigma(M) = M^{\sigma} := (m_{\sigma^{-1}(i)\sigma^{-1}(j)})_{1 \leq i,j \leq p}$$

Es gilt $M^{\sigma} \in \mathbf{N}^{p \times p}$.

3.2.3.6.3. KL - Ergebnis

Die Klasse KL - ERGEBNIS enthält als Attribut einen Vektor beliebiger Länge und die Vergleichsoperatoren eine Implementation der in 3.2.3.6.2. darauf definierten Ordnung. Eine Beschreibung dieses abstrakten Datentyps liefert:

KL - ERGEBNIS
Attribute: MatrixVektor Länge Methoden: Operator < (KL - ERGEBNIS, KL - ERGEBNIS) Operator == (KL - ERGEBNIS, KL - ERGEBNIS)

Die Implementation der Vergleichsoperatoren ist entsprechend 3.2.3.6.2. realisiert.

3.2.3.6.4. kl - Methode

Die Methode *kl* erzeugt eine Instanz der obigen KL - ERGEBNIS - Klasse und belegt die Eigenschaften MatrixVektor und Länge entsprechend den übergebenen Parametern. Sei M die Adjazenzmatrix aus

Abbildung 8 mit $M \in \mathbb{N}^{8 \times 8}$. $\mathcal{E}$ sei die Menge aller Einbettungen. Eine partielle Einbettung habe die Länge p<8. Die Bewertungsfunktion $\mathcal{kl}$ ist dann definiert durch:

GEFÄRBTER WÜRFEL :: $k\mathcal{l}(\sigma)$
Erzeuge Instanz K vom Typ KL - ERGEBNIS Belege K.MatrixVektor durch $\Upsilon(\sigma(M))$ $(= \Upsilon_p(\sigma(M)))$ Belege K.Länge durch $\frac{n(n-1)}{2}$ (mit $\lvert\sigma\rvert = n$) Gib K zurück

Die Matrix M wird in der Regel wiederum durch ein Objekt vom Typ MATRIX realisiert sein.

Sei $\sigma \in \mathcal{P}$ also beispielsweise definiert durch $\sigma = \begin{pmatrix} 3 & 1 & 5 & 7 \\ 1 & 2 & 3 & 4 \end{pmatrix}^*$. Dann gilt für die Adjazenzmatrix M aus 3.2.3.6.1.:

$$M^{\sigma} := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

und damit $\Upsilon(\sigma(M)) = (0, 0, 0, 0, 0, 0)$:

Im Vergleich dazu gilt für $\mu = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}$: $\Upsilon(\sigma(M)) = (1, 0, 1, 0, 0, 1)$

und somit $\mathcal{kl}(\sigma) <_K \mathcal{kl}(\mu)$. Die durch μ definierte Anordnung der Atome ist also in einem gewissen Sinne „besser" (weil $\mathcal{kl}(\mu)$ größer als $\mathcal{kl}(\sigma)$), als die durch σ induzierte.

Damit sind nun alle zuvor beschriebenen, notwendigen Schnittstellen zur diskreten Struktur „gefärbter Würfel" definiert. Die letzte Schnittstelle, beschrieben durch die Methode $\mathcal{kl}$, wird lediglich bei der Berechnung einer kanonischen Numerierung verwendet und muß für die Isomorphie- oder Ähnlichkeitserkennung nicht notwendigerweise vorhanden sein.

3.3. Ähnlichkeit

Benutzt man die zuvor definierte Schnittstelle zur Darstellung der Strukturen, so kann mit ihrer Hilfe die Definition der Ähnlichkeit von Strukturen mit objektorientierter Darstellung vorgenommen werden.

Dazu ist es notwendig zu verstehen, welche Informationen und Aufgaben die zuvor definierten Schnittstellen erfüllen.

Hinter die abstrakten Klasse TRÄGER verbergen sich, wie bereits erwähnt, diejenigen Elemente der betrachteten Struktur, die bei einem Isomorphietest zweier Strukturen so aufeinander abgebildet werden müssen, daß sie zum einen bezüglich ihrer internen Vergleichsoperatoren „$<_T$" und „$=_T$" übereinstimmen, aber auch *rel* und Γ bezüglich ihrer Bilder, und bezüglich der auf den im REL - und Γ - ERGEBNIS definierten Vergleichsoperatoren, übereinstimmen.

Die Methode $\mathcal{kl}$ wird ausschließlich bei der kanonischen Numerierung benötigt und spielt für die Ähnlichkeitsbetrachtung zunächst keine Rolle.

Im Verlauf der Arbeit werden wir auch Strukturen betrachten, bei denen bestimmte Attribute oder Beziehungen lediglich „bis auf eine gewisse Toleranz" übereinstimmen müssen. Da die Überprüfung aller Struktureigenschaften in Vergleichsoperatoren gekapselt ist, können solche „Tests bezüglich Toleranzen" sehr leicht realisiert werden.

Die in den zuvor erklärten Operatoren $=_S$, $=_R$ und $=_G$ ausgeführten Vergleiche stehen dann nicht

mehr für Identität im eigentlichen Sinne, sondern sind eher unter dem Gesichtpunkt zu betrachten, daß zwei Objekte „innerhalb bestimmter Vorgaben" übereinstimmen.

Als Beispiel, das auch später noch detailliert beschrieben wird, sind hier wiederum Molekülstrukturen zu nennen, bei denen in räumlichen Plazierungen Toleranzen in der Darstellung - der Bindungs- oder Tetraederwinkeln oder der Bindungslängen - ausdrücklich erlaubt werden sollen.

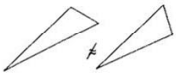
Eine Abbildung zwischen den Trägern zweier Strukturen, für die eine Übereinstimmung innerhalb der in den Vergleichsfunktionen definierten Toleranzen erreicht wird, nennt man Ähnlichkeitsabbildung.

Will man jedoch deutlich machen, daß keine Toleranzen erlaubt sind, so verwendet man im folgenden statt der Operatoren $=_S$, $=_R$ und $=_G$, eine Darstellung bzgl. „=". Ähnlichkeitsabbildungen, bei denen keine Toleranz erlaubt ist, bezeichnet man auch als Isomorphismen.

Ähnlichkeitsabbildungen spielen immer dort eine große Rolle, wo nichtdiskrete reale Probleme durch diskrete Modelle approximiert werden.

Bei der Ähnlichkeitsabbildung handelt es sich im Grunde um eine Klassifizierung von Strukturen. Im Gegensatz zum Isomorphismus ist sie i. a. jedoch keine Äquivalenzrelation, da gewisse Toleranzen absichtlich zugelassen werden. Am besten vergegenwärtigt man sich diese Eigenschaft an folgendem Beispiel:

Alle benachbarten Dreiecke sind ähnlich zueinander, wenn in jedem Schritt eine vorgegebene Toleranz akzeptiert wird. Man wird jedoch wohl kaum das erste und letzte Dreieck als ähnlich bezeichnen:



Damit ist es jedoch durchaus auch möglich, daß eine Struktur ähnlich zu zwei Strukturen ist, die wiederum nicht ähnlich zueinander sind. So könnte das mittlere Dreieck aus der obigen Serie sowohl zum ersten als auch zum letzten ähnlich sein.

Es besteht eine enge Verwandtschaft mit dem Begriff der Fuzzy - Morphismen, die bei der Beschreibung von Fuzzy - Mengen und - Logic verwendet werden. (siehe [KY] oder [BG]) Eine Anwendung der dort beschriebenen Theorie wird in Abschnitt 6.3.2.2.3. bei der Abstandsbetrachtung bei Molekülen beschrieben. Das mögliche Einsatzgebiet der unscharfen Mengen beschränkt sich jedoch ausschließlich auf die Beschreibungen der Struktureigenschaften. Eine Ähnlichkeitsabbildung φ bildet lediglich die Elemente der Trägermenge zweier Strukturen aufeinander ab. Ein Grad der Ähnlichkeit zweier Strukturen kann deshalb nur aufgrund der Aussage erfolgen, wie „gut" die Ergebnisse in den Vergleichsmethoden mit den maximal vorgegebenen Toleranzgrenzen übereinstimmen.

3.3.1. Definition

Die Struktur S_1 sei gemäß der zuvor definierten Schnittstelle definiert durch:

- Die Klassen TRÄGER, REL - ERGEBNIS und Γ - ERGEBNIS.
- Die Menge der Instanzen $S_1.S$ vom abstrakten Typ TRÄGER
- Die Methoden $S_1.rel$ und $S_1.\Gamma$.

Die Struktur S_2 sei gemäß der zuvor definierten Schnittstelle definiert durch:

- Klassen TRÄGER, REL - ERGEBNIS und Γ - ERGEBNIS.
- Die Menge der Instanzen $S_2.S$ vom abstrakten Typ TRÄGER

- Die Methoden $\mathcal{S}_2.rel$ und $\mathcal{S}_2.\Gamma$.

Die Funktion φ, definiert durch die Abbildungen $\varphi : \mathcal{S}_1.S \rightarrow \mathcal{S}_2.S$, ist genau dann eine *Ähnlichkeitsabbildung* von $\mathcal{S}_1$ auf $\mathcal{S}_2$, wenn für alle $A, B \in \mathcal{S}_1.S$ gilt:

ISO1 φ ist bijektiv

ISO2 $A =_S \varphi(A)$

ISO3 $\mathcal{S}_1.rel(\mathrm{id}, B) =_R \mathcal{S}_2.rel(\varphi, \varphi(B))$

ISO4 $\mathcal{S}_1.\Gamma(\mathrm{id}) =_G \mathcal{S}_2.\Gamma(\varphi)$

Durch die Operatoren $=_S$, $=_R$ und $=_G$ wird ein Ähnlichkeitsmaß für die betrachtete Struktur definiert. Existiert eine Ähnlichkeitsabbildung $\varphi : \mathcal{S}_1.S \rightarrow \mathcal{S}_2.S$ auf den Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$, so sind $\mathcal{S}_1$ und $\mathcal{S}_2$ *ähnlich* ($\mathcal{S}_1 \approx \mathcal{S}_2$).

3.3.2. Bemerkung

- Sind die Bedingung **ISO2, ISO3** und **ISO4** mit „=" erfüllt, sind also keine Toleranzen innerhalb der Vergleichsoperatoren erlaubt, so sind $\mathcal{S}_1$ und $\mathcal{S}_2$ isomorph. ($\mathcal{S}_1 \cong \mathcal{S}_2$)
- Isomorphismen φ von $\mathcal{S}_1$ auf $\mathcal{S}_1$ bezeichnet man als Automorphismen.
- Implementationen objektorientierter Strukturen verwenden natürlich die von den abstrakten Klassen TRÄGER, REL - ERGEBNIS und Γ - ERGEBNIS abgeleiteten Klassen, die in der Definition von $\mathcal{S}_1$ und $\mathcal{S}_2$ nicht notwendigerweise übereinstimmen müssen (siehe 3.3.7.).

3.3.3. Satz

Die Automorphismen bilden in Verbindung mit der Hintereinanderausführung eine Gruppe, die sog. Automorphismengruppe.

3.3.4. Bemerkung

- Gilt $\mathcal{S} \cong \mathcal{T}$ und $\mathcal{T} \cong \mathcal{U}$, dann gilt auch $\mathcal{S} \cong \mathcal{U}$. (Transitivität)
- Satz 3.3.3. gilt nicht für Ähnlichkeitsabbildungen, da keine Transitivität vorliegt.

Notwendige Voraussetzung für die Ähnlichkeit oder auch Isomorphie zweier Strukturen ist also, entsprechend der vorherigen Definition, die Existenz einer bijektiven Abbildung φ, welche die Trägerinstanzen zweier Strukturen unter Erhaltung ihrer Eigenschaften aufeinander abbildet. Liegt eine derartige Abbildung vor, so muß man überprüfen, ob alle durch die Methoden *rel* und Γ gekapselten Eigenschaften identisch oder ähnlich sind mit denjenigen der Bildmenge unter φ. Bei Identität dieser Werte liegt ein Isomorphismus vor, bei einer Toleranz innerhalb vorgegebener Grenzwerte spricht man von Ähnlichkeit.

Es folgt nun eine Neuformulierung der obigen Definition im Hinblick auf die später vorgestellten Algorithmen. Dort versucht man nicht direkt eine Abbildung φ mit diesen Eigenschaften zu finden, sondern erreicht durch eine Umnumerierung der Trägerinstanzen in den beiden betrachteten Strukturen, daß die Identität diese Ähnlichkeitsabbildung darstellt.

3.3.5. Satz

Gegeben seien zwei Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ mit den Voraussetzungen aus Definition 3.3.1.. Seien weiterhin $\sigma_1 \in \mathcal{P}_{|\mathcal{S}_1.S|} \subseteq S_{|\mathcal{S}_1.S|}$ eine Umnumerierung von $\mathcal{S}_1.S$ und analog σ_2 eine Umnumerierung von $\mathcal{S}_2.S$ und eine Abbildung $\varphi : \mathcal{S}_1.S \rightarrow \mathcal{S}_2.S$ mit $\sigma_2^{-1} \circ \sigma_1 = \varphi$ gegeben. Dann gilt: φ ist genau dann eine

Ähnlichkeitsabbildung zwischen S_1 und S_2, wenn die Identität Ähnlichkeitsabbildung zwischen den durch σ_1 und σ_2 neu numerierten Strukturen ist.

Beweis:

Die Zusammenhänge zwischen den Abbildungen gibt das folgende Diagramm wieder:

$$\begin{tikzcd} S_1.S \arrow[r, "\sigma_1"] \arrow[d, "\varphi"'] & S_1.S \arrow[d, "id"] \\ & \\ \arrow[r, "\sigma_2"'] & \end{tikzcd}$$

Damit ist anschaulich klar, daß eine Ähnlichkeit zwischen den beiden Strukturen S_1 und S_2 auch eine Ähnlichkeit zwischen den beiden Strukturen mit den durch σ_1 und σ_2 neu numerierten Trägermengen zur Folge hat, und umgekehrt.

Sei also φ als Ähnlichkeitsabbildung von S_1 auf S_2 gegeben. Dann ist aufgrund der Invarianz von *ref* und Γ bezüglich der Anwendung einer Umnumerierung auch $\sigma_2 \circ \varphi \circ \sigma_1^{-1}$ eine Ähnlichkeitsabbildung zwischen den durch σ_1 und σ_2 neu numerierten Strukturen S_1 und S_2. Aus der Voraussetzung $\sigma_2^{-1} \circ \sigma_1 = \varphi$ folgt damit sofort die Behauptung.

Sei umgekehrt id als Ähnlichkeitsabbildung zwischen den durch σ_1 und σ_2 neu numerierten Strukturen S_1 und S_2 gegeben. Dann ist aufgrund der Invarianz von *ref* und Γ bezüglich der Anwendung einer Umnumerierung $\sigma_2^{-1} \circ \sigma_1$ eine Ähnlichkeitsabbildung zwischen den Strukturen S_1 und S_2, woraus unter Anwendung der Voraussetzung wiederum die Behauptung folgt.

Durch die Anwendung von Umnumerierungen der Form $\sigma_2^{-1} \circ \sigma_1 = \varphi$ kann also - die Existenz eines Isomorphismus φ vorausgesetzt - die identische Abbildung als Ähnlichkeitsabbildung festgelegt werden.

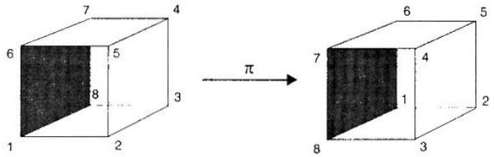
Die nachfolgenden Algorithmen, insbesondere die kanonischen Numerierung nutzen diese Eigenschaft der Ähnlichkeit aus.

3.3.6. Beispiel

Betrachten wir den gefärbten Würfel aus Beispiel 3.2.3.. In diesem Beispiel beschränken wir uns auf die Isomorphie von Strukturen. Beispiele, in denen eine echte Ähnlichkeit benutzt wird, folgen später. Der einzige zulässige nichttriviale Automorphismus lautet:

$$\varphi = / \ 8 \ 3 \ 2 \ 5 \ 4 \ 7 \ 6 \ 1 \ /$$

Anschaulich ändert sich die Numerierung der Eckpunkte dadurch wie folgt:



Um zu beweisen, daß dies (neben der Identität) die einzig mögliche Umnumerierung ist, genügt es, die Bedingung $A =_S \varphi(A)$ zu betrachten. Die in 3.2.2.7. aufgestellte Wertetabelle der Farbwerte und die daraus resultierende Sortierung gibt die möglichen Kandidaten für eine Permutation vor. Knoten 1 kann also lediglich auf Knoten 8 oder sich selbst abgebildet werden, da nur dann die obige Bedingung erfüllt

ist. Analog dazu gibt es für jeden weiteren Knoten nur eine mögliche Abbildung neben der Identität. Es ist leicht einzusehen, daß in diesem Beispiel jedoch die Festlegung der ersten Permutation die übrigen Permutationen eindeutig festlegt. Ist die Permutation $1 \rightarrow 8$ festgelegt, so muß eine geeignetes Bild für Knoten Nr. 2 gefunden werden. Um weiterhin die Isomorphiebedingungen zu erfüllen muß gelten: $rel\left(\left(\begin{smallmatrix}1\\1\end{smallmatrix}*\right),2\right) =_R rel\left(\left(\begin{smallmatrix}8\\1\end{smallmatrix}*\right),A\right)$, wobei $A \in \{2,3\}$, da nach 3.2.2.7. Knoten 2 entweder auf sich selbst oder auf Knoten 3 abgebildet werden kann. Das bedeutet, das Bild von Knoten 2 unter dem Automorphismus muß eine identische Bewertung aufzeigen wie die Bewertung der Relation (1,2) in der ursprünglich numerierten Struktur.

Nach der Wertetabelle für *rel* aus 3.2.3.5. gilt $rel\left(\left(\begin{smallmatrix}1\\1\end{smallmatrix}*\right),2\right) =_R (1,0)$. Da $rel\left(\left(\begin{smallmatrix}8\\1\end{smallmatrix}*\right),3\right) = (1,0)$ und $rel\left(\left(\begin{smallmatrix}8\\1\end{smallmatrix}*\right),2\right) = (0,0)$ muß Knoten 2 auf Knoten 3 abgebildet werden, Analog dazu werden alle anderen Permutationen festgelegt.

3.3.7. Ähnlichkeit bei Modellwechsel

Durch die Kapselung der Vergleichsoperatoren auf den Trägerinstanzen, dem REL - ERGEBNIS und dem Γ - ERGEBNIS ist es auch möglich, Strukturen mit unterschiedlichen zugrundeliegenden mathematischen Modellen auf Ähnlichkeit bzw. Isomorphie zu testen.

So ist es beispielsweise denkbar, daß in den betrachteten Strukturen Farben als Bewertungen eingesetzt werden. Es kann beispielsweise sein, daß diese in einer Struktur im RGB - Modell vorliegen, in der anderen jedoch als CMYK - Werte. Über die bekannte Transformation dieser Farbmodelle ist dann ein Vergleich möglich. Die Transformation ist jedoch in einer Operatorfunktion gekapselt.

3.3.7.1. Beispiel

In Struktur S_1 wird eine von der abstrakten Basisklasse TRÄGER abgeleitete Klasse KNOTEN_RGB verwendet. In der Struktur S_2 eine von der abstrakten Basisklasse TRÄGER abgeleitete Klasse KNOTEN_CMYK.

In der Klasse KNOTEN_RGB sei die Operatorfunktion „==" wie folgt implementiert:

```
KNOTEN_RGB :: Operator == ( T1, T2 )
---------------------------------------
Falls Typ(T2) == KNOTEN_CMYK
        Gib VGL_RGB_CMYK ( T1.Farbe, T2.Farbe ) zurück
Sonst
        Gib ( T1.Farbe == T2.Farbe ) zurück
```

Dabei ist vorausgesetzt, daß in beiden Strukturen ein Attribut FARBE zur Speicherung des Farbwertes eines Knotens verwendet wird, und die Methode VGL_RGB_CMYK zum Vergleich eines RGB und CMYK - Wertes existiert.

Läßt man also zu, daß unterschiedliche, von einer gemeinsamen Basisklasse abgeleitete Klassen in der Strukturdarstellung verwendet werden können, so ist es beispielsweise möglich, eine Ähnlichkeit zwischen Graphen und Molekülen zu definieren. Daß dabei natürlich der kleinste gemeinsame Nenner von Eigenschaften, in diesem Fall des Graphen, zugrunde liegen muß, ist klar.

Denkt man an die Anwendung in einer mathematischen Datenbank, so macht es jedoch durchaus Sinn, solche Vergleiche durchzuführen.

Das folgende Schema verdeutlicht die Vorgehensweise.

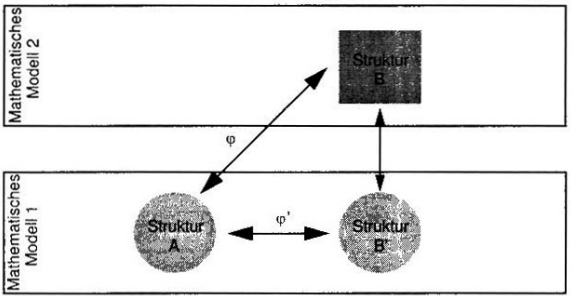


Abbildung 9: *Ähnlichkeit bei Modellwechsel*

Die Funktionalität, die für die Transformationen zwischen Modell 1 und Modell 2 notwendig ist, wird dabei in entsprechenden Methoden gekapselt und bleibt somit für die verwendeten Algorithmen unsichtbar.

Natürlich kann man Struktur B auch zuerst in Struktur B' umwandeln, indem diese Funktionalität explizit vor Beginn eines Isomorphietests angewandt wird. Sind zwei Strukturen mit unterschiedlichen zugrundeliegenden mathematischen Modellen jedoch nicht ähnlich, so kann ein Ähnlichkeitstest unter Ausnutzung des direkten Wegs (Struktur A ↔ Struktur B) deutliche Geschwindigkeitsvorteile bieten. Wird nämlich bereits im Anfangsstadium der Algorithmen ein Unterschied zwischen den beiden Strukturen festgestellt, so wird der Algorithmus sofort abgebrochen. Beim indirekten Weg (Struktur A ↔ Struktur B' ↔ Struktur B) wäre zu diesem Zeitpunkt bereits der Aufwand zur Konvertierung angefallen.

Diese Möglichkeit des direkten Vergleichs ist bei der klassischen Darstellung endlicher Strukturen nicht vorgesehen.

4. Algorithmen

Die in dieser Arbeit entwickelten Algorithmen behandeln auf der Ebene von diskreten Strukturen zwei wesentliche Fragestellungen, die bei realen Objekten von großem Interesse sind.

- *Es soll geprüft werden, ob zwei vorliegende reale Objekte isomorph bzw. ähnlich sind.*
- *Es soll geprüft werden, ob ein vorliegendes Objekt Bestandteil eines anderen ist.*
- *Die Objekte sollen in einer Datenbank durch direkten Zugriff oder Suche abrufbar sein.*

Die erste Fragestellung enthält bei der vorgestellten objektorientierten Darstellung diskreter Strukturen die Lösung des Isomorphieproblems. Eine entscheidende Rolle spielt dabei die geeignete Diskretisierung der Struktur und die Definition des dazugehörigen Isomorphie- bzw. Ähnlichkeitsbegriffes. Durch die Hinzunahme eines zusätzlichen Strukturmerkmals kann dabei in der Regel eine sehr viel effizientere Lösung des Isomorphieproblems erreicht werden als durch einen bis ins letzte Detail optimierten Algorithmus. Voraussetzung dafür ist jedoch ein Algorithmus, der die Strukturbeschreibung berücksichtigt, das heißt je genauer die Beschreibung, bzw. je mehr Möglichkeiten zur Unterscheidung von Strukturen existieren, desto größer ist die Effizienz des Algorithmus. Eng verwandt mit der Bestimmung der Isomorphie zwischen zwei diskreten Strukturen ist die Ermittlung der Automorphismengruppe einer Struktur. Dies ist die Menge aller Umnumerierungen der Elemente einer Struktur, die durchgeführt werden können, so daß die Struktur selbst jedoch invariant bleibt.

Neben der Vollstrukturisomorphie ist in vielen Bereichen auch die Frage einer Teilstrukturisomorphie von Interesse. In der Chemie entspricht das der Substruktursuche. Gerade bei dieser praktischen Anwendung gibt es zwar sehr spezielle Verfahren, die schnell - im Rahmen der bekannten Informationen - zu einem Ergebnis führen (siehe Kapitel 7.). Ändert sich jedoch die Strukturbeschreibung nur geringfügig, beispielsweise durch neue Erkenntnisse, die durchaus die Suche erleichtern könnten, oder ändert sich sogar die Fragestellung, beispielsweise von der Frage einer topologischen zu einer 3D Substruktursuche, so ist eine *einfache* Änderung nicht mehr möglich. Es ist also sinnvoll, hier ein *allgemeines* Hilfsmittel zur Isomorphieerkennung für *allgemeine diskrete Strukturen* zur Verfügung zu stellen, das leicht an geänderte Problemstellungen *und* Strukturbeschreibungen adaptierbar ist.

Die letzte Fragestellung einer effizienten Verwaltung von Strukturen führt zu der Frage einer einheitlichen Repräsentation, d. h. zur Berechnung eines kanonischen Repräsentanten. Auch hier soll ein Algorithmus verwendet werden, der in hohem Maße die Eigenschaften der diskreten Strukturen ausnutzt, um schnell und effizient zum Ziel zu führen. Ausgehend von einem kanonischen Repräsentanten ist es dann möglich, vorzugsweise unter Verwendung einer schnellen Hashverwaltung, einen *direkten* Zugriff auf Daten zu ermöglichen. Gerade in der Chemie sind sehr große Datenmengen mit weit mehr als 1 Mio Strukturen zu verwalten, und es ist davon auszugehen, daß sich dieser Wert in Zukunft noch weiter erhöht.

Betrachten wir zunächst jedoch das Problem der Erkennung der Ähnlichkeit zweier Strukturen.

4.1. Ähnlichkeitserkennung

Der im folgenden beschriebene Algorithmus zur Ähnlichkeitserkennung ist eine Erweiterung der Arbeit von [MK]. Im Gegensatz zu [MK] wird jedoch der ausschließlich für die Isomorphie von Graphen formulierte Algorithmus durch die Verwendung der im vorherigen Kapitel definierten allgemeine objektorientierte Darstellung endlicher bewerteter Strukturen wesentlich erweitert, wie in Kapitel 4.1.7. näher erklärt. Dies unterscheidet diesen Ansatz auch ganz wesentlich von [JL], wobei der Grundtyp dieser Algorithmen, nämlich durch geeignetes Partitionieren der Trägerinstanzen die Zahl der maximal zu berechnenden Permutationen zu verringern, durchaus Ähnlichkeiten aufweist. Daneben werden jedoch an entscheidenden Stellen Änderungen vorgenommen, die eine deutlich effektivere Ausnutzung aller über die betrachtete Struktur verfügbaren Informationen gewährleisten.

Beschränken wir uns in der folgenden Einführung zunächst auf den Spezialfall der Isomorphiebestimmung.

Die Bestimmung der Automorphismengruppe einer endlichen Struktur ist äquivalent zum Problem der Bestimmung der Isomorphie zweier diskreter Strukturen, indem man zwei identische Strukturen betrachtet und alle Möglichkeiten berechnet, Struktur S_1 in Struktur S_2 bzw. in diesem Fall in sich selbst einzubetten. Für die Beschreibung des folgende Algorithmus geht man deshalb anfänglich davon aus, daß zwei beliebige diskrete Strukturen betrachtet werden. Die Aufgabe des Algorithmus ist festzustellen, ob die beiden Strukturen isomorph sind und gegebenenfalls alle Isomorphismen zwischen den Strukturen zu liefern. Sind die beiden betrachteten Strukturen identisch, so liefert der Algorithmus damit die gewünschte Automorphismengruppe.

4.1.1. Beispiel

Lassen sich die Knoten des zweiten Graphen aus dem folgenden Bild so umordnen, daß der resultierende Graph dem ersten entspricht?

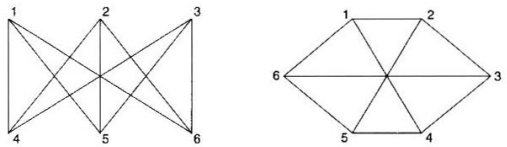


Abbildung 10: *Isomorphe Graphen?*

Um nun alle Isomorphismen zu berechnen gibt es verschiedene, unterschiedlich effiziente, Möglichkeiten.

4.1.2. Naiver Ansatz

Der einfachste Weg ist sicherlich das Testen aller Möglichkeiten, wie die Elemente der Trägermenge aus Struktur 1 auf die der Struktur 2 abgebildet werden können. Besteht die Trägermenge aus n Elementen, so entspricht dies der Berechnung und Testen von $|S_n| = n!$ Kombinationen. Die folgende Tabelle verdeutlicht die Problematik dieses Vorgehens:

n	$\|S_n\| = n!$
4	24
6	720
8	40.320
10	3.628.800
15	$\sim 1.31\ 10^{12}$
20	$\sim 2.43\ 10^{18}$

Bei 20 Elementen würde die Berechnung aller Kombinationen ~77.055 Jahrhunderte dauern (bei 10.000 Permutationen/Sekunde).

Diese maximale Anzahl von Berechnungen ist problembedingt vorgegeben. Jeder Algorithmus muß im schlechtesten Fall tatsächlich alle Permutationen durchlaufen. Das Ziel eines Algorithmus zur Isomorphieerkennung kann also nur eine möglichst starke Reduktion der Möglichkeiten im allgemeinen Fall

sein. Daß dies nicht nur möglich ist, sondern bei geeigneter Darstellung auch sehr große Strukturen getestet werden können, zeigt der folgende Algorithmus.

Eine Komplexitätsbetrachtung im Vergleich zu diesem Ansatz wird in 4.5. vorgenommen.

4.1.3. Backtracking

Die Verwendung eines Backtrackverfahren ermöglicht eine wesentliche Verminderung des Aufwandes zur Isomorphieberechnung. Man geht dabei von den vollständigen Trägermengen S_1.S und S_2.S aus. Die Variable `ISO` speichere die Menge aller vor dem Aufruf des Backtracking vorgenommenen partiellen Permutationen zum Aufbau des Isomorphismus.

Durch einen Aufruf von **`Backtrack(`**$\emptyset$**`,`** S_1**`.S,`** S_2**`.S)`** werden alle Isomorphismen von S_1 nach S_2 ausgegeben. Der Algorithmus selbst hat die Form:

```
Backtrack ( ISO, S, T )
{
   Ist S = T = ∅,
     dann gib ISO aus.
   Sonst
     Für ein i ∈ S
        Für alle j ∈ T
           Bilde ISO1 := ISO·(i → j).
           Ist ISO1 zulässig,
              dann bilde Backtrack (ISO1, S \{i},T \{j})
}
```

Dabei ist der Test, ob die aktuelle partielle Permutation (hier: `ISO1`) zulässig ist, von wesentlicher Bedeutung für die Effizienz. Je früher man entscheiden kann, ob sich eine getroffene Abbildung zu einem Isomorphismus fortsetzen läßt, desto geringer ist der Aufwand zur Bestimmung der Automorphismengruppe. Dies liegt am Prinzip des Backtrackalgorithmus. Die Wege zu den einzelnen Knoten im Backtrackbaum entsprechen in diesem Fall festgelegten Permutationen. Ein Weg von der Wurzel zu einem Blatt hat also maximal die Länge $|S_1.S|$. Weist man an einer Stelle im Algorithmus eine Zuordnung als unzulässig zurück, so muß man den Teilbaum unterhalb dieser Permutation nicht mehr berücksichtigen, da die dort getroffenen Festlegungen nicht mehr zu einem Isomorphismus führen können.

Die Anzahl der im Backtrackverfahren betrachteten Knoten kann jedoch noch weiter eingeschränkt werden.

Dazu nutzt man aus, daß die Automorphismen in Kombination mit der Hintereinanderausführung eine Gruppe bilden. Hat man also neben der Identität bereits einen weiteren Isomorphismus gefunden, so kann man beim Auffinden des nächsten Isomorphismus bereits diese Gruppeneigenschaft ausnutzen und ohne zusätzlichen Aufwand einen neuen Isomorphismus durch eine Kombination mit den zuvor ermittelten angeben.

Dazu ist es jedoch sinnvoll, eine kompakte Speicherung der bekannten Isomorphismen zu wählen, da deren Anzahl unter Umständen sehr groß werden kann.

Wie diese Eigenschaften effektiv ausgenutzt werden können, ist in Kapitel 4.4. beschrieben.

4.1.4. Backtracking mit Iterierter Klassifizierung

Setzt man zur Untersuchung der Fortsetzbarkeit die nachfolgend beschriebene iterierte Klassifizierung ein, kann man in Verbindung mit einem Backtrackverfahren einen wesentlichen Geschwindigkeitsvor-

teil erzielen. Man nutzt dabei die Tatsache aus, daß durch die Festlegung einer neuen Zuordnung von Trägerinstanzen zweier Strukturen im Verlauf des Backtrackings zusätzliche Informationen zur Verfügung stehen, die zur weiteren *Verfeinerung* der aktuellen Klassifizierung genutzt werden können.

Zudem führt man vor dem Start des eigentlichen Algorithmus eine Vorklassifizierung durch, welche die Anzahl der möglichen Permutationen weiter reduziert.

Mit Hilfe des Backtracking versucht man also, ausgehend von einer Vorklassifizierung, in jedem Iterationsschritt eine Verfeinerung der vorherigen Klassifizierung zu erzeugen, bis eine vollständige Klassifizierung erreicht wird. Dies geschieht quasi parallel für die beiden vorliegende Strukturen. Nach jedem Schritt muß geprüft werden, ob die erreichten Klassifizierungen verträglich sind, d.h. ob alle Bedingungen zum Vorhandensein einer Ähnlichkeitsabbildung bzw. eines Isomorphismus erfüllt sind, der die neu entstandenen Klassen der Klassifizierungen aufeinander abbildet.

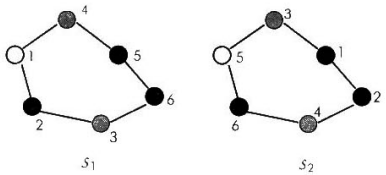
Können in einem Schritt keine neuen Permutationen festgelegt werden, testet man mit Hilfe des Backtracking alle verbleibenden Kombinationen.

Für diese Überprüfung werden ausschließlich die im vorherigen Kapitel definierten Schnittstellen TRÄGER, REL - ERGEBNIS und Γ - ERGEBNIS und die Methoden *rel*, und Γ zur objektorientierten Darstellung von Strukturen verwendet. Sie besitzen dabei - wie bereits erwähnt - ganz unterschiedliche Aufgaben.

Bevor der Algorithmus im Detail besprochen wird, soll anhand eines kleinen Beispiels der Einsatz der Schnittstellen beschrieben werden:

4.1.5. Beispiel

Folgende Strukturen (gefärbter Graph) sollen auf Isomorphie getestet und alle möglichen Einbettungen angegeben werden.



Dies bedeutet, es sollen alle Abbildungen $\varphi : \{1, ..., 6\} \rightarrow \{1, ..., 6\}$ gefunden werden, so daß φ ein Isomorphismus von S_1 nach S_2 ist. φ muß also die Inzidenzen und die Farben der Punkte 1 bis 6 erhalten. Die Trägerklassen und die Klassen für das *rel* - Ergebnis sind für beide Strukturen identisch und es sind keine Toleranzen in den Vergleichsoperatoren zugelassen.

Die Klasse KNOTEN ist abgeleitet von der abstrakten Klasse TRÄGER und enthält als einziges Attribut die Farbe des Knotens. Die Vergleichsoperatoren auf den KNOTEN vergleichen damit die Farbwerte. Die Ausgangsklassifizierung $\mathcal{K} = (\{1, 2, 3, 4, 5, 6\})$ kann damit in einzelne Klassen unterteilt, deren Elemente Knoteninstanzen mit identische Farben sind. Es ergibt sich also die folgende Vorklassifizierung:

S_1	S_2
$\mathcal{K} = (\{1\}, \{2, 5, 6\}, \{3, 4\})$ ○ ● ●	$\mathcal{K} = (\{5\}, \{1, 2, 6\}, \{3, 4\})$ ○ ● ●

Dabei wird vorausgesetzt, daß die Ordnung ○ $<_S$ ● $<_S$ ● auf dem Attribut Farbe einer Knotenin-

stanz definiert ist.

Es ist klar, daß nur Knoteninstanzen mit gleichen Farben aufeinander abgebildet werden können. Da bei beiden Strukturen nur *eine* Knoteninstanz mit der Farbe ◯ auftaucht, kann man bereits eine partielle Permutation festlegen, nämlich $\varphi(1) = 5$, da bei Struktur S_1 Knoten 1 als erster in einer einelementigen Klasse erschienen ist, und bei Struktur S_2 Knoten 5, und beide Knoten identische Farbwerte besitzen. Man führt deshalb die beiden partiellen Einbettungen σ_1 von S_1 und σ_2 von S_2 ein, in denen gespeichert wird, wann ein Knoten Element einer einelementigen Klasse wurde. Da in Struktur 1 Knoten Nummer 1 zuerst Element einer einelementigen Klasse wurde, fügt man σ_1 (die zunächst noch leer ($\sigma_1 = \emptyset$) ist) die partielle Permutation $1 \to 1$ hinzu. In Struktur 2 wurde Knoten Nummer 5 zuerst Element einer einelementigen Klasse. Deshalb ergänzt man die partielle (bisher leere) Permutation σ_2 durch die Abbildung $5 \to 1$. Die rechte Seite der Abbildung zeigt „den Zeitpunkt" an, wann der Knoten auf der linken Seite einelementig wurde (hier: 1).

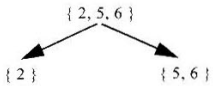
Die gesuchte Abbildung φ ist dann identisch mit der Hintereinaderausführung $\sigma_2^{-1}\sigma_1$ der Numerierungen σ_1 und σ_2. Die folgenden Algorithmen nutzen alle diese Tatsache aus. φ wird in ihnen damit Schritt für Schritt durch die parallele Numerierungen σ_1 von Struktur 1 und σ_2 von Struktur 2 definiert. Bei jeder neuen Festlegung werden auch alle zum dortigen Zeitpunkt möglichen Relationen getestet, so daß frühzeitig ein Abbruch der Algorithmen erfolgen kann, wenn Bedingungen, die für Isomorphie oder Ähnlichkeit notwendig sind, nicht erfüllt sind.

Leider ist im Moment noch keine eindeutige Zuordnung der Knoten aus den übrigen beiden Klassen möglich. Normalerweise müßten jetzt alle Kombinationen getestet werden. Statt dessen nutzt man jedoch die Eigenschaft aus, daß bereits ein Knoten fest zugeordnet werden konnte. Hierfür wird nun die Methode *rel* benötigt. Jede der Klassen $\{2, 5, 6\}$ und $\{3, 4\}$ aus S_1 bzw. $\{1, 2, 6\}$ und $\{3, 4\}$ aus S_2 kann nun in zwei neue Klassen aufgeteilt (verfeinert) werden. Die eine enthält die Knoten in der betrachteten Klasse, die mit dem bereits festgelegten Knoten verbunden sind, die andere enthält die übrigen, nicht damit verbundenen Knoten.

Die Methode *rel* erfüllt diese Aufgabe, indem die aktuelle partielle Numerierung σ_1 bzw. σ_2 übergeben wird, zusammen mit einer zu klassifizierenden Trägerinstanz. Aus der Numerierung wird nun die linke Seite der zuletzt festgelegten Permutation verwendet, also der zuletzt eingebettete Knoten. Das zweite Argument sind dann alle Knoten aus der betrachteten Klasse. Das Ergebnis ist eine Instanz der Klasse KONNEKTIVITÄT, abgeleitet von REL - ERGEBNIS die lediglich ein Attribut enthält, nämlich die Vielfachheit der Bindung zwischen dem zuletzt festgelegten Knoten aus σ_1 und dem Knoten, der als zweites Argument übergeben wird. Die auf dieser Klasse definierten Vergleichsoperatoren entsprechen der Ordnung auf $\mathbb{N}$.

Man durchläuft also mit $rel(\sigma_1, A)$ alle A aus der Klasse $\{2, 5, 6\}$ und berechnet die Konnektivität zwischen dem in diesem Fall zuletzt festgelegten Knoten 1 und den Knoten 2, 5 und 6. Da Knoten 1 nur mit Knoten 2 benachbart ist, aber nicht mit den Knoten 5 und 6, ergibt sich folgende Aufspaltung:

Struktur S_1:

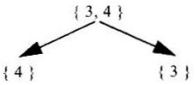


Denn $rel(\sigma_1, B) = rel\left(\left(\begin{smallmatrix} 1 & * \\ 1 & \end{smallmatrix}\right), B\right) = 1 \qquad \forall B \in \{2\}$, aber

$rel(\sigma_1, B) = rel\left(\left(\begin{smallmatrix} 1 & * \\ 1 & \end{smallmatrix}\right), B\right) = 0 \qquad \forall B \in \{5, 6\}$.

Analog dazu für die Klasse $\{3, 4\}$:

Struktur S_1:

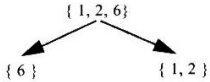


Denn $rel(\sigma_1, B) = rel\left(\left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix} * \right), B\right) = 1 \qquad \forall B \in \{4\}$, aber

$rel(\sigma_1, B) = rel\left(\left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix} * \right), B\right) = 0 \qquad \forall B \in \{3\}$.

Ebenso wird *rel* in Struktur 2 verwendet. Allerdings wird dort nach σ_2, und damit nach dem zuletzt dort festgelegten Knoten 5 verfeinert. Für die Klasse {1, 2, 6} ergibt sich aus der Konnektivität der darin enthaltenen Knoten mit Knoten Nr. 5 die Aufteilung:

Struktur S_2:

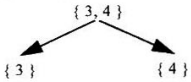


Denn $rel(\sigma_2, B) = rel\left(\left(\begin{smallmatrix} 5 \\ 1 \end{smallmatrix} * \right), B\right) = 1 \qquad \forall B \in \{6\}$, aber

$rel(\sigma_2, B) = rel\left(\left(\begin{smallmatrix} 5 \\ 1 \end{smallmatrix} * \right), B\right) = 0 \qquad \forall B \in \{1, 2\}$.

Die Verfeinerung der Elemente von {3, 4} bezüglich 5 ergibt

Struktur S_2:

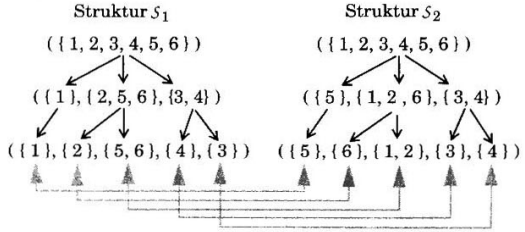


Denn $rel(\sigma_2, B) = rel\left(\left(\begin{smallmatrix} 5 \\ 1 \end{smallmatrix} * \right), B\right) = 1 \qquad \forall B \in \{3\}$, aber

$rel(\sigma_2, B) = rel\left(\left(\begin{smallmatrix} 5 \\ 1 \end{smallmatrix} * \right), B\right) = 0 \qquad \forall B \in \{4\}$.

Es ist klar, daß die aus der Verfeinerung von {2, 5, 6} in Struktur 1 entstandenen beiden Klassen nur auf die in Struktur 2 aus der Verfeinerung der Klasse {1, 2, 6} entstandenen Klassen abgebildet werden können, und dort nur auf diejenigen, die gleiche Werte bzgl. des Ergebnisses von *rel* ergeben haben. Ein wichtiges Kriterium für die Zulässigkeit einer Abbildung ist natürlich, daß die Anzahl der Elemente in den durch die Verfeinerung entstandenen Klassen identisch sind, was hier auch der Fall ist. Analog können die verfeinerten Klassen aus Struktur 1 {3, 4} nur auf die entsprechenden, aus der Verfeinerung von {3, 4} in Struktur 2 hervorgegangenen Klassen abgebildet werden.

Insgesamt ergeben sich durch die Ausnutzung der Konnektivitätseigenschaften neue Klassifizierung für die Elemente von S_1 und S_2:



Die Abbildung zeigt die Entwicklung der Klassifizierungen und darunter die Zuordnung der Klassen aus Struktur 1 und Struktur 2.

Im Laufe der obigen Verfeinerung haben sich nun die zusätzlichen Zuordnungen von einelementigen Klassen und damit von Knoten zwischen S_1 und S_2 ergeben: $2 \to 6$, $4 \to 3$ und $3 \to 4$, da die Ergeb-

nisse bezüglich *rel* auf beiden Seiten übereinstimmen. Das bedeutet wiederum, daß Knoten 2 in Struktur 1 als zweiter, Knoten 4 als dritter und Knoten 3 als vierter fest zugeordnet werden konnte. Demgegenüber wurde Knoten 6 in Struktur 2 als zweiter, Knoten 3 als dritter und Knoten 4 als vierter fest einem Knoten aus Struktur 1 zugeordnet werden konnte. Die Permutation σ_1, welche diese Reihenfolge für Struktur 1 speichert und σ_2, die sie für Struktur 2 speichert, haben nun folgende Gestalt:

$$\sigma_1 = \begin{pmatrix} 1\ 2\ 4\ 3\ * \\ 1\ 2\ 3\ 4 \end{pmatrix},\ \sigma_2 = \begin{pmatrix} 5\ 6\ 3\ 4\ * \\ 1\ 2\ 3\ 4 \end{pmatrix}$$

Die obere Zeile sind die bisherigen einelementigen Knoten, die untere der Zeitpunkt, zu dem sie in eine einelementige Klasse eingeteilt wurden.

Die aktuelle partielle Abbildung σ, die Elemente der Trägermenge von S_1 auf die Elemente der Trägermenge von S_2 abbildet, läßt sich dann also beschreiben durch $\varphi = \sigma_2^{-1}\sigma_1$:

$$\varphi = \begin{pmatrix} 1\ 2\ 4\ 3\ * \\ 5\ 6\ 3\ 4 \end{pmatrix}$$

Für jede neue Festlegung aus σ wird nun eine Verfeinerung der noch nicht einelementigen Klasse {5, 6} in Struktur 1 bzw. {1, 2} in Struktur 2 versucht.

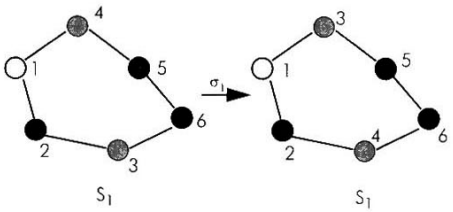
Es ist offensichtlich, daß dadurch auch diese Klassen in einelementige Klassen verfeinert werden und damit eine eindeutige Zuordnung der Trägermengen von Struktur 1 und 2 erzeugt wird. Letztendlich gilt:

$$\sigma_1 = \begin{pmatrix} 1\ 2\ 4\ 3\ 5\ 6 \\ 1\ 2\ 3\ 4\ 5\ 6 \end{pmatrix},\ \sigma_2 = \begin{pmatrix} 5\ 6\ 3\ 4\ 1\ 2 \\ 1\ 2\ 3\ 4\ 5\ 6 \end{pmatrix},\ \varphi = \begin{pmatrix} 1\ 2\ 4\ 3\ 5\ 6 \\ 5\ 6\ 3\ 4\ 1\ 2 \end{pmatrix}$$

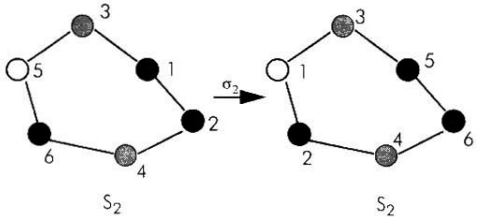
Die Methode Γ wird nach jedem Verfeinerungsschritt aufgerufen, um die Zulässigkeit von σ_1 bzw. σ_2 zu testen. Sie wird in diesem Beispiel nicht verwendet, ist jedoch in einigen Fällen notwendig, wie in den Anwendungskapiteln gezeigt.

Daß nach der Vorklassifizierung bereits eine einelementige Klasse existiert, ist nicht notwendigerweise der Fall. Sind ausschließlich mehrelementige Klassen vorhanden, so müssen durch Backtracking „künstlich" einelementige Klassen erzeugt werden, damit eine Verfeinerung nach obigem Schema überhaupt möglich wird.

Eine Besonderheit von σ_1 bzw. σ_2 wird deutlich, wenn man S_1 bezüglich der dort festgelegten Permutation neu numeriert:



Führt man diese Numerierung auch mit σ_2 und S_2 durch, wie in der folgenden Abbildung:



dann fällt auf, daß die Bilder der Graphen S_1 und S_2 unter σ_1 bzw. σ_2 identisch sind. Das bedeutet jedoch, daß die Bildgraphen durch die Identität aufeinander abgebildet werden. Die so erreichte Umnumerierung erfüllt damit genau die Eigenschaften, die für eine kanonische Numerierung erforderlich ist. In Kapitel 4.2. wird dieser Umstand genau beschrieben.

Prinzipiell stimmt der nachfolgende Algorithmus mit diesem eben beschriebenen Ablauf überein. Es sind dabei natürlich sehr viele Einzelheiten zu beachten, welche die Zulässigkeit der Ähnlichkeitsabbildungen sicherstellen. Zudem können nicht nur einfache Beziehungen zwischen Trägerinstanzen, wie die hier verwendete Konnektivität, sondern durch die Kapselung in der Klasse *rel* - Ergebnis auch komplexere Zusammenhänge genutzt werden.

In den hier gezeigten Beispielen zur Verdeutlichung des Algorithmus wird aus Gründen der Übersichtlichkeit keine Ähnlichkeitserkennung beschrieben. Ihr Einsatz für die Ähnlichkeitserkennung, also der Test auf eine Übereinstimmung zweier Strukturen unter vorgegebenen Toleranzwerten wird ab Kapitel 6.3. beschrieben. Die Vergleichsoperatoren der Trägerinstanzen und die Methode *rel* liefern dort dann auch nichtdiskrete Werte, so daß die Ähnlichkeit (aus den vorgegebenen Toleranzen) bereits während des Ablaufs der Algorithmen an den Stellen mit einbezogen wird, wo diese Schnittstellen verwendet werden.

Wenden wir uns also der detaillierteren Beschreibung des Verfahrens zu. Den komplexen Ablauf des Algorithmus gibt das folgende Diagramm wieder. Es dient zur Orientierung bei der nachfolgenden

Beschreibung der einzelnen Stationen.

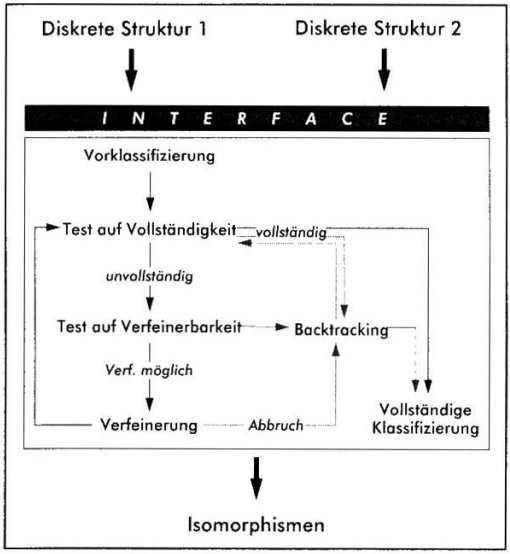


Abbildung 11: *Schematischer Ablauf des Ähnlichkeitsalgorithmus*

Gesucht sind alle möglichen Abbildungen φ, welche die Menge der Trägerinstanzen $\mathcal{S}_1.S$ auf $\mathcal{S}_2.S$ abbilden, unter Berücksichtigung der Eigenschaften **ISO1** - **ISO4**.

Gehe dazu wie folgt vor:

4.1.5.1. Vorklassifizierung

Berechne für die Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ die Vorklassifizierungen $\mathcal{S}_1.\mathcal{K}^0$ und $\mathcal{S}_2.\mathcal{K}^0$ der Trägermenge.

4.1.5.1.1. Satz (Notwendige Bedingungen)

Um die Voraussetzungen **ISO1** und **ISO2** zu erfüllen, muß für die Vorklassifizierungen $\mathcal{S}_1.\mathcal{K}^0$ und $\mathcal{S}_2.\mathcal{K}^0$ gelten:

1. $|\mathcal{S}_1.\mathcal{K}^0| = |\mathcal{S}_2.\mathcal{K}^0|$

2. $\forall\, \mathcal{S}_1.\mathcal{K}_i^0 \;\exists!\, \mathcal{S}_2.\mathcal{K}_j^0$ mit

   a) $A =_S B$ für $A \in \mathcal{S}_1.\mathcal{K}_i^0, B \in \mathcal{S}_2.\mathcal{K}_j^0$

   b) $|\mathcal{S}_1.\mathcal{K}_i^0| = |\mathcal{S}_2.\mathcal{K}_j^0|$

Beweis:

O.B.d.A. angenommen, es gilt $|\mathcal{S}_1.\mathcal{K}^0| > |\mathcal{S}_2.\mathcal{K}^0|$. Damit gilt für mindestens ein $1 \le i \le |\mathcal{S}_1.\mathcal{K}^0|$: $A \neq_S B$ für

ein $A \in S_1.\mathcal{K}_i^0$ und für alle $B \in S_2.S$. Dies steht jedoch im direkten Widerspruch zu Forderung **ISO2**.

Man nehme an, für Klasse $S_1.\mathcal{K}_i^0$ existieren Klassen $S_2.\mathcal{K}_{j_1}^0$ und $S_2.\mathcal{K}_{j_2}^0$, $j_1 \neq j_2$, so daß gilt:

$A =_S B$ für $A \in S_1.\mathcal{K}_i^0, B \in S_2.\mathcal{K}_{j_1}^0$ und

$A =_S C$ für $A \in S_1.\mathcal{K}_i^0, C \in S_2.\mathcal{K}_{j_2}^0$.

Dies ist jedoch nach Definition der Vorklassifizierung nicht möglich ist. Das heißt, genau eine Klasse erfüllt die zweite Bedingung.

Angenommen, es existiert keine Klasse $S_2.\mathcal{K}_j^0$ mit obigen Eigenschaften, dann existiert mindestens eine Klasse $S_1.\mathcal{K}_i^0$, so daß gilt:

$$\forall A \in S_1.\mathcal{K}_i^0 \nexists S_2.\mathcal{K}_j^0 \mid A =_S B \text{ für ein } B \in S_2.\mathcal{K}_j^0$$

Dies verletzt jedoch Forderung **ISO1**.

Das bedeutet, bei der Vorklassifizierung müssen in beiden Strukturen gleich viele Klassen mit einer jeweils identischen Anzahl von Trägerinstanzen erzeugt werden. Die Trägerinstanzen müssen bezüglich ihrer Vergleichsoperatoren übereinstimmen.

Da im Falle einer Isomorphiebetrachtung bei der Vorklassifizierung alle Trägerinstanzen einer Klasse der Vorklassifizierung identische Attribute aufweisen, genügt es, die Identität für jeweils einen Repräsentanten festzustellen. Im Falle der Ähnlichkeit, muß aus Gründen der fehlenden Transitivität ein Vergleich mit allen Elementen vorgenommen werden.

Sind die Bedingungen 4.1.5.1.1. nicht erfüllt, endet der Algorithmus. Es existiert dann keine Ähnlichkeitsabbildung, die S_1 auf S_2 abbildet. Dieser Fall tritt nicht ein, wenn $S_2 \equiv S_1$, da wenigstens die Identität als mögliche Ähnlichkeitsabbildung ermittelt wird.

Die Vorklassifizierungen $S_1.\mathcal{K}^0$ und $S_2.\mathcal{K}^0$ werden im folgenden als aktuelle Klassifizierungen $S_1.\mathcal{K}^a$ und $S_2.\mathcal{K}^a$ bezeichnet.

Definiere eine Funktion $\mathcal{F} : S_1.\mathcal{K}^a \rightarrow S_2.\mathcal{K}^a$, so daß für alle Klassen $\mathcal{K}_i \in S_1.\mathcal{K}^a$ das Paar $\left(S_1.\mathcal{K}_i^a, S_2.\mathcal{K}_j^a\right)$ Satz 4.1.5.1.1., Forderung 2 mit $\mathcal{F}(S_1.\mathcal{K}_i^a) = S_2.\mathcal{K}_j^a$ erfüllt.

Definiere eine Abbildung $\xi \in S_{|S_1.S|}$, die alle bisher festgelegten Permutationen aufnimmt. Setze zu diesem Zeitpunkt $\xi = \{\emptyset\}$. Wird mit dem Algorithmus eine Ähnlichkeit zwischen S_1 und S_2 festgestellt, so enthält ξ die aktuelle Abbildung der Elemente von $S_1.S$ auf $S_2.S$.

Definiere weiterhin Abbildungen $\sigma_1, \sigma_2 \in S_{|S_1.S|}$, die alle bisherigen Numerierungen innerhalb von S_1 bzw. S_2 aufnehmen. Diese Numerierung ist ein Abbild der Reihenfolge, in der einelementige Klassen in der aktuellen Klassifizierung im Algorithmus erzeugt werden.

4.1.5.1.2. Bemerkung

Aus Satz 4.1.5.1.1. und insbesondere daraus, daß Trägerinstanzen nur auf identische Trägerinstanzen abgebildet werden können, folgt, daß für eine Isomorphismus φ auf S gelten muß:

$$\varphi \in \bigcap_{i=1}^{|S_1.\mathcal{K}^a|} N_{S_n}\left(S_1.\mathcal{K}_i^a\right) \cong \bigoplus_{i=1}^{|S_1.\mathcal{K}^a|} S_{|S_2.\mathcal{K}_i^a} \subseteq S_n$$

Dadurch verringert sich die Anzahl der theoretisch zu testenden Permutationen der Trägerinstanzen:

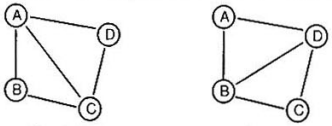
$$\left|\bigcap_{i=1}^{|S_1.\mathcal{K}^a|} N_{S_n}\left(S_1.\mathcal{K}_i^a\right)\right| = \left|\bigoplus_{i=1}^{|S_1.\mathcal{K}^a|} S_{|S_1.\mathcal{K}_i^a|}\right| \leq |S_n|$$

$$\qquad\qquad \| \qquad\qquad \|$$

$$\prod_{i=1}^{|S_1.\mathcal{K}^a|} |S_1.\mathcal{K}_i^a|! \qquad n!$$

Unter Ausnutzung dieser Eigenschaft versucht man nun mit geeigneten Mitteln die Klassifizierungen so fein wie möglich zu unterteilen, um die Anzahl der zu testenden Permutationen zu verringern.

Zuvor muß jedoch die Verträglichkeit der bisherigen Zuordnung $\mathcal{F}$ mit **ISO3** und **ISO4** überprüft werden, um letztendlich alle Bedingungen für eine Ähnlichkeitsabbildung zwischen S_1 und S_2 zu erfüllen. Denn betrachtet man beispielsweise die beiden folgenden Graphen, ergeben sich zwar bereits vollständige Vorklassifizierungen und alle Bedingungen von Satz 4.1.5.1.1. gelten, sie sind jedoch nicht isomorph, wenn zwischen den Knotenattributen A,B,C und D unterschieden wird:



Graph A Graph B

Zur Überprüfung von **ISO3** führe man den folgenden Algorithmus mit den aktuellen Klassifizierungen $S_1.\mathcal{K}^a$ und $S_2.\mathcal{K}^a$ aus:

Für alle $A \in S_1.\mathcal{K}_i^a$, $1 \leq i \leq |S_1.\mathcal{K}^a|$ mit $|S_1.\mathcal{K}_i^a| = 1$

Sei $\{A'\} = \mathcal{F}(\{A\})$

Setze $\xi := \xi \cdot (A \rightarrow A')$

Setze $\sigma_1 := \sigma_1 \cdot (A \rightarrow |\sigma_1| + 1)$ und $\sigma_2 := \sigma_2 \cdot (A' \rightarrow |\sigma_2| + 1)$

Für alle $B \in S_1.\mathcal{K}_j^a$, $1 \leq j \leq |S_1.\mathcal{K}^a|$ mit $|S_1.\mathcal{K}_j^a| = 1$ und $B \neq A$

Sei $\{B'\} = \mathcal{F}(\{B\})$

Falls $S_1.rel(\sigma_1, B) \neq_R S_2.rel(\sigma_2, B')$

dann brich den Algorithmus ab. Es gibt in diesem Fall keine Ähnlichkeit zwischen S_1 und S_2.

Wurde der Algorithmus nicht abgebrochen, muß noch **ISO4** getestet werden. ξ enthält dann alle bisher festgelegten Transpositionen. Ist auch die Bedingung

$$S_1.\Gamma(\sigma_1) = S_2.\Gamma(\sigma_2)$$

erfüllt, kann eine Fortsetzung φ von ξ existieren, so daß $\varphi : S_1.S \rightarrow S_2.S$ eine Ähnlichkeitsabbildung von

S_1 auf S_2 ist.

Dabei wird Satz 3.3.5. ausgenutzt, was zulässig ist, da die Voraussetzung $\sigma_2^{-1} \circ \sigma_1 = \xi$ erfüllt ist. Zu jedem Zeitpunkt im Algorithmus gilt $|\sigma_1| = |\sigma_2|$. Weiterhin werden die Fortsetzbarkeitseigenschaften von Γ und *ref* ausgenutzt, um bereits an dieser Stelle den Versuch einer Fortsetzung von Abbildungen zu verhindern, die nicht zu einer Ähnlichkeitsabbildung fortgesetzt werden können.

Die Variable V_{max} wird zu diesem Zeitpunkt auf 0 gesetzt. In ihr wird später die maximale Anzahl der für eine Verfeinerung bereits herangezogenen Festlegungen in ξ gespeichert.

4.1.5.2. Test auf Vollständigkeit

Sind $S_1.\mathcal{K}^a$ und $S_2.\mathcal{K}^a$ vollständige Klassifizierungen (siehe 3.2.2.3.), so ist eine Ähnlichkeitsabbildung von S_1 auf S_2 gefunden worden.

Man unterscheidet nun in Abhängigkeit des Aufrufes von 4.1.5.2. zwei Fälle:

1. Wurde dieser Test während eines Backtracklaufes aufgerufen, so wird der Algorithmus mit 4.1.5.5. fortgesetzt. Die vorliegende Ähnlichkeitsabbildung muß dann nicht die einzig mögliche sein.
2. Wurde kein Backtracking gestartet, so setze den Algorithmus mit 4.1.5.6. fort. Die vorliegende Ähnlichkeitsabbildung ist dann eindeutig bestimmt.

Sind $S_1.\mathcal{K}^a$ und $S_2.\mathcal{K}^a$ nicht vollständig, so verzweige zum nächsten Schritt (Test auf Verfeinerbarkeit).

4.1.5.3. Test auf Verfeinerbarkeit

Gilt $V_{max} < |\xi|$, so wurde ξ seit dem letzten Aufruf dieses Moduls durch mindestens eine Permutation $(V_1 \rightarrow V_2)$ erweitert. Damit existiert eine Klasse $S_1.\mathcal{K}_i^a$ mit $1 \le i \le |S_1.\mathcal{K}^a|$, für die gilt: $|S_1.\mathcal{K}_i^a| = |\{V_1\}| = 1$ (und damit auch $|\mathcal{F}(S_1.\mathcal{K}_i^a)| = |S_2.\mathcal{K}_j^a| = |\{V_2\}| = 1$), wobei $S_1.\mathcal{K}^a$ und $S_2.\mathcal{K}^a$ die aktuellen Klassifizierungen der beiden betrachteten Strukturen sind.

Es ist daher möglich, eine Verfeinerung der Klassifizierung durchzuführen. Da die Variable V_{max} im Verlauf des Algorithmus lediglich vergrößert wird, wurde die durch $(V_1 \rightarrow V_2)$ erweiterte Permutation nicht zuvor für eine Verfeinerung verwendet. Erhöhe V_{max} um 1. Der nächste Schritt im Algorithmus (Verfeinerung) wird mit dem Index V_{max} aufgerufen.

Gibt es jedoch keine neue einelementige Klasse in der aktuellen Klassifizierung, kann keine Verfeinerung durchgeführt werden. In diesem Falle gilt $V_{max} = |\xi|$ und es wird der Backtracking - Algorithmus gestartet. Dafür wird zur Funktion Backtracking verzweigt. Ist das Backtracking beendet und dieser Test nicht Teil eines vorherigen Backtracklaufes, wird der Algorithmus beendet.

4.1.5.4. Verfeinerung

Zur Verfeinerung werden nun die partiellen Permutationen σ_1 und σ_2 herangezogen, wobei lediglich die ersten V_{max} Festlegungen berücksichtigt werden. Der erste Schritt der Verfeinerung wird bei beiden Strukturen unabhängig voneinander ausgeführt. Die folgende Beschreibung beschränkt sich daher auf das Vorgehen bei Struktur S_1.

Schritt 1: Partitionierung

Erzeuge aus σ_1 eine neue Permutation μ_1, in der alle bisherigen Permutationen $(V_i \rightarrow i)$, $i < V_{max}$ enthalten sind.

Ermittle die erste Klasse mit dem Index j ($1 \le j \le |S_1.\mathcal{K}^a|$) in der aktuellen Klassifizierung, für die $|S_1.\mathcal{K}_j^a| > 1$ gilt, und die bei diesem Aufruf der Verfeinerung noch nicht verwendet wurde. Diese Klasse wird nun bezüglich der in 3.2.2.11. eingeführten Methode $S_1.ref$ ähnlich wie bereits im Beispiel 3.2.3.5.

beschrieben partitioniert. Erzeuge dazu disjunkte Zerlegungen $S_1.\mathcal{K}^a_{j_p}$, $1 \le p \le p_j$ mit:

- $S_1.\mathcal{K}^a_j = \bigcup_{p=1}^{p_j} S_1.\mathcal{K}^a_{j_p}$
- $S_1.\mathcal{K}^a_{j_q} \cap S_1.\mathcal{K}^a_{j_r} = \emptyset \quad \forall\, 1 \le q, r \le p_j, q \ne r$
- $rel(\mu_1, A) =_R rel(\mu_1, B) \quad \forall\, A, B \in S_1.\mathcal{K}^a_{j_q}, 1 \le q \le p_j$
- $rel(\mu_1, A) <_R rel(\mu_1, B) \quad \forall\, A \in S_1.\mathcal{K}^a_{j_q}, B \in S_1.\mathcal{K}^a_{j_{q+1}}$

mit $1 \le q \le p_j - 1$

Analog dazu wird mit Hilfe von σ_2 bzw. μ_2 und $S_2.rel$ die Klasse $\mathcal{F}(S_1.\mathcal{K}^a_j) = S_2.\mathcal{K}^a_v$ der zweiten Struktur in q_v disjunkte Teilklassen zerlegt.

Schritt 2: Identifizierung

Um weiterhin die Fortsetzung von ξ zu einem Ähnlichkeitsabbildung von S_1.S auf S_2.S zu gewährleisten, müssen die folgenden Bedingungen erfüllt sein:

1. $p_j = q_v$
2. $\forall\, S_1.\mathcal{K}^a_{j_r}, 1 \le r \le p_j \;\; \exists!\, S_2.\mathcal{K}^a_{v_q}, 1 \le q \le q_v$ mit

$$rel(\mu_1, A) =_R rel(\mu_2, B) \quad \forall\, A \in S_1.\mathcal{K}^a_{j_r}, B \in S_2.\mathcal{K}^a_{v_q}$$
$$\left|S_1.\mathcal{K}^a_{j_r}\right| = \left|S_2.\mathcal{K}^a_{v_q}\right|$$

Für die Überprüfung dieser Bedingung führe man den folgenden Algorithmus aus:

```
Für alle S1.K^a_jr ,  1 ≤ r ≤ pj

   Setze A ∈ S1.K^a_jr beliebig

   Falls kein oder mehrere S2.K^a_vq ,  1 ≤ q ≤ qv existieren mit:

      S1.rel(μ1, A) =R S2.rel(μ2, A´) und |S1.K^a_jr| = |S2.K^a_vq|

     für ein beliebiges  A´ ∈ S2.K^a_vq ,

        dann brich den Algorithmus ab.
   sonst

     Setze F(S1.K^a_jr) := S2.K^a_vq .

     Falls S1.K^a_jr = {A}  (und damit auch S2.K^a_vq = {A´})

       Setze ξ:= ξ·(A → A´)

       Setze σ1:= σ1·(A → |σ1| + 1) und σ2:= σ2·(A´ → |σ2| + 1)

       Für alle bisherigen einelementigen Klassen { B} mit {B´} = F({B}):

         Falls S1.rel(σ1, B) ≠R S2.rel(σ2, B´)

           brich den Algorithmus ab.

       Falls S1.Γ(σ1) ≠G S2.Γ(σ2)

          brich den Algorithmus ab.
```

Da aufgrund der Konstruktion der Klassifizierungen für jede Klasse $\mathcal{K}$ immer gilt $rel(\mu_1, A) =_R rel(\mu_2, B)$ für alle $A, B \in S_1.\mathcal{K}^a_{i_r}$, muß die Übereinstimmung der Ergebnisse der Methode *rel* jeweils nur für ein Element aus den entsprechenden Klassen in S_1 und S_2 getestet werden.

Wurde der Algorithmus abgebrochen, kann ξ nicht zu einer Ähnlichkeitsabbildung von S_1 auf S_2 fortgesetzt werden. Wurde die Verfeinerung in einem Backtrackaufruf bearbeitet, gehe zurück zum Backtracking. Ansonsten brich den gesamten Algorithmus ab.

Ist der Algorithmus jedoch erfolgreich beendet worden, so gehe zum nächsten Schritt:

Schritt 3: Konstruktion

Die zuvor gewonnene Unterteilung der i - ten Klasse von S_1 und der j - ten von S_2 in $p_i = q_v$ neue Teilklassen führt zu einer neuen Klassifizierung. (o.B.d.A. $i < p_i$)

Es gilt damit

$$S_1.\mathcal{K}^{a+1} = \left(S_1.\mathcal{K}^a_1, \ldots, S_1.\mathcal{K}^a_{i-1}, S_1.\mathcal{K}^a_{i_1}, \ldots, S_1.\mathcal{K}^a_{i_q}, S_1.\mathcal{K}^a_{i+1}, \ldots, S_1.\mathcal{K}^a_{|S_1.\mathcal{K}^a|}\right)$$

und damit $|S_1.\mathcal{K}^{a+1}| = |S_1.\mathcal{K}^a| + p_i - 1$

Analoges gilt für S_2.

Mache nun die neue Klassifizierung zur aktuellen. Bei der Implementierung ist darauf zu achten, daß alle Querverweise auf Klassen der vorherigen Klassifizierung beim Konstruieren der neuen Klassifizierung aktuell bleiben.

Ist die neue Klassifizierung erzeugt, und sind noch Klassen in der aktuellen Klassifizierung, die aus mehr als einem Element bestehen und noch nicht mit Hilfe von μ_1 und μ_2 zur Verfeinerung herangezogen worden sind, so gehe zurück zu Schritt 1.

Erhöhe sonst V_{max} und rufe das Modul Test auf Vollständigkeit auf.

4.1.5.5. Backtracking

Das Backtracking - Modul wird aus verschiedenen Situationen heraus aufgerufen:

Situation 1: Erstmaliger Aufruf

In diesem Fall geschieht der Aufruf immer aus dem Modul 4.1.5.3. (Test auf Verfeinerbarkeit). Es stehen keine einelementigen Klassen in der aktuellen Klassifizierung zur Verfügung, mit denen eine Verfeinerung durchgeführt werden könnte. Mit Hilfe des Backtracking wird nun stattdessen, quasi „künstlich", eine einelementige Klasse geschaffen, damit der Algorithmus fortgesetzt werden kann.

Gehe in diesem Fall wie folgt vor:

Bestimme die Klasse $S_1.\mathcal{K}^a_i$ mit

$$|S_1.\mathcal{K}^a_i| = \min_{1 \le r \le |S_1.\mathcal{K}^a|, |S_1.\mathcal{K}^a_r| > 1} (|S_1.\mathcal{K}^a_r|)$$

Die Wahl der Klasse mit der geringsten Anzahl von Elementen zum Start des Backtrackalgorithmus führt nach Knuth [DK2] zum effektivsten Ablauf. Nach neueren Betrachtungen, etwa von Andrew Kirk [AK], kann jedoch auch jede beliebige nicht - einelementige Klasse gewählt werden. Dies führt in den meisten Fällen nicht zu einer negativen Beeinflussung des Algorithmus.

Ein Index i mit der obigen Eigenschaft muß immer existieren. Gäbe es ihn nicht, so wären, da nur diskrete Strukturen betrachtet werden, alle Klassen einelementig. In diesem Falle wäre das Backtrackmodul jedoch nicht aufgerufen worden, da diese vollständige Klassifizierung entweder ein Isomorphismus wäre oder unzulässig.

Die Klasse $S_1.\mathcal{K}_i^a$ wird nun gesplittet in zwei neue Klassen, wobei eine davon einelementig ist:

Wähle dazu $A \in S_1.\mathcal{K}_i^a$ mit $A = \min_{X \in S_1.\mathcal{K}_i^a}(X)$ als Minimum der Atomindizes und setze $S_1.\mathcal{K}_{i_1}^a = \{A\}$ und $S_1.\mathcal{K}_{i_2}^a = S_1.\mathcal{K}_i^a \setminus \{A\}$. Erzeuge dann eine neue Klassifizierung der Form

$$S_1.\mathcal{K}^{a+1} = \left(S_1.\mathcal{K}_1^a, \ldots, S_1.\mathcal{K}_{i-1}^a, S_1.\mathcal{K}_{i_1}^a, S_1.\mathcal{K}_{i_2}^a, S_1.\mathcal{K}_{i+1}^a, \ldots, S_1.\mathcal{K}_{|S_1.\mathcal{K}^a|}^a\right)$$

Es gilt dann $|S_1.\mathcal{K}^{a+1}| = |S_1.\mathcal{K}^a| + 2$.

Da sich jedoch in $S_2.\mathcal{K}_j^a = \mathcal{F}(S_1.\mathcal{K}_i^a)$ nach Konstruktion ebenso viele Elemente befinden wie in $S_1.\mathcal{K}_i^a$, und jede beliebige Permutation der Form $S_1.\mathcal{K}_i^a \ni A \rightarrow A' \in S_2.\mathcal{K}_j^a$ unter Umständen zu einem Isomorphismus führen würde, geht man wie folgt vor:

Für *jedes* $A' \in S_2.\mathcal{K}_j^a$, wobei die A' in aufsteigender Reihenfolge durchlaufen werden, bilde jeweils wie zuvor die Klassifizierung

$$S_2.\mathcal{K}^{a+1} = \left(S_2.\mathcal{K}_1^a, \ldots, S_2.\mathcal{K}_{j-1}^a, S_2.\mathcal{K}_{j_1}^a, S_2.\mathcal{K}_{j_2}^a, S_2.\mathcal{K}_{j+1}^a, \ldots, S_2.\mathcal{K}_{|S_2.\mathcal{K}^a|}^a\right)$$

mit $|S_2.\mathcal{K}^{a+1}| = |S_2.\mathcal{K}^a| + 2$. Es gilt analog $S_2.\mathcal{K}_j^a = S_2.\mathcal{K}_{j_1}^a \cup S_2.\mathcal{K}_{j_2}^a$ mit $S_2.\mathcal{K}_{j_1}^a = \{A'\}$ und $S_2.\mathcal{K}_{j_2}^a = S_2.\mathcal{K}_j^a \setminus \{A'\}$.

Setze weiterhin $\mathcal{F}(S_1.\mathcal{K}_{i_1}^a) = S_2.\mathcal{K}_{j_1}^a$ und $\mathcal{F}(S_1.\mathcal{K}_{i_2}^a) = S_2.\mathcal{K}_{j_2}^a$. Zur Sicherung von **ISO3** durchlaufe den folgenden Algorithmus:

```
Setze ξ:= ξ ∪ (A → A´),
setze σ1:= σ1·(A → |σ1| + 1) und σ2:= σ2·(A´ → |σ2| + 1)
Falls S1.Γ(σ1) ≠G S2.Γ(σ2)
      brich den Algorithmus ab.
Für alle bisherigen einelementigen Klassen { B} mit {B´} = F({B}):
   Falls    S1.rel(σ1, B) ≠R S2.rel(σ2, B´)
      brich den Algorithmus ab.
```

Gilt darüber hinaus $|S_1.\mathcal{K}_{i_2}^a| = |S_1.\mathcal{K}_{i_1}^a| = 1$, bestand die Ausgangsklasse also lediglich aus zwei Atomen, so durchlaufe den Algorithmus auch mit der Permutation $C \rightarrow C'$, wobei $S_1.\mathcal{K}_{i_2}^a = \{C\}$ und $S_2.\mathcal{K}_{j_2}^a = \{C'\}$.

Wurde der Algorithmus abgebrochen, fahre mit Situation 2 fort. Die Auswahl dieses Elementes führt dann nicht zu einer verträglichen Abbildung der Trägermengen.

Daraus ergeben sich insgesamt $|S_2.\mathcal{K}_j^a|$ Kombinationsmöglichkeiten und damit ebensoviele Backtrackschritte. Sind beide neuen Klassifizierungen erstellt worden, so wird wiederum das Modul 4.1.5.2. aufgerufen.

Wichtig ist, zu Beginn des Backtracking alle relevanten Informationen zu speichern, in diesem Fall also $S_1.\mathcal{K}^a$, $S_2.\mathcal{K}^a$ $\mathcal{F}$, V_{max}, ξ, σ_1 und σ_2.

Die Wahl des Elements mit minimalem Atomindex als neue einelementige Klasse ist vor dem Hintergrund der erweiterten Isomorphiebetrachtungen in Kapitel 4.4. zu sehen. Wird der dort beschriebene Ansatz nicht verfolgt, so kann jedes beliebige Element verwendet werden.

Situation 2: Aufruf aus Verfeinerung

Dieser Fall tritt auf, wenn in Modul 4.1.5.4. die Ähnlichkeitsbedingungen beim Erstellen einer Verfeinerung nicht erfüllt sind. In diesem Fall führt die aktuelle Klassifizierung von S_1 und S_2 nicht mehr zu einer Ähnlichkeitsabbildung. Da ein Aufruf von dieser Stelle voraussetzt, daß Situation 1 bereits zuvor eingetreten ist, versuche durch die nächste Kombinationsmöglichkeit evtl. eine Ähnlichkeitsabbildung zu finden. Wähle dazu nicht mehr das minimalste Element aus der in Situation 1 betrachteten Klasse $S_2.\mathcal{K}_j^a$ als Bild von $A \in S_1.\mathcal{K}_i^a$, sondern das nächstgrößere. Erzeuge die entsprechenden Klassifizierungen wie zuvor und teste **ISO3** und **ISO4**. Rufe dann wieder den Test auf Vollständigkeit auf. Sind dagegen alle Kombinationen ausgeschöpft, so beende das Backtracking.

Situation 3: Aufruf nach vollständiger Klassifizierung

Diese Situation ist ähnlich zur vorherigen. Nur war in diesem Fall eine vollständige Klassifizierung möglich, d.h. es wurde eine Ähnlichkeitsabbildung zwischen S_1 und S_2 gefunden. Gib diesen mit Hilfe von 4.1.5.6. aus. Auch hier muß zuvor Situation 1 erreicht worden sein. Sind dort noch neue Aufteilungen von $S_2.\mathcal{K}_j^a$ möglich, so führe diese aus. Beende sonst das Backtracking.

4.1.5.6. Vollständige Klassifizierung

Dieses Modul wird aufgerufen, wenn in 4.1.5.2. eine vollständige Klassifizierung für S_1 und damit - durch den Algorithmus bedingt - auch für S_2 ermittelt wurde. Das bedeutet, daß eine Ähnlichkeitsabbildung φ zwischen S_1 und S_2 vorhanden ist. Die gesuchte Abbildung φ wird dabei definiert durch die während des Algorithmus in ξ gespeicherten Permutationen.

4.1.5.7. Satz

Die ermittelte Abbildung φ ist eine Ähnlichkeitsabbildung zwischen den endlichen bewerteten Strukturen S_1 und S_2.

Beweis:

Der Beweis folgt konstruktiv anhand des beschriebenen Algorithmus. Alle Verfeinerungen der aktuellen Klassifizierungen wurden unter Berücksichtigung der Isomorphievoraussetzungen **ISO1** - **ISO4** unter Anwendung von Satz 3.3.5. getroffen. Deshalb sind diese Bedingungen auch für die gefundenen Abbildungen zutreffend. Alle übrigen Abbildungen verletzen eine dieser Forderungen. Es werden zudem alle möglichen Abbildungen zwischen S_1 und S_2 gefunden.

4.1.6. Beispiel

Zur Verdeutlichung des Algorithmus wird ein Lauf am Beispiel des bereits zuvor betrachteten gefärbten Würfels ausgeführt. Zunächst ein Fall, in dem ein Isomorphismus und damit eine vollständige Klassifizierung erreicht wird:

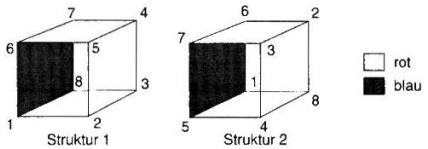


Abbildung 12: *Isomorphie Testbeispiel*

Als Schnittstelle verwenden wir die bereits in 3.2.2.7. verwendeten Funktionen.

4.1.6.1. Ablauf des Algorithmus

Schritt 1: Vorklassifizierung

Für Struktur 1 gilt: $s_1.\mathcal{K}^0 = (\{2,3\},\{1,8\},\{4,5\},\{6,7\})$. Analog dazu für Struktur 2: $s_2.\mathcal{K}^0 = (\{4,8\},\{1,5\},\{2,3\},\{6,7\})$. Alle Bedingungen aus Satz 4.1.5.1.1. sind erfüllt. Es gilt zu diesem Zeitpunkt $\xi = \sigma_1 = \sigma_2 = \emptyset$, daher muß auch **ISO3** und **ISO4** nicht getestet werden. Für die Zuordnungsfunktion $\mathcal{F}$ gilt:

$\mathcal{K}$	**{2,3}**	**{1,8}**	**{4,5}**	**{6,7}**
$\mathcal{F}(\mathcal{K})$	{4,8}	{1,5}	{2,3}	{6,7}

Unter Anwendung von Bemerkung 4.1.5.1.2. reduziert sich die Anzahl der möglichen Permutationen von $8! = 40320$ auf $2!\cdot 2!\cdot 2!\cdot 2! = 16$.

V_{max} besitzt den Wert 0.

Schritt 2: Test auf Vollständigkeit

Da in der aktuellen Klassifizierung Klassen mit mehr als einem Element existieren, kann keine vollständige Klassifizierung vorliegen. Gehe zum nächsten Schritt.

Schritt 3: Test auf Verfeinerbarkeit

Da $V_{max} = |\xi| = 0$, also keine Klasse in $s_1.\mathcal{K}^0$ existiert, die aus nur einem Element besteht, kann keine Verfeinerung durchgeführt werden. Deshalb wird der Backtrackalgorithmus gestartet.

Schritt 4: Backtracking

Alle Klassen in $s_1.\mathcal{K}^0$ sind gleich groß. Deshalb wird die erste Klasse mit den Elementen { 2, 3 } aufgespalten. Es gibt 2! Kombinationsmöglichkeiten, d.h. es müssen zwei Backtrackschritte durchlaufen werden. Im ersten Schritt ergibt die Aufspaltung die beiden neuen Klassifizierungen

$s_1.\mathcal{K}^1 = (\{2\},\{3\},\{1,8\},\{4,5\},\{6,7\})$ und

$s_2.\mathcal{K}^1 = (\{4\},\{8\},\{1,5\},\{2,3\},\{6,7\})$.

Für $\mathcal{F}$ gilt dann:

$\mathcal{K}$	**{2}**	**{3}**	**{1,8}**	**{4,5}**	**{6,7}**
$\mathcal{F}(\mathcal{K})$	{4}	{8}	{1,5}	{2,3}	{6,7}

Weiterhin ergeben sich aus dem Algorithmus zur Überprüfung von **ISO3**´:

$$\sigma_1^{-1} = /23*/,\ \sigma_2^{-1} = /48*/,\ \xi = \sigma_2^{-1} \circ \sigma_1$$

Mit der neuen Klassifizierung wird wieder der Test auf Vollständigkeit aufgerufen. Um zu verdeutlichen, daß sich die folgenden Schritte aus der Backtrackkombination 1 heraus aufgerufen wurden, werden sie mit B1 gekennzeichnet.

Schritt 5 / B1: Test auf Vollständigkeit

Analog zu Schritt 2.

Schritt 6 / B1: Test auf Verfeinerbarkeit

Durch den Backtrackalgorithmus sind ξ zwei neue Abbildungen hinzugefügt worden. Da weiterhin $V_{max} = 0$ ($< |\xi| = 2$) gilt, kann also eine Verfeinerung durchgeführt werden. Gehe zu Schritt 7 mit $V_{max} = 1$.

Schritt 7 / B1: Verfeinerung

Zur Verfeinerung werden nun die Permutationen μ_1 und μ_2 verwendet. Da $V_{max}=1$, wird zunächst nur die erste Permutation aus σ_1 und σ_2 verwendet. Es gilt also: $\mu_1 = \binom{2}{1}^{*}$ und $\mu_2 = \binom{4}{1}^{*}$. In $s_1.\mathcal{K}^1$ wird, wenn möglich, die Klasse $s_1.\mathcal{K}^1_3 = \{1, 8\}$, und in $s_2.\mathcal{K}^1$ die Klasse $s_2.\mathcal{K}^1_3 = \{1, 5\}$ aufgespalten. Die nachfolgende Tabelle gibt die Ergebnisse der Methode *ref* an, die bei der Verwendung von μ_1 für die Elementen aus $s_1.\mathcal{K}^1_3$ und bei μ_2 für die Elementen aus $s_2.\mathcal{K}^1_3$ erzielt werden. Die Tabellenwerte können in 3.2.3.5. nachgeschlagen werden.

Struktur 1	
ref (μ_1 , 1)	(1, 1)
ref (μ_1 , 8)	(0, 1)

Struktur 2	
ref (μ_2 , 1)	(0, 1)
ref (μ_2 , 5)	(1, 1)

Da in s_1 *ref* (μ_1 , 1) $>_R$ *ref* (μ_1 , 8) und in s_2 *ref* (μ_2 , 1) $>_R$ *ref* (μ_2 , 5), wird $s_1.\mathcal{K}^1$ aufgeteilt in

$s_1.\mathcal{K}^{1'} = (\{2\}, \{3\}, \underbrace{\{8\}, \{1\}}, \{4, 5\}, \{6, 7\})$, und $s_2.\mathcal{K}^1$ in

$s_2.\mathcal{K}^{1'} = (\{4\}, \{8\}, \underbrace{\{1\}, \{5\}}, \{2, 3\}, \{6, 7\})$. Wie man sich leicht selbst überzeugen kann, sind alle Forderungen aus 4.1.5.4. / Schritt 1 erfüllt.

Desweiteren sind auch die Bedingungen aus 4.1.5.4. / Schritt 2 erfüllt. Aus dem Algorithmus zur Identifizierung ergibt sich dann für $\mathcal{F}$:

$\mathcal{K}$	{2}	{3}	{8}	{1}	{4,5}	{6,7}
$\mathcal{F}(\mathcal{K})$	{4}	{8}	{1}	{5}	{2,3}	{6,7}

und für die Einbettungen σ_1 und σ_2 sowie ξ:

$$\sigma_1^{-1} = /2381*/,\ \sigma_2^{-1} = /4815*/,\ \xi = \sigma_2^{-1} \circ \sigma_1$$

Für die neue Klassifizierung $s_1.\mathcal{K}^2$ gilt $s_1.\mathcal{K}^2 = s_1.\mathcal{K}^{1'}$ und analog $s_2.\mathcal{K}^2 = s_2.\mathcal{K}^{1'}$.

Die Klassen $s_1.\mathcal{K}^2_5$ und $s_1.\mathcal{K}^2_6$ bzw. $s_2.\mathcal{K}^2_5$ und $s_2.\mathcal{K}^2_6$ wurden noch nicht mit Hilfe von μ_1 und μ_2 verfeinert.

Für $s_1.\mathcal{K}^2_5$ gilt analog zu oben:

Struktur 1	
ref (μ_1 , 4)	(0, 1)
ref (μ_1 , 5)	(1, 1)

Struktur 2	
ref (μ_2 , 2)	(0, 1)
ref (μ_2 , 3)	(1, 1)

Damit wird $s_1.\mathcal{K}^2$ aufgeteilt in

$s_1.\mathcal{K}^2 = (\{2\},\{3\},\{8\},\{1\},\underbrace{\{4\},\{5\}},\{6,7\})$, und $s_2.\mathcal{K}^2$ in $s_2.\mathcal{K}^2 = (\{4\},\{8\},\{1\},\{5\},\underbrace{\{2\},\{3\}},\{6,7\})$. Die Forderungen aus 4.1.5.4. / Schritt 1 und aus 4.1.5.4. / Schritt 2 sind erfüllt. Damit ergeben sich die neuen Klassifizierungen $s_1.\mathcal{K}^3 = s_1.\mathcal{K}^2$ und $s_2.\mathcal{K}^3 = s_2.\mathcal{K}^2$ mit:

$\mathcal{K}$	{2}	{3}	{8}	{1}	{4}	{5}	{6,7}
$\mathcal{F}(\mathcal{K})$	{4}	{8}	{1}	{5}	{2}	{3}	{6,7}

und für die Einbettungen σ_1 und σ_2 sowie ξ:

$$\sigma_1^{-1} = /238145*/,\ \sigma_2^{-1} = /481523*/,\ \xi = \sigma_2^{-1} \circ \sigma_1$$

Für $s_1.\mathcal{K}_7^3 (= s_1.\mathcal{K}_6^2)$ gilt dann:

Struktur 1	
$ref(\mu_1, 6)$	(0, 1)
$ref(\mu_1, 7)$	(0, 0)

Struktur 2	
$ref(\mu_2, 6)$	(0, 0)
$ref(\mu_2, 7)$	(0, 1)

Damit wird $s_1.\mathcal{K}^3$ aufgeteilt in

$s_1.\mathcal{K}^3 = (\{2\},\{3\},\{8\},\{1\},\{4\},\{5\},\underbrace{\{6\},\{7\}})$,

und $s_2.\mathcal{K}^3$ in

$s_2.\mathcal{K}^3 = (\{4\},\{8\},\{1\},\{5\},\{2\},\{3\},\underbrace{\{7\},\{6\}})$.

Die Forderungen aus 4.1.5.4. / Schritt 1 und aus 4.1.5.4. / Schritt 2 sind wiederum erfüllt. Damit ergeben sich die neuen Klassifizierungen $s_1.\mathcal{K}^4 = s_1.\mathcal{K}^3$, $s_2.\mathcal{K}^4 = s_2.\mathcal{K}^3$ und $s_2.\mathcal{K}^3 = s_2.\mathcal{K}^2$, sowie :

$\mathcal{K}$	{2}	{3}	{8}	{1}	{4}	{5}	{6}	{7}
$\mathcal{F}(\mathcal{K})$	{4}	{8}	{1}	{5}	{2}	{3}	{7}	{6}

und die Einbettungen σ_1 und σ_2 sowie ξ:

$$\sigma_1^{-1} = /23814567/,\ \sigma_2^{-1} = /48152376/,\ \xi = \sigma_2^{-1} \circ \sigma_1$$

Da zwar V_{max} weiterhin den Wert 1 besitzt und inzwischen durch die bei der Verfeinerung neu entstandenen einelementigen Klassen $|\xi| = 8$ gilt, aber in $s_1.\mathcal{K}^4$ bzw. $s_2.\mathcal{K}^4$ keine Klassen mehr vorhanden sind, die nach 4.1.5.4. zu einer Verfeinerung herangezogen werden könnten, wird zum nächsten Schritt verzweigt.

Schritt 8 / B1: Test auf Vollständigkeit

$s_1.\mathcal{K}^4$ und $s_2.\mathcal{K}^4$ sind vollständige Klassifizierungen. Demnach wurde eine Ähnlichkeitsabbildung φ_1 gefunden, die s_1 auf s_2 abbildet. Mit Hilfe von 4.1.5.6. kann diese angegeben werden. Es gilt:

$$\varphi_1 = \xi = \sigma_2^{-1} \circ \sigma_1 = /54823761/$$

Vergleicht man dieses Ergebnis mit Abbildung 12, so kann man sich leicht von dessen Korrektheit überzeugen.

Da Schritt 8 aus einem Backtrackschritt (B1) aufgerufen wurde, wird wieder zum Backtracking ver-

zweigt.

Schritt 9: Backtracking

In Schritt 4 wurde die Klasse $s_1.\mathcal{K}_1^0$ mit den Elementen { 2, 3 } aufgespalten. Die erste der sich daraus ergebenden zwei Kombinationen wurde erfolgreich bearbeitet. Die zweite Kombinationsmöglichkeit führt zu den beiden neuen Klassifizierungen

$s_1.\mathcal{K}^1 = (\{2\}, \{3\}, \{1,8\}, \{4,5\}, \{6,7\})$ und

$s_2.\mathcal{K}^1 = (\{8\}, \{4\}, \{1,5\}, \{2,3\}, \{6,7\})$.

Für $\mathcal{F}$ gilt dann:

$\mathcal{K}$	{2}	{3}	{1,8}	{4,5}	{6,7}
$\mathcal{F}(\mathcal{K})$	{8}	{4}	{1,5}	{2,3}	{6,7}

Weiterhin ergeben sich aus dem Algorithmus zur Überprüfung von **ISO3**′:

$$\sigma_1^{-1} = /23*/,\ \sigma_2^{-1} = /84*/,\ \xi = \sigma_2^{-1} \circ \sigma_1$$

Mit dieser neuen, zweiten möglichen Klassifizierung wird wieder der Test auf Vollständigkeit aufgerufen. Um zu verdeutlichen, daß die folgenden Schritte aus der Backtrackkombination 2 heraus aufgerufen wurden, werden sie mit B2 gekennzeichnet.

Schritt 10 / B2: Test auf Vollständigkeit

Analog zu Schritt 2.

Schritt 11 / B2: Test auf Verfeinerbarkeit

Die durch den Backtrackalgorithmus in Schritt 9 für ξ neu erzeugten Abbildungen können zur Verfeinerung verwendet werden, da $V_{max} = 0$ ($< |\xi| = 2$). Gehe zu Schritt 12 mit dem $V_{max} = 1$.

Schritt 12 / B2: Verfeinerung

Zur Verfeinerung werden nun die Permutationen μ_1 und μ_2 verwendet. Da $V_{max}=1$, wird wiederum nur die erste Permutation aus σ_1 und σ_2 verwendet. Es gilt also: $\mu_1 = \binom{2}{1}*$ und $\mu_2 = \binom{8}{1}*$. Beachte, daß μ_2 nicht identisch zu Schritt 7 / B1 ist. In $s_1.\mathcal{K}^1$ wird nun, wenn möglich, die Klasse $s_1.\mathcal{K}_3^1 = \{1,8\}$, und in $s_2.\mathcal{K}^1$ die Klasse $s_2.\mathcal{K}_3^1 = \{1,5\}$ aufgespalten. μ_2 Es ergeben sich nach analogem Vorgehen zu Schritt 7 / B1 zunächst die beiden neuen Klassifizierungen

$s_1.\mathcal{K}^2 = (\{2\}, \{3\}, \{8\}, \{1\}, \{4,5\}, \{6,7\})$ und

$s_2.\mathcal{K}^2 = (\{8\}, \{4\}, \{5\}, \{1\}, \{2,3\}, \{6,7\})$ mit:

$\mathcal{K}$	{2}	{3}	{8}	{1}	{4,5}	{6,7}
$\mathcal{F}(\mathcal{K})$	{8}	{4}	{5}	{1}	{2,3}	{6,7}

und für die Einbettungen σ_1 und σ_2 sowie ξ:

$$\sigma_1^{-1} = /2381*/,\ \sigma_2^{-1} = /8451*/,\ \xi = \sigma_2^{-1} \circ \sigma_1$$

Nach der Verfeinerung von $S_1.\mathcal{K}_5^2$ bzw. $S_2.\mathcal{K}_5^2$ liegen die Klassifizierungen

$$S_1.\mathcal{K}^3 = (\{2\}, \{3\}, \{8\}, \{1\}, \underbrace{\{4\}, \{5\}}, \{6, 7\}), \text{ und}$$

$$S_2.\mathcal{K}^3 = (\{8\}, \{4\}, \{5\}, \{1\}, \underbrace{\{3\}, \{2\}}, \{6, 7\}) \text{ vor,}$$

wobei:

$\mathcal{K}$	{2}	{3}	{8}	{1}	{4}	{5}	{6,7}
$\mathcal{F}(\mathcal{K})$	{8}	{4}	{5}	{1}	{3}	{2}	{6,7}

und für die Einbettungen σ_1 und σ_2 sowie ξ:

$$\sigma_1 = /238145*/,\ \sigma_2 = /845132*/,\ \xi = \sigma_2 \circ \sigma_1^{-1}$$

Ein weiterer Verfeinerungsschritt ergibt

$$S_1.\mathcal{K}^4 = (\{2\}, \{3\}, \{8\}, \{1\}, \{4\}, \{5\}, \underbrace{\{6\}, \{7\}}) \text{ und}$$

$$S_2.\mathcal{K}^4 = (\{8\}, \{4\}, \{5\}, \{1\}, \{3\}, \{2\}, \underbrace{\{6\}, \{7\}}) \text{ mit}$$

$\mathcal{K}$	{2}	{3}	{8}	{1}	{4}	{5}	{6}	{7}
$\mathcal{F}(\mathcal{K})$	{8}	{4}	{5}	{1}	{3}	{2}	{6}	{7}

und die Einbettungen σ_1 und σ_2 sowie ξ:

$$\sigma_1^{-1} = /23814567/,\ \sigma_2^{-1} = /84513267/,\ \xi = \sigma_2^{-1} \circ \sigma_1$$

Danach kann keine weitere Verfeinerung durchgeführt werden.

Schritt 13 / B2: Test auf Vollständigkeit

$S_1.\mathcal{K}^4$ und $S_2.\mathcal{K}^4$ sind vollständige Klassifizierungen. Demnach wurde eine zweite Ähnlichkeitsabbildung φ_2 gefunden, die S_1 auf S_2 abbildet. Mit Hilfe von 4.1.5.6. kann diese angegeben werden. Es gilt:

$$\varphi_2 = \xi = \sigma_2^{-1} \circ \sigma_1 = /18432675/$$

Vergleicht man dieses Ergebnis mit Abbildung 12, so ist offensichtlich auch diese Zuordnung der Knoten möglich.

Da Schritt 13 aus einem Backtrackschritt (B2) aufgerufen wurde, wird wieder zum Backtracking verzweigt.

Schritt 14: Backtracking

Da keine weiteren Kombinationsmöglichkeiten aus Schritt 4 mehr übrig sind, wird der Algorithmus beendet. Die gefundenen Abbildungen φ_1 und φ_2 werden ausgegeben.

4.1.6.2. Fazit

Die durch das Backtracking erzeugten Zuordnungen der Elemente der ersten Klassen in S_1 und S_2 haben jeweils direkt zu der einzigen sich daraus noch ergebenden Ähnlichkeitsabbildung geführt. Statt der möglichen 16 Permutationen mußten letztendlich nur 2 erzeugt werden !

4.1.6.3. Bemerkung

Den Test, der bei jeder neu festgelegten Zuordnung eines Elementes der Trägermenge zur Sicherung von Forderung **ISO3** durchgeführt wird, könnte zum Teil in Methode *rel* mit aufgenommen werden. Man erweitert dazu die Klasse REL - ERGEBNIS derart, daß nicht nur die von der Methode *rel* berechneten Werte gespeichert werden, sondern auch das Ergebnis der Methode Γ, das ja nur aufgrund der übergebenen Numerierung berechnet wird. Die zu einer der Methode *rel* übergebenen Numerierung gehörende Instanz Γ - ERGEBNIS speichert man jedoch dann möglichst zwischen, um doppelte Berechnungen zu vermeiden.

4.1.7. Unterschiede zu McKay

Gegenüber dem Originalalgorithmus von Brendan D. Mc.Kay [MK] wurde hier einige Änderungen vorgenommen.

4.1.7.1. Verfeinerung

Bei McKay wird eine Verfeinerung immer bezüglich einer *Klasse* der aktuellen Klassifizierung vorgenommen. In der Arbeit von Schimmel wird der Ablauf der Verfeinerung von Klassen nach der Festlegung mindestens einer *einelementigen Klasse* beschrieben. Sollte diese nicht vorhanden sein, so kann sie wie hier mit Hilfe eines Backtrackalgorithmus erzeugt werden. Der vorliegende Algorithmus bietet die Möglichkeit, eine Verfeinerung bezüglich aller bereits festgelegten Zuordnungen zwischen den Trägerinstanzen vorzunehmen, um auch die Zusammenhänge zwischen mehr als zwei Trägerinstanzen zu ermöglichen. Da die Festlegung von Zuordnungen einelementige Klassen voraussetzt, kann (wie bei Schimmel) eine Verfeinerung nur nach dem Auftreten mindestens einer einelementigen Klasse durchgeführt werden. Das bedeutet, daß bei McKay unter Umständen eher verfeinert werden kann, bevor das Backtracking aufgerufen wird. Dies ist jedoch im Vergleich mit McKay vertretbar, da der Gesamtaufwand in beiden Fällen identisch ist.

Das folgende Beispiel zeigt einen Vergleich des Aufwandes beider Ansätze bei der Verfeinerung.

4.1.7.2. Beispiel

Man betrachte den folgenden Graphen:

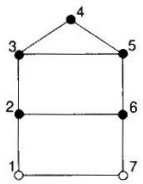


Abbildung 13: *Referenzbeispiel für McKay*

Die Vorklassifizierung nach der Färbung ergibt:

$$\mathcal{K}^0 = (\{1, 7\}, \{2, 3, 4, 5, 6\})$$

Als weiteres Klassifizierungsmerkmal wird ausschließlich die Konnektivität verwendet. Während im vorliegenden Algorithmus jetzt bereits das Backtracking einsetzt, versucht man im Originalalgorithmus die größere Klasse mit Hilfe der kleineren zu verfeinern. Dafür faßt man alle Knoten zusammen, die mit mindestens einem Knoten aus der kleineren Klasse verbunden sind. Das folgende Diagramm zeigt

vereinfacht die Klassenentwicklung des Originalverfahrens:

Klassifizierungen		Aufwand
({1, 7}, {2, 3, 4, 5, 6})		2 x 5
({1, 7}, {2, 6}, {3, 4, 5})		2 x 3
({1, 7}, {2, 6}, {3, 5}, {4})		
({1}, {7}, {2, 6}, {3, 5}, {4})	({7}, {1}, {2, 6}, {3, 5}, {4})	1 x 4 + 1 x 4
({1}, {7}, {2}, {6}, {3, 5}, {4})	({7}, {1}, {6}, {2}, {3, 5}, {4})	1 x 2 + 1 x 2
({1}, {7}, {2}, {6}, {3}, {5}, {4})	({7}, {1}, {6}, {2}, {5}, {3}, {4})	28

Die unterstrichenen Klassen wurden zur Verfeinerung herangezogen. Dem gegenüber steht der hier aufgezeigte Algorithmus mit dem folgenden Verlauf / Aufwand:

Klassifizierungen		Aufwand
({1, 7}, {2, 3, 4, 5, 6})		
({1}, {7}, {2, 3, 4, 5, 6})	({7}, {1}, {2, 3, 4, 5, 6})	1 x 5 + 1 x 5
({1}, {7}, {2}, {3, 4, 5, 6})	({7}, {1}, {6}, {2, 3, 4, 5})	1 x 4 + 1 x 4
({1}, {7}, {2}, {6}, {3, 4, 5})	({7}, {1}, {6}, {2}, {3, 4, 5})	1 x 3 + 1 x 3
({1}, {7}, {2}, {6}, {3}, {4, 5})	({7}, {1}, {6}, {2}, {5}, {3, 4})	1 x 2 + 1 x 2
({1}, {7}, {2}, {6}, {3}, {5}, {4})	({7}, {1}, {6}, {2}, {5}, {3}, {4})	28

Der Aufwand bezieht sich in diesem Fall lediglich auf die Anzahl der Konnektivitätsprüfungen.

Die Elemente in den unterstrichenen Klassen sind die für eine Verfeinerung nach der Konnektivität relevanten, zuletzt festgelegten Elemente der Trägermenge.

Es ist einfach, allgemein nachzuweisen, daß dieser Aufwand bei beiden Verfahren immer identisch ist, indem man die Verfeinerung bezüglich einer mehrelementigen Klasse als Kombination der Verfeinerung bezüglich den einzelnen Elementen und einer nachfolgenden Vereinigung der Elemente mit gleichen Konnektivitätsbeziehungen auffaßt.

Insofern ist auch der evtl. anfallende zusätzliche Aufwand zur Verwaltung des Backtrackbaumes im Vergleich zur im anderen Falle notwendigen Bildung der Vereinigung von Knoten zu vernachlässigen.

Dennoch bietet die zweite Möglichkeit einen unter Umständen entscheidenden Vorteil. Durch das frühere Erstellen des Backtrackbaumes werden auch früher Permutationen festgelegt. Unter Ausnutzung der Isomorphieeigenschaften kann man dadurch jedoch auch zu einem früheren Zeitpunkt Teilbäume eliminieren, was zu einer Reduktion des Gesamtaufwandes führt.

4.1.7.3. Klassifikationsfunktion

Da der Originalalgorithmus ausschließlich auf Graphenisomorphie ausgerichtet war, bzw. auf die Isomorphie von Strukturen, die sich als Graphen darstellen lassen, beschränkt sich die Klassifikationsfunktion, die bei der Verfeinerung benutzt wird, lediglich auf die direkten Konnektivitäten. Die hier definierte, allgemeine, objektorientierte Schnittstelle für diskrete Strukturen ermöglicht natürlich darüber hinaus eine wesentliche Erweiterung des Einsatzgebietes des Algorithmus.

Selbst wenn man sich weiterhin auf die Betrachtung von Graphen beschränkt, ist es doch sehr viel einfacher, zusätzliche Informationen mit bei der Klassifizierung der Strukturen zu berücksichtigen.

Beispielsweise können zur Verfeinerung nicht nur die Konnektivität herangezogen werden, sondern auch die Entfernung zu übrigen, nicht direkt verbundenen Trägerinstanzen. Allein dadurch wird die Effektivität des Verfahrens - gerade bei großen Beispielen - weiter erhöht.

4.1.7.4. Modellwechsel

Ein weiterer, wichtiger Punkt ist auch die Möglichkeit, durch eine sehr flexible Isomorphiedefinition, und die Kapselung von Informationen innerhalb der Schnittstellen, Strukturen, die auf verschiedene mathematischen Modellen basieren, miteinander zu vergleichen, ohne zuvor eine aufwendige Konvertierung der gesamten Struktur vornehmen zu müssen.

Dennoch ist der vorgestellte Algorithmus durch die allgemeinen Schnittstellen so flexibel, daß ein zum Originalalgorithmus identisches Verhalten erreicht werden kann. Auf diese Art kann man auch die Effektivität, die durch Verbesserung der Schnittstellen zu dem einer Struktur zugrundeliegenden mathematischen Modells relativ einfach vergleichen.

Das nun folgende Kapitel beschäftigt sich mit der Berechnung einer kanonischen Numerierung. Zugrunde liegt die auch für die Ähnlichkeit verwendete allgemeine, objektorientierte Beschreibung endlicher Strukturen.

4.2. Kanonische Numerierung

Eine einheitliche (= kanonische) Repräsentation von diskreten Strukturen ist eine wesentliche Voraussetzung für deren effiziente Speicherung und Verwaltung. Die in 3.2. vorgestellte standardisierte Schnittstelle ist hierfür bereits von großem Vorteil. Der Algorithmus zur Isomorphiebestimmung ist bereits äußerst effizient, ein Verfahren mit linearem Aufwand wäre jedoch wünschenswert. Um dies realisieren zu können, muß die diskrete Struktur in einer standardisierten Form vorliegen, durch die isomorphe Strukturen auch identisch repräsentiert werden. Dieses Ziel wird für diskrete Strukturen durch eine kanonische Numerierung der Trägerinstanzen erreicht, eine Lösung des bekannten Normalformproblems, das allenthalben auftritt.

Eine kanonische Numerierung ermöglicht eine wesentliche Beschleunigung und Vereinfachung der meisten Algorithmen für diskrete Strukturen, insbesondere jedoch der Isomorphieerkennung.

Ziel einer kanonischen Numerierung ist eine eindeutige Numerierung der Trägerinstanzen anhand spezifischer Eigenschaften. Isomorphe (vgl. Kapitel 3.3.) Strukturen erhalten damit eine übereinstimmende Numerierung der Trägerinstanzen und können leicht, d.h. mit annähernd linearem Zeitaufwand, identifiziert werden. Das bedeutet, die Struktureigenschaften von Trägerinstanzen mit identischer Nummer und deren Beziehungen untereinander müssen nach einer kanonischen Numerierung der Struktur übereinstimmen.

Vor allem bei einer Anwendung in großen Datenbanken, wo unter Umständen viele Isomorphietests durchzuführen sind, bringt eine kanonische Darstellung der Objekte erhebliche Geschwindigkeitsvorteile.

Praktische Anwendung findet dieses Verfahren bereits in einer Datenbank zur Speicherung der mit MOLGEN erzeugten chemischen Strukturen, sowie in einem Programmpaket zur kombinatorischen Chemie, das als Erweiterung des MOLGEN Projektes zum Einsatz kommen wird. (siehe auch 6.4. und 6.9.)

Für einige spezielle Anwendungsbereiche wurden kanonische Numerierungen bereits berechnet (siehe auch 7.3.), jedoch immer durch ein starres Regelwerk, so daß eine Anpassung oder Übertragung auf andere Probleme nicht möglich war. Der folgende Algorithmus ist durch die vorgestellte Beschreibung diskreter Strukturen im Kapitel 3.2. allgemein einsetzbar und effizient.

4.2.1. Definition

Sei S eine diskrete Struktur. Sei $\sigma \in \mathcal{E}_{|S.S|}$ eine beliebige Numerierung der Trägerinstanzen von S. $S_\sigma = S_{\sigma(S.S)}$ bezeichne die Struktur, die aus der Struktur S durch die neue Numerierung der Trägerinstanzen mit Hilfe von σ entsteht. Unter einer kanonischen Numerierung versteht man eine *ausgezeichnete* Numerierung $\sigma_{can} \in \mathcal{E}_{|S.S|}$ mit folgenden Eigenschaften:

CAN1 $S_{\sigma_{can}(S.S)} \cong S$

CAN2 $S_{\sigma_{can}(S.S)} \cong S_{\sigma_{can}(\sigma(S.S))} \quad \forall\, \sigma \in \mathcal{E}_{|S.S|}$

CAN3 $S_{\delta(\sigma_{can})} \cong S_{\sigma_{can}} \Leftrightarrow (\delta) \in \mathrm{Aut}(S)$

Das heißt, die kanonische Numerierung ist invariant bzgl. der aktuellen Numerierung der Trägerinstanzen. Außerdem sind kanonisch numerierte Strukturen isomorph zur Ausgangsstruktur. Existieren zwei unterschiedliche Numerierungen, so gehen sie unter Anwendung von geeigneten Elementen der Automorphismengruppe der Struktur ineinander über.

4.2.2. Berechnungsmöglichkeiten

Wie auch bei der Berechnung der Isomorphie gibt es verschiedene Möglichkeiten, die kanonische Numerierung zu bestimmen. Die dabei erzielten Ergebnisse müssen sich nicht zwangsläufig entsprechen, d.h. die sich ergebenden Numerierungen sind immer unter Berücksichtigung des verwendeten Algorithmus zu unterscheiden. Die kanonische Form ist also immer nur kanonisch bezüglich des eingesetzten Algorithmus. Es gibt im Prinzip nicht *die* ganz spezielle kanonische Form für diskrete Strukturen. Vielmehr bietet der folgende Ansatz die Möglichkeit gerade durch die individuellen Eigenschaften der betrachteten Struktur die diesbezüglich effizienteste Numerierung durchzuführen. Wichtig ist dabei lediglich, daß die Forderungen aus 4.2.1. erfüllt werden. Als zusätzliche Komponente zur Erzeugung einer kanonischen Numerierung wird im folgenden Ansatz die Klassifizierungsfunktion $\mathcal{kl}$ verwendet.

4.2.2.1. Brute-Force Methode

Der einfachste Weg zum Auffinden einer kanonischen Numerierung ist wiederum das Testen aller Möglichkeiten. Dazu werden alle Permutationen auf den Instanzen der Trägermenge berechnet und mit Hilfe der Funktion $\mathcal{kl}$ klassifiziert. Neben dem ganz offensichtlichen Komplexitätsproblem, das sich aus diesem Ansatz ergibt, werden jedoch auch viele Eigenschaften der diskreten Struktur, die eine leichte Klassifizierung und damit auch einfache spezifische Numerierung erlauben würden, nicht betrachtet.

4.2.2.2. Ausnutzung der Automorphismengruppe

Eine zweite Möglichkeit besteht in der Bestimmung der Automorphismengruppe einer diskreten Struktur $\mathcal{S}$ nach dem Algorithmus aus Kapitel 4.1.. Hat man diese ermittelt, so berechnet man für alle Permutationen φ aus der Automorphismengruppe eine Bewertung der Permutation mit Hilfe der Methode $\mathcal{S}.\mathcal{kl}$. Das φ mit maximalem Wert des KL - ERGEBNISSES $\mathcal{S}.\mathcal{kl}(\varphi)$ sei dann die gewünschte kanonische Numerierung.

Die sich aus dem nun folgenden Algorithmus ergebende Numerierung ist identisch mit dieser Berechnung über die Automorphismengruppe. Der Vorteil liegt jedoch darin, daß nicht explizit alle Elemente berechnet werden müssen, sondern Permutationen, die nicht mehr zu einer kanonischen Numerierung führen, können bereits frühzeitig erkannt und müssen dann nicht mehr weiter verfolgt werden.

Um den Zusammenhang zwischen dem Isomorphiealgorithmus und dem Algorithmus zur kanonischen Numerierung deutlicher zu machen, betrachten wir nochmals das Diagramm aus dem Beweis zu Satz 3.3.5.:

$$\begin{tikzcd} \mathcal{S}_1.S \arrow[r, "\sigma_1"] \arrow[d, "\varphi"'] & \mathcal{S}_1.S \arrow[d, "id"] \\ & \end{tikzcd}$$
$$\mathcal{S}_1.S \xrightarrow{\sigma_2} \mathcal{S}_1.S$$

Bei der Isomorphiebestimmung wird die Invarianz der Isomorphiedefinition bezüglich einer parallel durchgeführten, mit φ im Sinne des Diagrammes verträglichen Umnumerierung der betrachteten diskreten Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ ausgenutzt.

Die Permutationen σ_1 und σ_2 numerieren die Trägerinstanzen von $\mathcal{S}_1$ und $\mathcal{S}_2$ so, daß durch die Identität ein Isomorphismus zwischen den Strukturen definiert wird.

Dies ist jedoch genau die Eigenschaft, die bei einer kanonischen Numerierung gewünscht wird.

Der Algorithmus zur Isomorphiebestimmung berechnet also parallel für $\mathcal{S}_1$ und $\mathcal{S}_2$ eine kanonische Numerierung und prüft dabei in jedem Schritt deren Verträglichkeit. Die Numerierung selbst ist jedoch unabhängig von der jeweils anderen Struktur, d.h. es wäre auch möglich, zuerst die Numerierung zu berechnen und erst danach die Verträglichkeit zu testen, was natürlich mehr Aufwand im Einzelfall

bedeutet und deshalb im Algorithmus nicht gemacht wird. Möchte man jedoch eine Struktur in einer Datenbank halten, so geht man in der Regel davon aus, daß mehrere Isomorphietests mit dieser Struktur benötigt werden. In diesem Fall berechnet man einmal die kanonische Numerierung und muß in Zukunft bei einem Isomorphietest lediglich die Verträglichkeit der Strukturen prüfen. Ein Algorithmus dazu ist in 4.2.5. beschrieben.

4.2.3. Algorithmus

Betrachtet wird die diskrete Struktur $\mathcal{S}$ gemäß der Definitionen in Kapitel 3.2.. Das folgende Diagramm zeigt wieder den Ablauf des Algorithmus, der sehr ähnlich zu dem der Isomorphieberechnung ist.

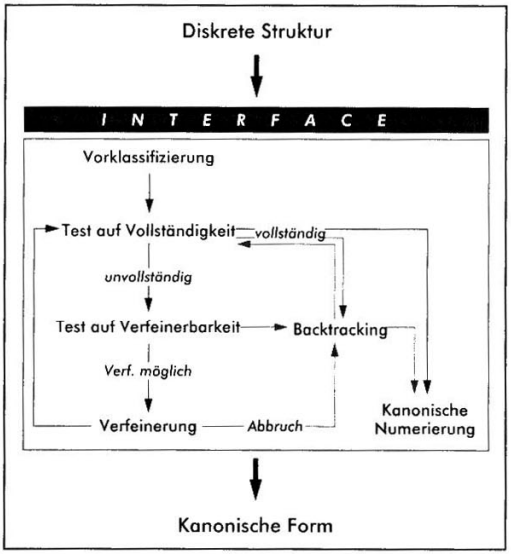


Abbildung 14: *Schematischer Ablauf der kanonischen Numerierung*

Betrachten wir die Methode $\mathcal{kl}$ zur Klassifizierung der aktuell vorliegenden Numerierung. Sei Δ eine Instanz der Klasse KL - ERGEBNIS. Δ behält im Laufe des Algorithmus immer den aktuellen maximalen Rückgabewert der Methode $\mathcal{kl}$ auf der Menge aller (partiellen) Numerierungen der Trägermenge von $\mathcal{S}$. Zu Beginn des Algorithmus wird Δ auf einen (bzgl. der Vergleichsoperatoren in KL - ERGEBNIS) beliebig kleinen Wert gesetzt, so daß jede zuerst gefundene Numerierung einen höheren Wert ergibt.

Gesucht ist nun eine Numerierung σ, welche die Forderungen **CAN1** - **CAN3** einer kanonischen Numerierung erfüllt.

4.2.3.1. Vorklassifizierung

Berechne die Vorklassifizierung $\mathcal{S}.\mathcal{K}^0$ der Struktur $\mathcal{S}$. Führe den folgenden Algorithmus aus:

Für alle $A \in \mathcal{S}.\mathcal{K}_i^0$, $1 \le i \le |\mathcal{S}.\mathcal{K}^0|$ mit $|\mathcal{S}.\mathcal{K}_i^0| = 1$

setze $\sigma := \sigma \cdot (A \rightarrow |\sigma| + 1)$

Setze $\Delta := \mathcal{K}(\sigma)$

Setze danach $\mathcal{S}.\mathcal{K}_i^a := \mathcal{S}.\mathcal{K}_i^0$. V_{max} nehme wiederum die Anzahl der bereits zur Verfeinerung genutzten Zuordnungen aus σ auf. Setze V_{max} zunächst auf 0.

4.2.3.2. Test auf Vollständigkeit

Ist $\mathcal{S}.\mathcal{K}^a$ eine vollständige Klassifizierungen (siehe 3.2.2.3.), so wurde eine kanonische Numerierung von $\mathcal{S}$ gefunden. Um diese Numerierung direkt anzugeben, siehe 4.2.3.6.. Man unterscheidet nun in Abhängigkeit des Testaufrufes zwei Fälle:

1. Wurde dieser Test während eines Backtracklaufes aufgerufen, so wird der Algorithmus mit 4.2.3.5. fortgesetzt. Die vorliegende Numerierung muß dann noch nicht die optimalste sein.
2. Wurde kein Backtracking gestartet, so setze den Algorithmus mit 4.2.3.3. fort. Die vorliegende Numerierung ist dann eindeutig bestimmt.

Ist $\mathcal{S}.\mathcal{K}^a$ nicht vollständig, so verzweige zum nächsten Schritt.

4.2.3.3. Test auf Verfeinerbarkeit

Gilt $V_{max} < |\sigma|$, so wurde σ seit dem letzten Aufruf dieses Moduls durch mindestens eine Permutation $(V_i \rightarrow i+1)$ erweitert. Damit existiert eine Klasse $\mathcal{S}_1.\mathcal{K}_i^a$ mit $1 \le i \le |\mathcal{S}_1.\mathcal{K}^a|$, für die gilt: $|\mathcal{S}_1.\mathcal{K}_i^a| = |\{V_1\}| = 1$, wobei $\mathcal{S}_1.\mathcal{K}^a$ die aktuellen Klassifizierungen der beiden betrachteten Strukturen sind.

Damit ist es möglich eine Verfeinerung der Klassifizierung vorzunehmen. Erhöhe V_{max} um 1. Der nächste Schritt im Algorithmus (Verfeinerung) wird mit dem Index V_{max} aufgerufen.

Gibt es jedoch keine neuen einelementige Klasse in der aktuellen Klassifizierung, gilt also $V_{max} = |\sigma|$, dann kann keine Verfeinerung durchgeführt werden. In diesem Falle wird der Backtracking - Algorithmus gestartet. Dafür wird zum Modul Backtracking (4.2.3.5.) verzweigt.

4.2.3.4. Verfeinerung

Zur Verfeinerung wird die aktuelle Numerierung σ verwendet, wobei lediglich die ersten V_{max} Festlegungen berücksichtigt werden. Zu diesem Zwecke wird eine neue Permutation μ erzeugt, die alle Abbildungen $(V_i \rightarrow i+1)$ mit $i < V_{max}$ enthält, die auch in σ vorhanden sind

Schritt 1: Partitionierung

Ermittle die erste Klasse mit dem Index j ($1 \le j \le |\mathcal{S}.\mathcal{K}^a|$) in der aktuellen Klassifizierung, für die $|\mathcal{S}.\mathcal{K}_j^a| > 1$ gilt, und die bei diesem Aufruf der Verfeinerung noch nicht verwendet wurde. Diese Klasse wird nun bezüglich der in 3.2.2.11. eingeführten Funktion *ref* analog zu Schritt 1 in 4.1.5.4. partitioniert. Erzeuge dazu neue Klassen $\mathcal{S}.\mathcal{K}_{j_p}^a$ mit:

- $\mathcal{S}.\mathcal{K}_j^a = \bigcup_{p=1}^{p_j} \mathcal{S}.\mathcal{K}_{j_p}^a$

- $S.\mathcal{K}^a_{j_q} \cap S.\mathcal{K}^a_{j_r} = \emptyset \quad \forall\ 1 \le q, r \le p_j, q \ne r$
- $rel(\mu, A) =_R rel(\mu, B) \quad \forall\ A, B \in S.\mathcal{K}^a_{j_q},\ 1 \le q \le p_j$
- $rel(\mu, A) <_R rel(\mu, B) \quad \forall\ A \in S.\mathcal{K}^a_{j_q},\ B \in S.\mathcal{K}^a_{j_{q+1}}$

mit $1 \le q \le p_j - 1$

Schritt 2: Konstruktion

Die zuvor gewonnene Unterteilung der j - ten Klasse von S in p_j neue Teilklassen führt zu einer neuen Klassifizierung. (o.B.d.A. $j<p_j$)

Es gilt also $S.\mathcal{K}^{a+1} = (S.\mathcal{K}^a_{j_1}, \ldots, S.\mathcal{K}^a_{j-1}, S.\mathcal{K}^a_{j_1}, \ldots, S.\mathcal{K}^a_{j_q}, S.\mathcal{K}^a_{j+1}, \ldots, S.\mathcal{K}^a_{|S.\mathcal{K}^a|})$

und damit

$$|S.\mathcal{K}^{a+1}| = |S.\mathcal{K}^a| + p_j - 1$$

$S.\mathcal{K}^{a+1}$ ist die neue aktuellen Klassifizierung. Wie bereits bei der Ermittlung der Ähnlichkeitsabbildung ist darauf zu achten, daß alle Querverweise auf Klassen der vorherigen Klassifizierung bei der Konstruktion der neuen Klassifizierung aktuell bleiben.

Schritt 3: Bewertung

Die durch die neue Klassifizierung festgelegte neue Permutation wird nun mit Hilfe der Methode $\mathcal{kl}$ (siehe 3.2.2.16.) bewertet. Dies dient dazu festzustellen, ob durch eine Fortsetzung der zum aktuellen Zeitpunkt durch die Klassifizierung festgelegten Permutation überhaupt noch eine kanonische Numerierung bestimmt werden kann. Dies bedeutet, wenn bereits eine Numerierung gefunden wurde, die einen „besseres" Ergebnis der $\mathcal{kl}$ - Funktion aufweist, so ist es nicht notwendig weitere Verfeinerungen der aktuellen Klassifizierung zu bestimmen, da diese - durch die Eigenschaft der Fortsetzbarkeit von $\mathcal{kl}$ - letztendlich immer zu einer „schlechteren" Numerierung führen würde. Führe dazu den folgenden Algorithmus aus:

Für alle $S.\mathcal{K}^a_{j_q} = \{A\}$, wobei $1 \le q \le p_j$

  setze $\sigma := \sigma \cdot (A \to |\sigma| + 1)$

  Falls $\mathcal{kl}(\sigma) \ge_K \Delta$,

    Falls $\mathcal{kl}(\sigma) >_K \Delta$ setze $\Delta := \mathcal{kl}(\sigma)$

  sonst

    Brich den Algorithmus ab

Wurde der Algorithmus nicht vollständig durchlaufen, brich die Verfeinerung ab und gehe zu Modul Backtracking. Das Backtracking muß zuvor aufgerufen worden sein, da Δ bereits mit einem größeren Wert belegt war.

Schritt 4: Test

Ist die neue Klassifizierung erzeugt und im Sinne von Schritt 3 zulässig und sind noch Klassen in der aktuellen Klassifizierung, die aus mehr als einem Element bestehen und noch nicht mit Hilfe von V zur

Verfeinerung herangezogen worden sind, so gehe zurück zu Schritt 1. Rufe sonst das Modul Test auf Vollständigkeit auf.

4.2.3.5. Backtracking

Das Backtracking - Modul kann aus den folgenden Situationen heraus aufgerufen werden:

Situation 1: Erstmaliger Aufruf

In diesem Fall geschieht der Aufruf immer aus dem Modul 4.2.3.3. (Test auf Verfeinerbarkeit) aufgerufen. Es stehen dann keine einelementigen Klassen in der aktuellen Klassifizierung mehr zur Verfügung, mit denen eine Verfeinerung durchgeführt werden könnte. Mit Hilfe des Backtracking wird nun wiederum, wie beim Isomorphietest „künstlich", eine einelementige Klasse zur Fortsetzung des Algorithmus bereitgestellt:

Bestimme die Klasse $S.\mathcal{K}_j^a$ mit $|S.\mathcal{K}_j^a| = \min\limits_{1 \le r \le |S.\mathcal{K}^a|, |S.\mathcal{K}_r^a| > 1} \left(|S.\mathcal{K}_r^a|\right)$

Auch hier wird also diejenige Klasse mit der geringsten Anzahl von Elementen gewählt. Ein Index j mit der obigen Eigenschaft existiert immer, da die Routine aus dem Test auf Verfeinerung aufgerufen worden ist. Die Klasse $S.\mathcal{K}_j^a$ wird nun in zwei neue Klassen gesplittet, wobei eine davon einelementig ist:

Wähle dazu $A \in S.\mathcal{K}_j^a$ wie beim Isomorphietest zunächst als minimales Indexelement der Klasse und setze $S.\mathcal{K}_{j_1}^a = \{A\}$ und $S.\mathcal{K}_{j_2}^a = S.\mathcal{K}_j^a \setminus \{A\}$. Erzeuge dann eine neue Klassifizierung der Form

$S.\mathcal{K}^{a+1} = (S.\mathcal{K}_1^a, \ldots, S.\mathcal{K}_{j-1}^a, S.\mathcal{K}_{j_1}^a, S.\mathcal{K}_{j_2}^a, S.\mathcal{K}_{j+1}^a, \ldots, S.\mathcal{K}_{|S.\mathcal{K}^a|}^a)$

mit $|S.\mathcal{K}^{a+1}| = |S.\mathcal{K}^a| + 2$.

Daraus ergeben sich insgesamt $|S.\mathcal{K}_j^a|$ Möglichkeiten, die Klasse $S.\mathcal{K}_{j_1}^a$ auszuwählen, wenn man beim nächsten Aufruf das minimale Element größer dem vorherigen bestimmt, und damit ebensoviele mögliche Backtrackschritte. Nach der obigen Konstruktion der neuen Klasse $S.\mathcal{K}^{a+1}$ wird geprüft, ob für $\sigma := \sigma \cdot (A \to |\sigma| + 1)$ $\ell(\sigma) \ge_K \Delta$ gilt und setze, falls $\ell(\sigma) >_K \Delta$ $\Delta := \ell(\sigma)$. Analog zum Isomorphietest wird die Fortsetzbarkeit auch noch für eine evtl. auftretende einelementige Klasse $S.\mathcal{K}_{j_2}^a$ getestet. Rufe dann wieder das Modul 4.2.3.2. auf. Ist die Bedingung nicht erfüllt, so fahre mit Situation 2 fort. Diese Auswahl des nächsten Elements führt dann nicht mehr zu einer kanonischen Numerierung.

Situation 2: Aufruf aus Verfeinerung

Dieser Fall tritt auf, wenn in Modul 4.2.3.2. festgestellt wurde, daß die aktuelle Klassifizierung nicht mehr zu einer kanonischen Numerierung führen kann. Da ein Aufruf von dieser Stelle voraussetzt, daß Situation 1 bereits zuvor eingetreten ist, versuche durch die nächste mögliche Auswahl von $S.\mathcal{K}_{j_1}^a$ evtl. zu einer weiteren Numerierung zu finden. Konstruiere dafür wie in Situation 1 eine neue aktuelle Klassifizierung. Dazu ist es wiederum notwendig, die Klassifizierung $S.\mathcal{K}^a$ wiederherzustellen, die zum Zeitpunkt des Aufrufes von Situation 1 aktuell war. Bei der Programmierung des Algorithmus ist dabei darauf zu achten, daß alle verwendeten Variablen wieder mit den Werten belegt werden, die sie vor dem Aufruf von Situation 1 hatten. Diesmal wird jedoch nicht wie in Situation 1 das kleinste Klassenelement isoliert, sondern das jeweils nächstgrößere, so daß jedesmal neue einelementige Klassen erzeugt werden. Ist dann wie in Situation 1 beschrieben eine neue Klassifizierung $S.\mathcal{K}^{a+1}$ erstellt worden, die zu einer kanonischen Numerierung führen kann, so verzweige damit zum Test auf Vollständigkeit.

Sind alle Kombinationen ausgeschöpft, so beende das Backtracking nach dem Aufruf von 4.2.3.6. mit der aktuellen maximalen Numerierung Δ.

Situation 3: Aufruf nach vollständiger Klassifizierung

Diese Situation ist ähnlich zur vorherigen. Nur war in diesem Fall eine vollständige Klassifizierung möglich, d. h. es wurde eine zulässige Numerierung der gegebenen diskreten Struktur gefunden. Vergleiche den $\mathcal{kl}$ - Wert der Numerierung mit dem bisher maximalen. Ist der neue Wert geringer, so ist die aktuelle Numerierung „besser". Speichere dann die entsprechende Numerierung. Analog zu Situation 2 wird, wenn möglich, die nächste Klassifizierung $S.\mathcal{K}^{a+1}$ erzeugt und getestet. Muß keine weitere Klassifizierung getestet werden, beende das Backtracking nach dem Aufruf von 4.2.3.6. mit der aktuellen maximalen Numerierung Δ.

4.2.3.6. Kanonische Numerierung

Dieses Modul wird aus dem Backtracking aufgerufen oder wenn in 4.2.3.2. eine vollständige Klassifizierung für S ermittelt wurde. Das bedeutet, daß durch den Algorithmus eine Numerierung der Elemente der Trägermenge gefunden wurde. Die Numerierung ist in σ gespeichert und entspricht damit der Reihenfolge, in der einelementige Klassen in der aktuellen Klassifizierung erzeugt wurden.

4.2.3.7. Satz

Der Algorithmus liefert eine kanonische Numerierung entsprechend den Forderungen aus 4.2.1.

Beweis:

Der Beweis ergibt sich wiederum konstruktiv aus dem Aufbau der Numerierung σ aus dem Algorithmus. σ erfüllt insbesondere auch Forderung **CAN1** mit dem Isomorphismus σ, denn: Sei die Struktur S gegeben. S_σ sei S mit der durch σ permutierten Trägermenge. Dann gilt offensichtlich σ ist Isomorphismus von S nach S_σ.

Durch die Vorklassifizierung, die Verfeinerung bezüglich des Ergebnisses der Methode *rel* und die Methode $\mathcal{kl}$ zur Bewertung der aktuellen Numerierung werden die Trägerinstanzen auf eindeutige Art im Algorithmus klassifiziert. Die Verträglichkeit von *rel* und $\mathcal{kl}$ bezüglich der aktuellen Numerierung σ werden überflüssige Wege im Backtrackbaum frühzeitig eliminiert.

4.2.4. Beispiel

Die einzelnen Schritte der kanonischen Numerierung sollen auch hier am bereits vorgestellten Beispiel des gefärbten Würfels verdeutlicht werden.

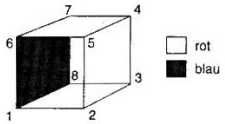


Abbildung 15: *Kanonische Numerierung Testbeispiel*

Als Schnittstelle dienen die bereits in 3.2.2.7. verwendeten Funktionen. Die aktuelle Numerierung wird aus technischen Gründen im Laufe des Algorithmus mit σ^{-1} bezeichnet.

4.2.4.1. Algorithmus

Schritt 1: Vorklassifizierung

Es gilt analog zur Ähnlichkeitsbestimmung: $\mathcal{S}.\mathcal{K}^0 = (\{2,3\},\{1,8\},\{4,5\},\{6,7\})$.

Da noch keine einelementigen Klassen existieren, gilt $\Delta = \min_{\mathcal{M}}$. σ enthält noch keine Permutationen. $V_{max} = 0$.

Schritt 2: Test auf Vollständigkeit

Da in der aktuellen Klassifizierung Klassen mit mehr als einem Element existieren, kann keine vollständige Klassifizierung vorliegen. Gehe zum nächsten Schritt.

Schritt 3: Test auf Verfeinerbarkeit

Da $V_{max} = |\sigma| = 0$, also keine Klasse in $\mathcal{S}_1.\mathcal{K}^0$ existiert, die aus nur einem Element besteht, kann keine Verfeinerung durchgeführt werden. Deshalb wird der Backtrackalgorithmus gestartet.

Schritt 4: Backtracking

Alle Klassen in $\mathcal{S}.\mathcal{K}^0$ sind gleich groß. Deshalb wird die erste Klasse mit den Elementen { 2, 3 } aufgespalten, woraus sich wiederum zwei mögliche Backtrackschritte ergeben. Im ersten Schritt ergibt die Aufspaltung bezüglich des Minimalelements 2 die neue Klassifizierung $\mathcal{S}.\mathcal{K}^1 = (\{2\},\{3\},\{1,8\},\{4,5\},\{6,7\})$. Für $\sigma = \sigma \cdot (2 \to 1)$ gilt $\mathcal{K}\ell(\sigma) >_K \Delta$. Setze $\Delta := \mathcal{K}\ell(\sigma)$. Da die aufgespaltete Klasse lediglich aus 2 Elementen bestand wird noch $\sigma = \sigma \cdot (3 \to 2)$ hinzugefügt. Auch dafür gilt $\mathcal{K}\ell(\sigma) \geq_K \Delta$. Setze $\Delta := \mathcal{K}\ell(\sigma)$.

Mit dieser Klassifizierung wird wieder der Test auf Vollständigkeit aufgerufen. Um zu verdeutlichen, daß sich die folgenden Schritte aus der Backtrackkombination 1 heraus aufgerufen wurden, werden sie mit B1 gekennzeichnet.

Schritt 5 / B1: Test auf Vollständigkeit

Analog zu Schritt 2.

Schritt 6 / B1: Test auf Verfeinerbarkeit

Durch den Backtrackalgorithmus sind σ zwei neue Abbildungen hinzugefügt worden. Da weiterhin $V_{max} = 0$ ($< |\sigma| = 2$) gilt, kann also eine Verfeinerung durchgeführt werden. Gehe zu Schritt 7 mit $V_{max} = 1$.

Schritt 7 / B1: Verfeinerung

Zur Verfeinerung wird, wie beschrieben aus $\sigma = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix}$ die neue Permutation $\mu = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ erstellt (V_{max}=1), bezüglich der mit der Methode *rel* verfeinert wird. Die nächste Klasse in $\mathcal{S}.\mathcal{K}^1$ bestehend aus mehr als einem Element ($\mathcal{S}.\mathcal{K}^1_3 = \{1,8\}$) wird aufgespalten. Folgende Ergebnisse der Methode *rel* wer-

den beim Vergleich von V mit den Elementen $s.\mathcal{K}_3^1$ erzielt:.

Struktur 1	
$ref(\mu, 1)$	$(1, 1)$
$ref(\mu, 8)$	$(0, 1)$

Da $ref(\mu, 1) >_R ref(\mu, 8)$, wird $s.\mathcal{K}^1$ aufgeteilt in

$s.\mathcal{K}^1 = (\{2\}, \{3\}, \underbrace{\{8\}, \{1\}}, \{4, 5\}, \{6, 7\})$. Die Bedingungen aus 4.2.3.4. / Schritt 1 sind dabei erfüllt.

Die nachfolgende Konstruktion ergibt $s.\mathcal{K}^2 = s.\mathcal{K}^1$.

Im nächsten Schritt wird die Bewertung der aktuellen Numerierung durchgeführt. Die Methode $\mathcal{kl}$ wird aus 3.2.3.6.4. übernommen. Durch die Verfeinerung wurden zwei neue einelementige Klassen erzeugt. Es wird der Algorithmus aus 4.2.3.4. / Schritt 3 durchlaufen. Da alle Festlegungen verträglich mit Δ sind, gilt danach:

$$\sigma^{-1} = /2\,3\,8\,1\,*/$$

und eine Instanz K der Klasse KL - ERGEBNIS enthält das Ergebnis der Funktion $\Upsilon_p(\sigma(M))$ als Attribut durch den Aufruf von $\mathcal{kl}(\sigma)$. Es gilt dann $K \geq_K \Delta$.

M ist dabei die Adjazenzmatrix des Graphen ist, die den gefärbten Würfel bestimmt.

4.2.3.4. / Schritt 3 wird aufgerufen und die Klassen $s.\mathcal{K}_5^2$ und $s.\mathcal{K}_6^2$ müssen noch mit Hilfe von μ verfeinert werden.

Für $s.\mathcal{K}_5^2$ gilt analog zu oben:

Struktur 1	
$ref(\mu, 4)$	$(0, 1)$
$ref(\mu, 5)$	$(1, 1)$

Damit wird $s.\mathcal{K}^2$ aufgeteilt in

$s.\mathcal{K}^2 = (\{2\}, \{3\}, \{8\}, \{1\}, \underbrace{\{4\}, \{5\}}, \{6, 7\})$. Die Forderungen aus 4.2.3.4. / Schritt 1 und Schritt 3 sind erfüllt. Damit ergibt sich die neuen Klassifizierungen $s.\mathcal{K}^3 = s.\mathcal{K}^2$ und die Permutation .

$$\sigma^{-1} = /2\,3\,8\,1\,4\,5\,*/$$

sowie $\Delta := \mathcal{kl}(\sigma)$.

Für $s.\mathcal{K}_5^3 (= s.\mathcal{K}_6^2)$ gilt dann:

Struktur 1	
$ref(\mu, 6)$	$(0, 1)$
$ref(\mu, 7)$	$(0, 0)$

Damit wird $s.\mathcal{K}^3$ aufgeteilt in

$s.\mathcal{K}^3 = (\{2\}, \{3\}, \{8\}, \{1\}, \{4\}, \{5\}, \underbrace{\{6\}, \{7\}})$.

Die Forderungen aus 4.2.3.4. / Schritt 1 - 3 sind wiederum erfüllt. Damit ergibt sich die neue Klassifizierung $\mathcal{S}.\mathcal{K}^4 = \mathcal{S}.\mathcal{K}^3$ und die Permutation.

$$\sigma^{-1} = /2\,3\,8\,1\,4\,5\,6\,7/$$

Da zwar V_{max} weiterhin den Wert 1 besitzt und inzwischen durch die bei der Verfeinerung neu entstandenen einelementigen Klassen $|\xi| = 8$ gilt, aber in $\mathcal{S}.\mathcal{K}^4$ keine Klassen mehr vorhanden sind, die nach 4.2.3.4. zu einer Verfeinerung herangezogen werden könnten, wird zum nächsten Schritt verzweigt.

Schritt 8 / B1: Test auf Vollständigkeit

$\mathcal{S}.\mathcal{K}^4$ ist eine vollständige Klassifizierung. Demnach wurde ein vollständige Numerierung σ der vorgegebenen Struktur $\mathcal{S}$ gefunden. Es gilt:

$$\sigma = /4\,1\,2\,5\,6\,7\,8\,3/$$

Da Schritt 8 aus einem Backtrackschritt (B1) aufgerufen wurde, wird wieder zum Backtracking verzweigt.

Schritt 9: Backtracking

Der zur vollständigen Numerierung σ gehörige kanonischen Wert der Klassifikationsfunktion $\mathcal{K}(\sigma)$ ist zulässig. Setze deshalb $\Delta := \mathcal{K}(\sigma)$ und speichere σ als σ_{can}. In Schritt 4 wurde die Klasse $\mathcal{S}.\mathcal{K}_1^0$ mit den Trägerinstanzen { 2, 3 } aufgespalten. Die erste der sich daraus ergebenden zwei Kombinationen wurde erfolgreich bearbeitet. Die zweite Kombinationsmöglichkeit führt zu der neuen Klassifizierung $\mathcal{S}.\mathcal{K}^1 = (\{3\}, \{2\}, \{1, 8\}, \{4, 5\}, \{6, 7\})$. Mit dieser neuen, zweiten möglichen Klassifizierung wird wieder der Test auf Vollständigkeit aufgerufen. Um zu verdeutlichen, daß die folgenden Schritte aus der Backtrackkombination 2 heraus aufgerufen wurden, werden sie mit B2 gekennzeichnet.

Schritt 10 / B2: Test auf Vollständigkeit

Analog zu Schritt 2.

Schritt 11 / B2: Test auf Verfeinerbarkeit

Die durch den Backtrackalgorithmus in Schritt 9 neu erzeugten Abbildungen können zur Verfeinerung verwendet werden, da $V_{max} = 0$ ($< |\sigma| = 2$). Gehe zu Schritt 12 mit dem $V_{max} = 1$.

Schritt 12 / B2: Verfeinerung

Zur Verfeinerung wird nun aus der aktuellen Zuordnung $\sigma = \begin{pmatrix} 3 & 2 \\ 1 & 2 \end{pmatrix}$ die neue Permutation $\mu = \begin{pmatrix} 3 & * \\ 1 & \end{pmatrix}$ erstellt. Analog zu Schritt 7 / B1 wird mit μ die Klasse $\mathcal{S}.\mathcal{K}_3^1 = \{1, 8\}$ aufgespalten. Es ergibt sich nach gleichem Vorgehen wie in Schritt 2 die neue Klassifizierung

$$\mathcal{S}.\mathcal{K}^2 = (\{3\}, \{2\}, \{1\}, \{8\}, \{4, 5\}, \{6, 7\}).$$

Nach der Verfeinerung von $\mathcal{S}.\mathcal{K}_5^2$ liegt die Klassifizierung

$$\mathcal{S}.\mathcal{K}^3 = (\{3\}, \{2\}, \{1\}, \{8\}, \underbrace{\{5\}, \{4\}}, \{6, 7\}) \text{ vor.}$$

Ein weiterer Verfeinerungsschritt ergibt

$$\mathcal{S}.\mathcal{K}^4 = (\{3\}, \{2\}, \{1\}, \{8\}, \{5\}, \{4\}, \underbrace{\{7\}, \{6\}}).$$

Danach kann keine weitere Verfeinerung durchgeführt werden, da die Klassifizierung vollständig ist. In jedem Schritt wird dabei natürlich die Konsistenz von $\mathcal{kl}(\sigma)$ bezüglich Δ getestet. Es gilt zum Schluß

$$\sigma^{-1} = /3\,2\,1\,8\,5\,4\,7\,6/$$

Schritt 13 / B2: Test auf Vollständigkeit

$\mathcal{S}.\mathcal{K}^4$ ist eine vollständige Klassifizierung von $\mathcal{S}$. Demnach wurde eine weitere Numerierung σ gefunden. Es gilt:

$$\sigma = /3\,2\,1\,6\,5\,8\,7\,4/$$

Da Schritt 13 aus einem Backtrackschritt (B2) aufgerufen wurde, wird wieder zum Backtracking verzweigt.

Schritt 14: Backtracking

In diesem Schritt wird die Klassifikationsfunktion der aktuellen Numerierung mit der bisherigen kanonischen verglichen. Es gilt in diesem Fall

$$\mathcal{kl}(\sigma_{can}) =_K \Delta =_K \mathcal{kl}(\sigma)$$

Da also σ nicht zu einer „besseren" Numerierung führt, wird es verworfen. Alle Kombinationsmöglichkeiten aus Schritt 4 sind ausgeführt worden. Damit wird das Backtracking und damit auch der Algorithmus durch den Aufruf von 4.2.3.6. beendet:

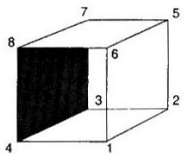
Schritt 15: Kanonische Numerierung

Die kanonische Numerierung kann nun anhand von σ_{can} ausgegeben werden. Es gilt:

$$\varphi = \sigma_{can} = /4\,1\,2\,5\,6\,7\,8\,3/$$

4.2.4.2. Kanonische Numerierung

Der auf diese Weise kanonisch numerierte gefärbte Würfel ist damit:



Man beachte, daß sich die zweite im Algorithmus gefundene, aber verworfene Numerierung durch einfache Anwendung des nichttrivialen Elements der Automorphismengruppe (id, / 8 3 2 5 4 7 6 1 /) aus 3.3.6.ermitteln läßt. Wendet man Die Permutation / 8 3 2 5 4 7 6 1 / auf die gefundene kanonische Numerierung an, so gilt:.

$$/4\,1\,2\,5\,6\,7\,8\,3/ \circ /8\,3\,2\,5\,4\,7\,6\,1/ = /3\,2\,1\,6\,5\,8\,7\,4/$$

Der Algorithmus „erkennt" demnach unter den getroffenen Definitionen TRÄGER, REL - ERGEBNIS, KL - ERGEBNIS, der *rel* und $\mathcal{kl}$ - Methode die Symmetrie des Würfels. Daher ist es auch letztendlich

unerheblich, ob Bild oder Spiegelbild kanonisch numeriert werden, da eine Unterscheidung zwischen diesen Numerierungen auf Basis der vorgegebenen Schnittstelle nicht möglich ist.

Es bleibt anzumerken, daß der im Algorithmus gespeicherte maximale Wert der Numerierung nicht der Maximalwert aller möglichen Numerierungen ist, der Algorithmus also nicht eine bezüglich KL - ERGEBNIS optimale Numerierung findet, sondern eine Optimale unter Berücksichtigung der übrigen Schnittstellen.

4.2.5. Anwendung zur Isomorphiebestimmung

Die kanonische Numerierung wird, wie bereits erwähnt, vor allem zur Vereinfachung des Algorithmus zur Bestimmung der Ähnlichkeit von Strukturen verwendet. Liegen zwei Strukturen S_1 und S_2 in einer kanonischen Form vor, so muß nicht der in 4.1. vorgestellte Algorithmus ausgeführt werden, um eine Ähnlichkeit feststellen zu können.

Der Grund hierfür liegt in der Definition der kanonischen Numerierung. Der Kernpunkt dieser speziellen Numerierung ist ihre Invarianz gegenüber der Anwendung von Isomorphismen. Geht man davon aus, daß S_1 und S_2 isomorph sind, so ist ihre aktuelle, kanonische Anordnung der Trägerinstanzen invariant bezüglich Isomorphie. Also auch invariant bezüglich der Identität, die trivialerweise Element der Automorphismengruppe von S_1 und S_2 ist.

Man betrachtet also den Isomorphismus id und überprüft, ob eine Abbildung der Trägerinstanzen zweier kanonisch numerierter Strukturen durch die Identität mit den Isomorphiebedingungen **ISO1**-**ISO4** verträglich ist.

Benutze dazu den folgenden Algorithmus:

Falls $|S_1.S| \neq |S_2.S|$, brich den Algorithmus ab.

Sei σ_1 die kanonische Numerierung von S_1.S und σ_2 diejenige von S_2.S.

Setze $\varphi(i) := \sigma_2^{-1}(\sigma_1(i)) \;\; \forall\, 1 \leq i \leq |S_1.S|$.

Falls **ISO4** nicht erfüllt, brich den Algorithmus ab.

Für alle $A \in S_1.S$

  Falls $A \neq_S \varphi(A)$, brich den Algorithmus ab.

  Für alle $B \in S_2.S$

    Falls **ISO3** nicht erfüllt, brich ab.

Wurde der Algorithmus durchlaufen, so sind S_1 und S_2 ähnlich. Die Anordnung der Tests spielt dabei keine Rolle. Das heißt, wenn der Anwender weiß, daß der Test bezüglich **ISO4** in der Regel keine Unterscheidungsmerkmale liefert, kann er diesen auch am Ende des Algorithmus plazieren. Es müssen jedoch zur vollständigen Konsistenz alle Tests vorhanden sein.

Wie man sieht, benötigt der Algorithmus maximal quadratischen Aufwand zur Feststellung der Ähnlichkeit. Obwohl damit natürlich eine erhebliche Vereinfachung der Vollstruktursuche zu erzielen ist, müßten beispielsweise alle Strukturen einer Datenbank sequentiell verglichen mit diesem Algorithmus verglichen werden, um alle ähnliche Strukturen zu finden. Daß dies zwar möglich, dem Benutzern aber aufgrund der zu erwartenden Antwortzeiten nicht zuzumuten.

Deshalb muß man zusätzliche Verfahren zur Einschränkung der zu betrachtenden Kandidaten anwenden, bevor dieser direkte Vergleich durchgeführt wird. Wie diese Verfahren in der Praxis aussehen, wird in Kapitel 6.4. dokumentiert.

Man beachte, daß obiger Algorithmus nicht dazu geeignet ist, die Automorphismengruppe der betrachteten Struktur zu berechnen. Er stellt lediglich fest, ob zwei Strukturen identisch sind. Ob es neben der Identität noch weitere Abbildungen der beiden Trägermengen aufeinander gibt, ist in diesem Fall nicht von Bedeutung.

4.2.6. Anwendung zur Bestimmung der Automorphismengruppe

Wird der allgemeine Algorithmus zur Isomorphieerkennung auch zur Bestimmung der Automorphismengruppe eingesetzt, so müssen alle Schritte des Algorithmus für Struktur 1 und Struktur 2 parallel vorgenommen werden. Da in diesem Fall Struktur 1 = Struktur 2 gilt, wird durch die Zerlegung der zweiten Struktur unnötiger Aufwand betrieben, da die Zerlegung immer identisch zu der aus Struktur 1 ist.

Wie bereits erwähnt entfällt dieser Aufwand beim sonst identischen Algorithmus zur kanonischen Numerierung.

Wendet man also den Algorithmus zur Bestimmung einer kanonischen Numerierung auf eine Struktur an und merkt sich alle möglichen vollständigen Numerierungen, auch wenn die Methode $\mathcal{K}\mathcal{L}$ bezüglich dieser Numerierungen nicht immer größere KL - ERGEBNISSe als den bisherigen Maximalwert Δ liefert, so bilden diese Numerierungen die Automorphismengruppe der betrachteten diskreten Struktur.

4.3. Teilstrukturisomorphie

In verschiedenen Anwendungsbereichen spielt die Frage der Teilstrukturisomorphie eine große Rolle. Das bedeutet, es muß festgestellt werden, ob eine diskrete Struktur in einer anderen enthalten ist. Bei den graphentheoretischen Grundlagen wurde bereits eine Definition von Teilgraphen gegeben. In ähnlicher Art ist es auch möglich, Teilstrukturen einer diskreten Struktur zu definieren

4.3.1. Definition

Seien $\mathcal{S}_1$ und $\mathcal{S}_2$ diskrete Strukturen gemäß den Schnittstellendefinitionen aus 3.3.1. Die Abbildung $\varphi : \mathcal{S}_1.S \rightarrow \mathcal{S}_2.S$ ist genau dann ein Teilstruktur - Ähnlichkeitsabbildung von $\mathcal{S}_1$ auf $\mathcal{S}_2$, wenn φ injektiv und für alle $A, B \in \mathcal{S}_1.S$ gilt:

PISO1 $|\mathcal{S}_1.S| \leq |\mathcal{S}_2.S|$

PISO2 $A =_S \varphi(A)$

PISO3 $\mathcal{S}_1.rel(\mathrm{id}, B) =_R \mathcal{S}_2.rel(\varphi, \varphi(B))$

PISO4 $\mathcal{S}_1.\Gamma(\mathrm{id}) =_G \mathcal{S}_2.\Gamma(\varphi)$

Mit dieser Definition der Teilstrukturisomorphie lassen sich für Graphen sowohl Subgraphen- als auch Teilgraphenisomorphie darstellen.

Analog zu Satz 3.3.5. ist diese Definition äquivalent zu folgender Aussage:

Gegeben seien zwei Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ mit den Voraussetzungen aus Definition 3.3.1.. Seien weiterhin $\sigma_1 \in \mathcal{E}_{|\mathcal{S}_1.S|} \subseteq S_{|\mathcal{S}_1.S|}$ eine Umnumerierung von $\mathcal{S}_1.S$ und analog σ_2 eine Umnumerierung von $\mathcal{S}_2.S$ und eine Abbildung $\varphi : \mathcal{S}_1.S \rightarrow \mathcal{S}_2.S$ mit $\sigma_2^{-1} \circ \sigma_1 = \varphi$ gegeben. Dann gilt: φ ist genau dann eine Teilstruktur - Ähnlichkeitsabbildung zwischen $\mathcal{S}_1$ und $\mathcal{S}_2$, wenn die Identität Teilstruktur - Ähnlichkeitsabbildung zwischen den durch σ_1 und σ_2 neu numerierten Strukturen ist.

σ_2 bildet die Trägerinstanzen, die nicht im Wertebereich von σ_1 liegen trivialerweise auf sich selbst ab. Das an den Algorithmus anschließende Beispiel zeigt Teilgraphenisomorphie. Im späteren Kapitel über Moleküle beschäftigt sich ein Abschnitt 5.2. jedoch auch mit der Subgraphenisomorphie.

4.3.2. Algorithmus

Der Algorithmus zur Erkennung von Teilstrukturisomorphie ist sehr ähnlich zum Algorithmus für Vollstrukturisomorphie. Der wesentliche Unterschied besteht darin, daß die Bedingungen für das Auftreten einer Teilstrukturisomorphie nicht so strikt wie für Vollstrukturisomorphie sind. Dies folgt auch aus obiger Definition. Weitere Hinweise zur Teilstrukturproblematik bei Molekülen sind auch in 6.2.5. beschrieben. Das Diagramm, das den Ablauf des Algorithmus zur Ermittlung der Teilstrukturisomorphie wiedergibt ist identisch zu dem der Vollstrukturisomorphie. Die Unterschiede werden lediglich in den einzelnen Stationen des Algorithmus deutlich. Zur Orientierung im Algorithmus betrachte man also Abbildung 11:.

Gegeben seien wiederum zwei Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ mit ihren entsprechenden standardisierten Schnittstellen. Gesucht sind alle möglichen Abbildungen φ, welche die Menge der Trägerinstanzen $\mathcal{S}_1.S$ auf $\mathcal{S}_2.S$ abbilden, unter Berücksichtigung der, der Eigenschaften **PISO1 - PISO4**.

Es soll also ermittelt werden, welche Einbettungen von $\mathcal{S}_1.S$ in $\mathcal{S}_2.S$ existieren. Der Algorithmus verwendet wiederum eine Funktion $\mathcal{F} : \mathcal{S}_1.\mathcal{K}^a \rightarrow \mathcal{S}_2.\mathcal{K}^a$, welche die Zuordnung der Klassen der Referenzstruktur zu Klassen der zu testenden Struktur, und letztendlich auch die Einbettung selbst darstellt.

4.3.2.1. Vorklassifizierung

Berechne für die Strukturen S_1 und S_2 die Vorklassifizierungen $S_1.\mathcal{K}^0$ und $S_2.\mathcal{K}^0$ der Trägerinstanzen.

4.3.2.1.1. Satz (Notwendige Bedingungen)

Falls eine Einbettung von S_1 in S_2 existiert, so muß für $S_1.\mathcal{K}^0$ und $S_2.\mathcal{K}^0$ gelten:

1. $|S_1.\mathcal{K}^0| \leq |S_2.\mathcal{K}^0|$

2. $\forall S_1.\mathcal{K}_i^0\ \exists! S_2.\mathcal{K}_j^0$ mit
   a) $A =_S B$ für $A \in S_1.\mathcal{K}_i^0, B \in S_2.\mathcal{K}_j^0$
   b) $|S_1.\mathcal{K}_i^0| \leq |S_2.\mathcal{K}_j^0|$

Beweis:

Der Beweis des Satzes erfolgt analog zum Beweis von 4.1.5.1.1..

Das bedeutet, jeder Klasse der Teilstruktur muß genau eine Klasse der Referenzstruktur zugeordnet werden können. Diese muß mindestens so viele Elemente besitzen, wie die Klasse der Referenzstrukturen. Die Trägerinstanzen müssen bezüglich ihrer Vergleichsoperatoren übereinstimmen.

Sind die Bedingungen 4.3.2.1.1. nicht erfüllt, so endet der Algorithmus. Es existiert dann keine Einbettung von S_1 in S_2.

Setze $S_2.\mathcal{K}_i^a := S_2.\mathcal{K}_i^0$ $S_1.\mathcal{K}_i^a := S_1.\mathcal{K}_i^0$.

Sei $\mathcal{F} : S_1.\mathcal{K}^a \rightarrow S_2.\mathcal{K}^a$ wie beim Isomorphietest definiert. Mit $(S_1.\mathcal{K}_i^a, S_2.\mathcal{K}_j^a)$ seien die Forderungen des Satzes erfüllt. Setze $\mathcal{F}(S_1.\mathcal{K}_i^a) = S_2.\mathcal{K}_j^a$. Analog dazu seien $\xi \in S_{|S_1.S|}$ und die beiden Numerierungen $\sigma_1, \sigma_2 \in S_{|S_1.S|}$ definiert.

Desweiteren gilt auch für eine Teilstruktur - Ähnlichkeitsabbildung Bemerkung 4.1.5.1.2..

Wurden bei der Vorklassifizierung sowohl bei S_1 als auch bei S_2 einelementige Klassen erzeugt, müssen **PISO3** und **PISO4** getestet werden. Dies geschieht wie in 4.1.5.1. mit dem folgenden Algorithmus:

Für alle $A \in S_1.\mathcal{K}_i^a$, $1 \leq i \leq |S_1.\mathcal{K}^a|$ mit $|S_1.\mathcal{K}_i^a| = 1$

Sei $\{A'\} = \mathcal{F}(\{A\})$

Setze $\xi := \xi \cdot (A \rightarrow A')$

Setze $\sigma_1 := \sigma_1 \cdot (A \rightarrow |\sigma_1| + 1)$ und $\sigma_2 := \sigma_2 \cdot (A' \rightarrow |\sigma_2| + 1)$

Für alle $B \in S_1.\mathcal{K}_j^a$, $1 \leq j \leq |S_1.\mathcal{K}^a|$ mit $|S_1.\mathcal{K}_j^a| = 1$

Sei $\{B'\} = \mathcal{F}(\{B\})$

Falls $S_1.rel(\sigma_1, B) \neq_R S_2.rel(\sigma_2, B')$

dann brich den Algorithmus ab. Es gibt in diesem Fall keine Ähnlichkeit zwischen von S_1 auf S_2.

Wurde der Algorithmus nicht abgebrochen, muß noch **PISO4** getestet werden. ξ enthält dann alle bisher festgelegten Transpositionen. Ist auch die Bedingung

$$S_1.\Gamma(\sigma_1) =_G S_2.\Gamma(\sigma_2)$$

erfüllt, kann eine Fortsetzung φ von ξ existieren, so daß $\varphi : \mathcal{S}_1.S \rightarrow \mathcal{S}_2.S$ ein Teilstrukturisomorphismus von $\mathcal{S}_1$ auf $\mathcal{S}_2$ ist.

Die Variable V_{max} wird zu diesem Zeitpunkt wieder auf 0 gesetzt. In ihr wird später die maximale Anzahl der für eine Verfeinerung bereits herangezogenen Festlegungen in ξ gespeichert.

4.3.2.2. Test auf Vollständigkeit

Ist $\mathcal{S}_1.\mathcal{K}^a$ eine vollständige Klassifizierung, d.h. $|\mathcal{S}_1.\mathcal{K}_i^a| = 1 \quad \forall\, 1 \le i \le |\mathcal{S}_1.\mathcal{K}^a|$, und gilt $|\mathcal{F}(\mathcal{S}_1.\mathcal{K}_i^a)| = 1$ $\forall\, 1 \le i \le |\mathcal{S}_1.\mathcal{K}^a|$, wurde eine Einbettung von $\mathcal{S}_1$ in $\mathcal{S}_2$ gefunden.

Es werden wieder in Abhängigkeit des Testaufrufes zwei Fälle unterschieden:

1. Wurde dieser Test während eines Backtracklaufes aufgerufen, so wird der Algorithmus mit 4.3.2.5. fortgesetzt. Die vorliegende Einbettung muß dann nicht die einzig mögliche sein.
2. Wurde kein Backtracking gestartet, so setze den Algorithmus mit 4.3.2.6. fort. Die vorliegende Einbettung ist dann eindeutig bestimmt.

Ist die obige Bedingung nicht erfüllt, so verzweige zum nächsten Schritt (Test auf Verfeinerbarkeit).

4.3.2.3. Test auf Verfeinerbarkeit

Um eine Verfeinerung durchführen zu können, muß ein Element der Referenzstruktur bereits fest auf ein Element der Teststruktur abgebildet worden sein.

Gilt $V_{max} < |\xi|$, so wurde ξ seit dem letzten Aufruf dieses Moduls durch mindestens eine Permutation $(V_1 \rightarrow V_2)$ erweitert. Damit existiert eine Klasse $\mathcal{S}_1.\mathcal{K}_i^a$ mit $1 \le i \le |\mathcal{S}_1.\mathcal{K}^a|$, für die gilt: $|\mathcal{S}_1.\mathcal{K}_i^a| = |\{V_1\}| = 1$ (und damit aufgrund des Algorithmus auch $|\mathcal{F}(\mathcal{S}_1.\mathcal{K}_i^a)| = |\mathcal{S}_2.\mathcal{K}_j^a| = |\{V_2\}| = 1$), wobei $\mathcal{S}_1.\mathcal{K}^a$ und $\mathcal{S}_2.\mathcal{K}^a$ die aktuellen Klassifizierungen der beiden betrachteten Strukturen sind. Erhöhe in diesem Fall V_{max} um 1.

Hierzu ist zu bemerken, daß aufgrund der Vorklassifizierung und der Zuordnung von Klassen bei der späteren Verfeinerung nur dann $|\mathcal{F}(\mathcal{S}_1.\mathcal{K}_i^a)| = 1$ sein kann, wenn auch $|\mathcal{S}_1.\mathcal{K}_i^a| = 1$ gilt.

Gibt es jedoch keine neue einelementige Klasse in der aktuellen Klassifizierung, kann keine Verfeinerung durchgeführt werden. In diesem Falle gilt $V_{max} = |\xi|$ und es wird der Backtracking - Algorithmus (4.3.2.5.) gestartet.

4.3.2.4. Verfeinerung

Zur Verfeinerung werden nun wiederum die partiellen Permutationen σ_1 und σ_2 herangezogen, wobei lediglich die ersten V_{max} Festlegungen berücksichtigt werden.

Die Verfeinerung besteht aus folgenden Schritten:

Schritt 1: Partitionierung

Ermittle zuerst die erste verfügbare Klasse mit dem Index q ($1 \le q \le |\mathcal{S}_1.\mathcal{K}^a|$) in der aktuellen Klassifizierung der Referenzstruktur, für die $|\mathcal{S}_1.\mathcal{K}_q^a| > 1$ gilt, und die bei diesem Aufruf der Verfeinerung noch nicht verwendet wurde. Sei $\mathcal{S}_2.\mathcal{K}_p^a := \mathcal{F}(\mathcal{S}_1.\mathcal{K}_q^a)$. Diese Klasse wird nun bezüglich der in 3.2.2.11. eingeführten Funktion *rel* partitioniert. Erzeuge dazu aus σ_1 eine neue Permutation μ_1, in der alle bisherigen Permutationen $(V_i \rightarrow i)$, $i < V_{max}$ enthalten sind und bestimme die Anzahl k_q der möglichen Teilmengen der Klasse $\mathcal{S}_2.\mathcal{K}_q^a$ wie folgt:

$$\mathcal{S}_1.\mathcal{K}_q^a = \bigcup_{k=1}^{k_q} \mathcal{S}_1.\mathcal{K}_{q_k}^a$$

$$\mathcal{S}_1.\mathcal{K}_{q_r}^a \cap \mathcal{S}_1.\mathcal{K}_{q_s}^a = \emptyset \quad \forall\, 1 \le r, s \le k_q,\ r \ne s$$

$$rel(\mu_1, A) =_R rel(\mu_1, B) \quad \forall\, A, B \in \mathcal{S}_1..\mathcal{X}^a_{q_k},\ 1 \le k \le k_q$$

$$rel(\mu_1, A) <_R rel(\mu_1, B) \quad \forall\, A \in \mathcal{S}_1.\mathcal{X}^a_{q_k},\ B \in \mathcal{S}_1.\mathcal{X}^a_{q_{k+1}}$$

mit $1 \le k \le k_q - 1$

Analog dazu wird mit Hilfe von μ_2 und *rel* die Klasse $\mathcal{F}(\mathcal{S}_1.\mathcal{X}^a_q)$ der zu testenden Struktur in k_p disjunkte Teilmengen zerlegt. Da die bezüglich der Referenzstruktur zu überprüfende Teststruktur $\mathcal{S}_2$ bei der Teilstrukturisomorphie in der Regel in mehr Klassen zerfällt als $\mathcal{S}_1$, d.h. im allgemeinen gilt $k_q < k_p$, müssen die auf beiden Seiten entstehenden neuen Klassen neu zugeordnet werden.

Schritt 2: Identifizierung

Um eine Teilstrukturisomorphie von $\mathcal{S}_2$ in $\mathcal{S}_1$ zu gewährleisten, müssen die folgenden Bedingungen erfüllt sein:

- $k_p \ge k_q$
- $\forall\, \mathcal{S}_1.\mathcal{X}^a_{q_k},\ 1 \le k \le k_q\ \exists!\, \mathcal{S}_2.\mathcal{X}^a_{p_r},\ 1 \le r \le k_p$, so daß gilt:
- $rel(\mu_1, A) =_R rel(\mu_2, B) \quad \forall\, A \in \mathcal{S}_1.\mathcal{X}^a_{q_k},\ B \in \mathcal{S}_2.\mathcal{X}^a_{p_r}$
- $|\mathcal{S}_2.\mathcal{X}^a_{p_r}| \ge |\mathcal{S}_1.\mathcal{X}^a_{q_k}|$

Für die Überprüfung dieser Bedingung führe man den folgenden Algorithmus aus:

Für alle $\mathcal{S}_1.\mathcal{X}^a_{q_k}$, $1 \le k \le k_q$

  Setze $A \in \mathcal{S}_1.\mathcal{X}^a_{q_k}$ beliebig

  Falls kein oder mehrere $\mathcal{S}_2.\mathcal{X}^a_{p_r}$, $1 \le r \le k_p$ existieren mit:

    $\mathcal{S}_1.rel(\mu_1, A) =_R \mathcal{S}_2.rel(\mu_2, A')$ und $|\mathcal{S}_1.\mathcal{X}^a_{q_k}| \le |\mathcal{S}_2.\mathcal{X}^a_{p_r}|$

    für ein beliebiges $A' \in \mathcal{S}_2.\mathcal{X}^a_{p_r}$,

    dann brich den Algorithmus ab.

  sonst

    Setze $\mathcal{F}(\mathcal{S}_1.\mathcal{X}^a_{q_k}) := \mathcal{S}_2.\mathcal{X}^a_{p_r}$.

    Falls $\mathcal{S}_1.\mathcal{X}^a_{q_k} = \{A\}$ und $\mathcal{S}_2.\mathcal{X}^a_{p_r} = \{A'\}$)

    setze $\xi := \xi \cdot (A \to A')$

    setze $\sigma_1 := \sigma_1 \cdot (A \to |\sigma_1| + 1)$ und $\sigma_2 := \sigma_2 \cdot (A' \to |\sigma_2| + 1)$

    Für alle bisherigen einelementigen Klassen $\{B\}$ mit $\{B'\} = \mathcal{F}(\{B\})$:

      Falls $\mathcal{S}_1.rel(\sigma_1, B) \ne_R \mathcal{S}_2.rel(\sigma_2, B')$

        brich den Algorithmus ab.

    Falls $S_1.\Gamma(\sigma_1) \ne_G S_2.\Gamma(\sigma_2)$

      brich den Algorithmus ab.

Wurde der Algorithmus abgebrochen, kann ξ nicht zu einem Teilstrukturisomorphismus von $\mathcal{S}_1$ auf $\mathcal{S}_2$ fortgesetzt werden. Wurde die Verfeinerung in einem Backtrackaufruf bearbeitet, gehe zurück zum

Backtracking. Ansonsten brich den gesamten Algorithmus ab.

Ist der Algorithmus jedoch erfolgreich beendet worden, so gehe zum nächsten Schritt:

Schritt 3: Konstruktion

Die zuvor erreichten Partitionierungen von $S_1.\mathcal{K}^a_p$ und $S_2.\mathcal{K}^a_q$ führen zu den folgenden neuen Klassifizierung. (o.B.d.A. $p < |S_1.\mathcal{K}^a|$)

Es gilt also

$$S_1.\mathcal{K}^{a+1} = \left(S_1.\mathcal{K}^a_1, \ldots, S_1.\mathcal{K}^a_{p-1}, S_1.\mathcal{K}^a_{p_1}, \ldots, S_1.\mathcal{K}^a_{p_{k_p}}, S_1.\mathcal{K}^a_{p+1}, \ldots, S_1.\mathcal{K}^a_{|S_1.\mathcal{K}^a|}\right)$$

und damit

$$|S_1.\mathcal{K}^{a+1}| = |S_1.\mathcal{K}^a| + k_p - 1$$

Analog gilt für S_2:

$$S_2.\mathcal{K}^{a+1} = \left(S_2.\mathcal{K}^a_1, \ldots, S_2.\mathcal{K}^a_{q-1}, S_2.\mathcal{K}^a_{q_1}, \ldots, S_2.\mathcal{K}^a_{q_{k_q}}, S_2.\mathcal{K}^a_{q+1}, \ldots, S_2.\mathcal{K}^a_{|S_1.\mathcal{K}^a|}\right)$$

sowie

$$|S_2.\mathcal{K}^{a+1}| = |S_2.\mathcal{K}^a| + k_q - 1$$

Zusätzlich muß die Anordnung $\mathcal{F}$ aktualisiert werden. Setze dazu

$$\mathcal{F}(S_1.\mathcal{K}^a_{p_i}) := S_2.\mathcal{K}^a_{q_k} \quad \forall\, S_2.\mathcal{K}^a_{q_k},\ 1 \le k \le k_q,$$

wobei $S_2.\mathcal{K}^a_q$ diejenige Klasse aus $S_2.\mathcal{K}^a$ ist, die für $S_1.\mathcal{K}^a_{p_i}$ die Forderungen aus Schritt 2 erfüllt.

Die neue Klassifizierung wird danach aktuell. Querverweise auf Klassen der vorherigen Klassifizierung müssen aktualisiert werden.

Ist die neue Klassifizierung erzeugt, und sind noch Klassen in der aktuellen Klassifizierung, die aus mehr als einem Element bestehen und noch nicht mit Hilfe von μ_1 und μ_2 zur Verfeinerung herangezogen worden sind, so gehe zurück zu Schritt 1.

Erhöhe sonst V_{max} und rufe das Modul Test auf Vollständigkeit auf.

4.3.2.5. Backtracking

Das Backtracking - Modul wird aus verschiedenen Situationen heraus aufgerufen:

Situation 1: Erstmaliger Aufruf

In diesem Fall geschieht der Aufruf immer aus dem Modul 4.3.2.3. (Test auf Verfeinerbarkeit) aufgerufen. Es stehen dann keine einelementigen Klassen in der aktuellen Klassifizierung mehr zur Verfügung, mit denen eine Verfeinerung durchgeführt werden könnte. Mit Hilfe des Backtracking wird nun stattdessen, quasi „künstlich", eine einelementige Klasse geschaffen, so daß der Algorithmus fortgesetzt werden kann. Das Vorgehen ist in diesem Fall absolut analog zu dem Vorgehen bei der Vollstrukturisomorphie. Zuerst bestimmt man die Klasse $S_1.\mathcal{K}^a_i$ mit der geringsten Anzahl von Elementen. Diese wird wiederum bezüglich des Elements mit dem niedrigsten Index in zwei neue Klassen $S_1.\mathcal{K}^a_{i_1}$ und $S_1.\mathcal{K}^a_{i_2}$ unterteilt. Analog dazu die Klasse $\mathcal{F}(S_1.\mathcal{K}^a_i) = S_2.\mathcal{K}^a_j$ in $S_2.\mathcal{K}^a_{j_1}$ und $S_2.\mathcal{K}^a_{j_2}$. Die neue Festlegung $\mathcal{F}(S_1.\mathcal{K}^a_{i_1}) := S_2.\mathcal{K}^a_{j_1}$ muß noch auf ihre Verträglichkeit mit den bisher festgelegt einelementigen Klassen (d.h. bezüglich **PISO3** und **PISO4**) getestet werden. Falls zusätzlich $|S_2.\mathcal{K}^a_{j_2}| = |S_1.\mathcal{K}^a_{i_2}| = 1$ gilt, muß auch die Festlegung $\mathcal{F}(S_1.\mathcal{K}^a_{i_2}) := S_2.\mathcal{K}^a_{j_2}$ diesbezüglich geprüft werden. Sind die Festlegungen nicht verträglich, muß analog zum Isomorphietest die nächste mögliche Kombination gewählt werden.

Sind beide neuen Klassifizierungen erstellt worden, so wird wiederum das Modul 4.3.2.2. aufgerufen.

Situation 2: Aufruf aus Verfeinerung

Dieser Fall tritt auf, wenn in Modul 4.3.2.4. die Isomorphiebedingungen beim Erstellen einer Verfeinerung nicht erfüllt sind. In diesem Fall führt die aktuelle Klassifizierung von S_1 und S_2 nicht mehr zu einer Einbettung von S_1 in S_2 führen. Sind noch Elemente aus $S_2.\mathcal{K}_j^a$ nicht als Bild von $S_1.\mathcal{K}_{i_1}^a$ konstruiert worden, erstelle wie in Situation 1 die entsprechenden Klassifizierungen und teste ihre Verträglichkeit mit **PISO3** und **PISO4**. Rufe damit wieder den Test auf Vollständigkeit auf. Sind alle Kombinationen ausgeschöpft, so beende das Backtracking.

Situation 3: Aufruf nach vollständiger Klassifizierung

Diese Situation ist ähnlich zur vorherigen. Nur war in diesem Fall eine vollständige Klassifizierung möglich, d.h. es wurde eine Einbettung von S_1 in S_2 gefunden. Gib diese mit Hilfe von 4.3.2.6. aus. Auch hier muß zuvor Situation 1 erreicht worden sein. Sind dort noch neue Aufteilungen von $S_2.\mathcal{K}_j^a$ möglich, so führe diese aus. Beende sonst den Algorithmus.

4.3.2.6. Vollständige Klassifizierung

Dieses Modul wird aufgerufen, wenn die Bedingungen in 4.3.2.2. für eine vollständige Klassifizierung für S_1 und entsprechende einelementige Klassen als Bilder unter $\mathcal{F}$ ermittelt worden sind. Das bedeutet, daß eine Einbettung von S_1 in S_2 gefunden wurde. Man beachte, daß die aktuelle Klassifizierung der Teststruktur S_2 nicht vollständig sein muß, da es sich hier um Teilstrukturisomorphie handelt und deshalb nicht alle Atome der Struktur S_2 für eine Einbettung von S_1 notwendig sind.

Das bedeutet, daß ein Teilstrukturisomorphismus $\varphi : S_1.\mathrm{S} \rightarrow S_2.\mathrm{S}$ gefunden wurde. Die gesuchte Einbettung φ wird dabei definiert durch die während des Algorithmus in ξ gespeicherten Permutationen.

4.3.2.7. Satz

Die durch den Algorithmus ermittelten Teilstrukturisomorphismen φ erfüllen die Bedingungen **PISO1** - **PISO4** und sind vollständig.

Beweis

Durch die Vorklassifizierung und Satz 4.3.2.1.1. sind **PISO1** und **PISO2** sichergestellt. Die weiteren Verfeinerungen der Klassifizierung werden so vorgenommen, daß immer die Bedingungen **PISO3** und **PISO4** erfüllt sind, sowie **PISO2** erhalten bleibt. Durch das Backtracking werden alle Möglichkeiten ausgeschöpft.

Dieser Algorithmus zur Teilstruktursuche eignet sich natürlich genauso zur Ermittlung der Vollstrukturisomorphie. Da hier jedoch weniger „scharfe" Bedingungen verwendet werden, ist der spezialisierte Algorithmus aus 4.1. immer effektiver und damit auch schneller.

4.3.3. Beispiel

Für die Darstellung der Teilstrukturisomorphie verwenden wir ein Beispiel aus der Chemie. Die

betrachteten Strukturen sind in diesem Fall:

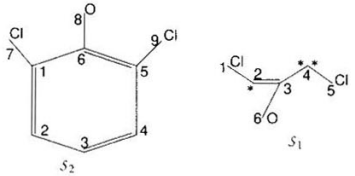


Abbildung 16: *Referenz- und Teststruktur bei Teilstrukturisomorphie*

Es werden alle möglichen Einbettungen von S_1 in S_2 gesucht. Dadurch, daß in diesem Beispiel nicht mehr wie in den vorherigen Kapiteln der gefärbte Würfel betrachtet wird, sondern Moleküle, ist es notwendig eine neue Schnittstelle zu dieser Art diskreter Struktur zu definieren.

Die nicht durch Bezeichnungen gekennzeichneten Atome im obigen Bild sind Kohlenstoffe, die vier Bindungen erlauben. Die Sterne in der Darstellung von S_1 weisen also auf die Anzahl der nicht verwendeten Bindungen hin.

Die folgenden Schnittstellen werden im nächsten Kapitel noch ausführlich erklärt, so daß diesbezüglich an dieser Stelle keine weiteren Erläuterungen gegeben werden.

Für die Einbettung von S_1 in S_2 benötigt man eine Abbildung, welche die Atome der beiden Moleküle aufeinander abbildet und dabei die Atomtypen (C, O, Cl) berücksichtigt und ihre Konnektivität. Zu diesem Zweck definieren wir eine von der abstrakten Klasse TRÄGER abgeleitete Klasse ATOM zur Darstellung der Atome. Bestandteil dieser Klasse ist das Attribut ATOMTYP, das den Namen eines Atoms aus dem Periodensystem aufnimmt. Die Vergleichsfunktionen „==" und „<" sind durch den lexikographischen Vergleich vorgegeben.

Die Trägerinstanzen können also bei S_2 eindeutig mit den Objektidentitäten 1-9, und bei S_1 mit 1-6 numeriert werden. Beide Strukturen basieren damit auf dem gleichen Modell und es sind keine Toleranzen in dieser Darstellung beim Vergleich zweier Atome erlaubt.

Die Klasse REL - ERGEBNIS besitzt - analog zum gefärbten Würfel - das Attribut Konnektivität vom Typ $\mathbb{N}$ und die damit darauf definierten natürlichen Vergleichsoperatoren. Die Methode *rel* ist derart implementiert, daß ein Aufruf von *rel* eine Instanz der Klasse REL - ERGEBNIS mit dem Attribut Konnektivität erzeugt und das Attribut Konnektivität anhand der Bindungsvielfachheit belegt wird, welche die als zweites Argument übergebene Instanz des Typs ATOM zu der zuletzt in der ebenfalls übergebenen aktuellen Numerierung σ festgelegten, weiteren Instanz vom Typ ATOM besitzt.

Die Methode *rel* ist also in diesem Fall äquivalent zu einer Funktion $rel : \mathcal{E} \times S \rightarrow \mathbb{N}$, definiert durch

$$rel(\sigma, B) := \mathrm{Conn}(A, B),$$

wobei A das zuletzt festgelegte Element der Trägermenge in der aktuellen Numerierung σ ist. Conn ist die Konnektivität oder Bindungsvielfachheit zwischen zwei Atomen. Es gilt also beispielsweise Conn $(1, 2) = 2$ in S_{test}.

Mit diesen Schnittstellen läßt sich nun bereits der Algorithmus durchführen.

Schritt 1: Vorklassifizierung

Es gilt $S_1.\mathcal{K}^0 = (\{2, 3, 4\}, \{6\}, \{1, 5\})$

und $S_2.\mathcal{K}^0 = (\{1, 2, 3, 4, 5, 6\}, \{8\}, \{7, 9\})$.

Es ist nun zu testen, ob alle Bedingungen aus Satz 4.3.2.1.1. erfüllt sind.

1. Es gilt $|\mathcal{S}_1.\mathcal{K}^0| \leq |\mathcal{S}_2.\mathcal{K}^0|$.

2. Es gilt $A =_S B$ für $A \in \mathcal{S}_1.\mathcal{K}_1^0, B \in \mathcal{S}_2.\mathcal{K}_1^0$ und

   $|\mathcal{S}_1.\mathcal{K}_1^0| \leq |\mathcal{S}_2.\mathcal{K}_1^0|$. Weiterhin gilt

   $A \neq_S B$ für $A \in \mathcal{S}_1.\mathcal{K}_1^0, B \in \mathcal{S}_2.\mathcal{K}_2^0, \mathcal{S}_2.\mathcal{K}_3^0$ aufgrund der Ordnung von $\mathbf{N}$ und des Aufbaus der Ausgangsklassifizierung.

   Es gilt $A =_S B$ für $A \in \mathcal{S}_1.\mathcal{K}_2^0, B \in \mathcal{S}_2.\mathcal{K}_2^0$ und

   $|\mathcal{S}_1.\mathcal{K}_2^0| \leq |\mathcal{S}_2.\mathcal{K}_2^0|$, sowie

   $A \neq_S B$ für $A \in \mathcal{S}_1.\mathcal{K}_2^0, B \in \mathcal{S}_2.\mathcal{K}_1^0, \mathcal{S}_2.\mathcal{K}_3^0$

   Es gilt $A =_S B$ für $A \in \mathcal{S}_1.\mathcal{K}_3^0, B \in \mathcal{S}_2.\mathcal{K}_3^0$ und

   $|\mathcal{S}_1.\mathcal{K}_3^0| \leq |\mathcal{S}_2.\mathcal{K}_3^0|$, sowie

   $A \neq_S B$ für $A \in \mathcal{S}_1.\mathcal{K}_3^0, B \in \mathcal{S}_2.\mathcal{K}_1^0, \mathcal{S}_2.\mathcal{K}_2^0$

Damit sind alle Forderungen des Satzes erfüllt und für die Zuordnungsfunktion $\mathcal{F}$ gilt:

K	{2, 3, 4}	{6}	{1, 5}
$\mathcal{F}(K)$	{1, 2, 3, 4, 5, 6}	{8}	{7, 9}

Da noch keine weiteren einelementigen Klassen vorhanden sind, gilt: $\xi := \xi \cdot (6 \rightarrow 8)$ und $\sigma_1 = \sigma_1 \cdot (6 \rightarrow 1)$ sowie $\sigma_2 = \sigma_2 \cdot (8 \rightarrow 1)$.

Es gilt $V_{max} = 0$.

Schritt 2: Test auf Vollständigkeit

Da $|\mathcal{F}(\mathcal{S}_1.\mathcal{K}_1^0)| > 1$, liegt keine vollständige Klassifizierung vor. Es wird deshalb zum Test auf Verfeinerbarkeit verzweigt.

Schritt 3: Test auf Verfeinerbarkeit

Da $V_{max} < |\xi| = 1$, kann eine Verfeinerung durchgeführt werden. Erhöhe V_{max} um 1.

Schritt 4: Verfeinerung

Partitionierung:

Durch die Verfeinerung bezüglich des zuletzt festgelegten Atoms Nummer 6 in der aktuellen Numerierung $\mu_1 = \sigma_1$, wird $\mathcal{S}_1.\mathcal{K}_1^0$ in zwei neue Klassen aufgespalten:

K	{3}	{2, 4}
$rel(\mu_1, x)$ mit $x \in K$	1	0

Analog dazu wird $\mathcal{S}_2.\mathcal{K}_1^0$ bezüglich $\mu_2 = \sigma_2$ und dem dort zuletzt festgelegten Atom Nummer 8 aufgespalten in:

K	{6}	{1, 2, 3, 4, 5}

$rel(\mu_2, x)$ mit $x \in K$	1	0

Identifizierung:

Es können nur Klassen mit gleichen Attributen des REL - ERGEBNISses aufeinander abgebildet werden. Dies bedeutet: $\mathcal{F}(\{3\}) = \{6\}$ und $\mathcal{F}(\{2,4\}) = \{1,2,3,4,5\}$. Da neue einelementige Klassen aus der Verfeinerung hervorgegangen sind, muß ihre Verträglichkeit bzgl. **PISO3** und **PISO4** noch überprüft werden.

Es gilt trivialerweise $rel(\mu_1, 3) =_R rel(\mu_2, 6)$, woraus ja diese Verfeinerung entstanden ist.

Damit sind alle Bedingungen für eine mögliche Einbettung der Referenz- in die Teststruktur erfüllt. Die Konstruktion der neuen Klassifizierung kann erfolgen.

Konstruktion:

Die Konstruktion der neuen Klassifizierungen liefert folgende Ergebnisse:

$$S_1.\mathcal{K}^1 = (\{3\}, \{2,4\}, \{6\}, \{1,5\})$$

$$S_2.\mathcal{K}^1 = (\{6\}, \{1,2,3,4,5\}, \{8\}, \{7,9\})$$

Die Zuordnungsfunktion $\mathcal{F}$ hat folgenden Wertebereich:

K	$\{3\}$	$\{2,4\}$	$\{6\}$	$\{1,5\}$
$\mathcal{F}$(K)	$\{6\}$	$\{1,2,3,4,5\}$	$\{8\}$	$\{7,9\}$

Die Klasse $S_1.\mathcal{K}^0_3$ (jetzt: $S_1.\mathcal{K}^1_4$) wird durch μ_1 nicht verfeinert, da beide Chlor - Atome nicht direkt benachbart zu dem Sauerstoffatom 6 sind und damit identische Werte bezüglich der Methode *rel* (=0) ergeben (vgl. Abbildung 16). Damit gilt: $\xi := \xi \cdot (3 \rightarrow 6)$ und $\sigma_1 = \sigma_1 \cdot (3 \rightarrow 2)$ sowie $\sigma_2 = \sigma_2 \cdot (6 \rightarrow 2)$.

Im nächsten Schritt wird dann der Test auf Vollständigkeit aufgerufen.

Schritt 5: Test auf Vollständigkeit

Da $|\mathcal{F}(S_1.\mathcal{K}^1_2)| > 1$, liegt keine vollständige Klassifizierung vor. Es wird deshalb zum Test auf Verfeinerbarkeit verzweigt.

Schritt 6: Test auf Verfeinerbarkeit

Da $1 = V_{max} < |\xi| = 2$, kann eine Verfeinerung durchgeführt werden. Erhöhe V_{max} um 1.

Schritt 7: Verfeinerung

Partitionierung:

Durch die Verfeinerung bezüglich $\mu_1 = \begin{pmatrix} 6 & 3 \\ 1 & 2 & * \end{pmatrix}$, d.h. bzgl. der Konnektivität zum zuletzt festgelegten Atom Nummer 3 wird $S_1.\mathcal{K}^1_2$ in zwei neue Klassen aufgespalten:

K	$\{2\}$	$\{4\}$
$rel(\mu_1, x)$ mit $x \in K$	2	1

Analog dazu wird $S_2.\mathcal{K}^1_2$ bezüglich $\mu_2 = \begin{pmatrix} 8 & 6 \\ 1 & 2 & * \end{pmatrix}$ bzw. Atom Nummer 6 aufgespalten in:

K	$\{5\}$	$\{1\}$	$\{2,3,4\}$

$rel(\mu_2, x)$ mit $x \in K$	2	1	0

Identifizierung:

Folgende Klassen der Teil- und Teststruktur können damit identifiziert werden: $\{2\} \rightarrow \{5\}$ und $\{4\} \rightarrow \{1\}$. Alle dafür notwendigen Bedingungen aus 4.3.2.4./Schritt 2 sind erfüllt. Somit gilt: $\xi := \xi \cdot (2 \rightarrow 5)$ und $\sigma_1 = \sigma_1 \cdot (2 \rightarrow 3)$ sowie $\sigma_2 = \sigma_2 \cdot (5 \rightarrow 3)$ und zusätzlich $\xi := \xi \cdot (4 \rightarrow 1)$ und $\sigma_1 = \sigma_1 \cdot (4 \rightarrow 4)$ und $\sigma_2 = \sigma_2 \cdot (1 \rightarrow 4)$

Konstruktion:

Die Konstruktion der neuen Klassifizierungen ergibt:

$$S_1.\mathcal{K}^2 = (\{3\}, \{2\}, \{4\}, \{6\}, \{1, 5\})$$

$$S_2.\mathcal{K}^2 = (\{6\}, \{5\}, \{1\}, \{2, 3, 4\}, \{8\}, \{7, 9\})$$

Die Zuordnunsfunktion $\mathcal{F}$ hat folgenden Wertebereich:

K	$\{3\}$	$\{2\}$	$\{4\}$	$\{6\}$	$\{1, 5\}$
$\mathcal{F}(K)$	$\{6\}$	$\{5\}$	$\{1\}$	$\{8\}$	$\{7, 9\}$

Die Klasse $S_1.\mathcal{K}_4^1$ (jetzt: $S_1.\mathcal{K}_5^2$) wird wiederum durch μ_1 und damit die Instanz der Klasse ATOM, identifiziert mit der Nummer 3, nicht verfeinert (da beide Cl - Atome nicht mit Nummer 3 verbunden sind).

Man beachte, daß keine Klasse der Referenzstruktur auf $S_2.\mathcal{K}_5^2$ abgebildet wird. Das bedeutet, diese Klasse spielt im weiteren Verlauf des Algorithmus keine Rolle mehr. Die Atome, die in dieser Klasse enthalten sind können nicht Bild eines Atoms der betrachteten Teilstruktur sein.

Es folgt ein neuer Test auf Vollständigkeit:

Schritt 8: Test auf Vollständigkeit

Da $|\mathcal{F}(S_1.\mathcal{K}_5^2)| > 1$, liegt keine vollständige Klassifizierung vor. Es wird deshalb zum Test auf Verfeinerbarkeit verzweigt.

Schritt 9: Test auf Verfeinerbarkeit

Da $2 = V_{max} < |\xi| = 3$, kann eine Verfeinerung durchgeführt werden. Erhöhe V_{max} um 1.

Schritt 10: Verfeinerung

Partitionierung:

Durch die Verfeinerung bezüglich $\mu_1 = \begin{pmatrix} 6 & 3 & 2 & \bullet \\ 1 & 2 & 3 & \end{pmatrix}$ bzw. durch ATOM Nummer 2 wird $S_1.\mathcal{K}_5^2$ in zwei neue Klassen aufgespalten:

K	$\{1\}$	$\{5\}$
$rel(\mu_1, x)$ mit $x \in K$	1	0

Analog dazu wird $\mathcal{F}(S_1.\mathcal{K}_5^2) = S_2.\mathcal{K}_6^2$ bezüglich $\mu_2 = \begin{pmatrix} 8 & 6 & 5 \\ 1 & 2 & 3 \end{pmatrix}$ bzw. Atom Nummer 5 aufgespalten in:

K	$\{9\}$	$\{7\}$
$rel(\mu_2, x)$ mit $x \in K$	1	0

Identifizierung:

Die Identifizierung der entstandenen Referenz- und Teststrukturklassen ergibt: $\{1\} \rightarrow \{9\}$ und $\{5\} \rightarrow \{7\}$. Alle notwendigen Bedingungen sind dabei erfüllt. Es gilt: $\xi := \xi \cdot (1 \rightarrow 9)$ und $\sigma_1 = \sigma_1 \cdot (1 \rightarrow 5)$ sowie $\sigma_2 = \sigma_2 \cdot (9 \rightarrow 5)$ und zusätzlich $\xi := \xi \cdot (5 \rightarrow 7)$ und $\sigma_1 = \sigma_1 \cdot (5 \rightarrow 6)$ und $\sigma_2 = \sigma_2 \cdot (7 \rightarrow 6)$

Konstruktion:

Die Konstruktion der neuen Klassifizierungen ergibt:

$$S_1.\mathcal{K}^3 = (\{3\}, \{2\}, \{4\}, \{6\}, \{1\}, \{5\})$$

$$S_1.\mathcal{K}^3 = (\{6\}, \{5\}, \{1\}, \{2, 3, 4\}, \{8\}, \{9\}, \{7\})$$

Die Zuordnunsfunktion $\mathcal{F}$ hat dann den Wertebereich:

K	{3}	{2}	{4}	{6}	{1}	{5}
$\mathcal{F}$(K)	{6}	{5}	{1}	{8}	{9}	{7}

Da neben $S_1.\mathcal{K}_S^2$ alle Bildklassen der Klassifizierung $S_1.\mathcal{K}^2$ unter der Zuordnungsfunktion $\mathcal{F}$ einelementig waren, kann keine weitere Klasse mit Hilfe der ATOMe 2 bzw. 5 verfeinert werden.

Schritt 11: Test auf Vollständigkeit

Die Bedingung für Vollständigkeit ist für die Klassifizierungen $S_1.\mathcal{K}^3$ und $S_2.\mathcal{K}^3$ erfüllt. Es wird zum Schritt Vollständige Klassifizierung verzweigt.

Schritt 9: Vollständige Klassifizierung

Es wurde eine Einbettung von S_1 in S_2 gefunden. Diese entspricht (natürlich) der bereits erwarteten:

A	1	2	3	4	5	6
ξ(A)	9	5	6	1	7	8

Eine weitergehende, effektivere Diskretisierung für die zweidimensionale Substruktursuche bei Molekülen wird in einem späteren Abschnitt gezeigt.

4.4. Berücksichtigung von Isomorphieeigenschaften

Sowohl der Suchprozeß bei der Vollstrukturisomorphie als auch bei der Teilstrukturisomorphie läßt sich vereinfachen, wenn bereits Informationen über die interne Isomorphie der betrachteten Strukturen vorliegen.

Es ist zu beachten, daß dieses Kapitel ausschließlich bei der Betrachtung von Isomorphie von Bedeutung ist. Entsprechende Aussagen können bei der Ähnlichkeitsbetrachtung aus den gegebenen Gründen nicht vorgenommen werden.

4.4.1. Beispiel

Betrachten wir wiederum ein Beispiel aus der Chemie. Als Teilstruktur ist der folgende Sechsring S_1

gegeben.

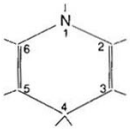


Abbildung 17: *Teilstruktur S_1*

Wie man leicht sieht, aber auch mit dem zuvor vorgestellten Algorithmus berechnen kann, gilt

$$\mathrm{Aut}(S_1) = \{\mathrm{id}, / 1\,6\,5\,4\,3\,2 /\}$$

Die Struktur S_2, in der nach S_1 gesucht werden soll ist:

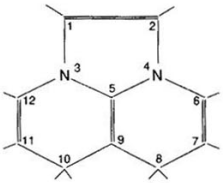


Abbildung 18: *Teststruktur S_2*

Es soll an dieser Stelle kein Anspruch auf die tatsächliche Existenz des Moleküls S_2 erhoben werden. Wichtig ist lediglich wiederum die Automorphismengruppe:

$$\mathrm{Aut}(S_2) = \{\mathrm{id}, / 2\,1\,4\,3\,5\,12\,11\,10\,9\,8\,7\,6 /\}$$

Im Verlauf des zuvor beschriebenen Algorithmus wird nun beispielsweise zuerst die Einbettung φ^1 von S_1.S in S_2.S mit

$$\varphi^1 = / 3\,5\,9\,10\,11\,12 /$$

gefunden. Aus der Isomorphieeigenschaft von S_1 folgt sofort, daß auch eine Einbettung φ^2 mit

$$\varphi^2 = (\mathrm{Aut}(S_1)\backslash\{\mathrm{id}\})\circ\varphi^1 = / 3\,12\,11\,10\,9\,5 /$$

existieren muß.

Unter Berücksichtigung von $\mathrm{Aut}(S_2)$ ergeben sich daraus wiederum sofort die beiden ebenfalls möglichen Einbettungen

$$\varphi^3 = / 4\,5\,9\,8\,7\,6 /$$

und

$$\varphi^4 = / 4\,6\,7\,8\,9\,5 /$$

Wie man sieht, werden durch das Auffinden bereits einer Einbettung mit Hilfe der verfügbaren Isomorphieinformationen bereits alle möglichen Einbettungen von S_1 in S_2 gefunden.

4.4.2. Satz

Seien S' und S diskrete Strukturen und φ ein Isomorphismus von S' auf S. Seien $\alpha \in \mathrm{Aut}(S')$ und

$\beta \in \mathrm{Aut}(S)$ Automorphismen der beiden gegebenen Strukturen. Dann gilt:

$$\beta \circ \varphi \circ \alpha$$

ist ebenfalls eine Isomorphismus von S' auf S.

Beweis

Folgt direkt aus der Definition der Isomorphie von diskreten Strukturen aus 3.3.2.. Die Struktureigenschaften bleiben unter Anwendung von Automorphismen erhalten.

Voraussetzung zur Ausnutzung dieses Satzes ist natürlich die Berechnung der Automorphismengruppe der betrachteten Strukturen, etwa nach dem in 4.1. vorgestellten Verfahren.

Verwendet man die Substruktursuche in einer Datenbank, wo in der Regel sehr viele Strukturen durchsucht werden müssen, so kann es durchaus sinnvoll sein, Informationen über die Automorphismengruppe mit abzuspeichern.

Benötigt man keine Informationen über die gesamte Anzahl der Einbettungen, sondern ist nur daran interessiert festzustellen, ob bestimmte Substrukturen in einer Struktur vorhanden sind, ist dies jedoch nicht nötig. Der Algorithmus kann dann bereits nach der ersten gefundenen Einbettung mit der gewünschten Information abgebrochen werden.

Isomorphieeigenschaften sind also zusätzliche, wesentliche Eigenschaften einer diskrete Struktur, die damit auch zu einer Beschleunigung der Klassifizierung eingesetzt werden können.

Die Algorithmen werden leicht verändert, um die Isomorphieeigenschaften ausnutzen zu können.

Wird - wie im obigen Beispiel gezeigt - eine Einbettung einer Referenzstruktur in die Teststruktur gefunden, so liegt eine vollständige Klassifizierung der Trägerinstanzen der Referenzstruktur vor. Nachdem die letzte Zuordnung einer einelementigen Klasse von S_1 in S_2 vorgenommen wurde, wird der folgende Algorithmus gestartet, der eine direkte Anwendung des vorherigen Satzes ist:

Gegeben sei die aktuelle Einbettung ξ, die $S_1.S$ nach $S_2.S$ abbildet.

Für alle $\alpha \in \mathrm{Aut}(S_2)$

  Für alle $\beta \in \mathrm{Aut}(S_1)$

    Gib $\alpha(\xi(\beta))$ *aus*

    Markiere $\alpha(\xi(\beta))$ *als bereits verwendete Einbettung*

Damit werden alle möglichen isomorphen Einbettungen ausgegeben. Das Markieren der bereits verwendeten Anordnungen ist dabei von besonderer Bedeutung.

Da die vorgestellten Algorithmen zur Isomorphiebestimmung und Teilstrukturisomorphiebestimmung immer *alle* möglichen Einbettungen berechnen, muß nun der Isomorphiealgorithmus Informationen über die automatisch durch den obigen Algorithmus bereits erreichten zusätzlichen Einbettungen erhalten und ausnutzen.

Besitzen dabei S_1 und S_2 eine triviale Automorphismengruppe, so wird natürlich kein Backtracking verwendet und der Algorithmus findet immer direkt die einzige Einbettung, falls überhaupt eine existiert. Im nichttrivialen Fall möchte man jedoch die Information über die bekannten Einbettungen möglichst frühzeitig nutzen, um Wege im Backtrackbaum zu sparen.

Um dies zu erreichen, muß die Möglichkeit bestehen, bei jeder Einbettung zu entscheiden, ob eine Fortsetzung ausschließlich zu Einbettungen führt, die durch vorherige Einbettungen erzeugt werden können.

Die Isomorphismen der Referenz- und Teststruktur müssen also in entsprechender Form dargestellt werden, um solche Tests einfach durchführen zu können.

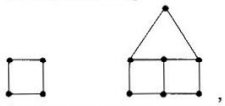
Da, wie bereits gezeigt haben, Permutationsgruppen in der Regel bereits für eine geringe Anzahl von Elementen äußerst umfangreich sind, sollte eine sehr kompakte Darstellung gewählt werden, die obigen Anforderungen genügt.

Die Isomorphieeigenschaften sind insbesondere auch bei der Vollstruktursuche effizient einsetzbar. Ist $\mathrm{Aut}(S_1)$ bekannt, so kann der Algorithmus bereits nach der ersten gefunden Einbettung abgebrochen werden, da bei der Vollstrukturisomorphie keine weiteren Einbettungen außer der gefundenen und den durch $\mathrm{Aut}(S_2)$ gemäß dem vorherigen Algorithmus berechneten existieren können.

Bei der kanonischen Numerierung tritt ein Spezialfall auf. Ist die Automorphismengruppe der betrachteten Struktur unbekannt, dann „überschneiden" sich quasi zwei Abbruchkriterien für den Backtrackbaum. Zum einen wird durch die fortsetzbare Methode $\mathcal{K}l$ frühzeitig ein Teilbaum abgetrennt, der nicht zu einer kanonischen Numerierung führen kann. Andererseits findet man dann evtl. keine neuen Numerierung zur Erweiterung der Automorphismengruppe. In diesem Fall liefert die Ausnutzung der Automorphismengruppe keine Abbruchkriterien. Es kann also bei hochsymmetrischen Strukturen durchaus sinnvoll sein, erst die Automorphismengruppe zu bestimmen, und dann daraus ein kanonisches Element gemäß Algorithmus 4.2. zu berechnen. Bei Strukturen mit einem geringen Grad von Isomorphie, wie beispielsweise Molekülen, ist dieses Vorgehen jedoch nicht effektiver.

Eine entsprechende Schnittstelle ist in den Algorithmus integriert worden.

Bei verschiedenen Problemen tritt jedoch auch ein dem oben beschriebenen Ansatz genau entgegengesetztes Problem auf. Betrachtet man die folgenden Graphen



und möchte alle Einbettungen des linken Graphen in den rechten Graphen berechnen, so werden durch den vorliegenden Algorithmus sehr viel mehr Einbettungen gefunden, als anfänglich erwünscht. Dies liegt daran, daß der Algorithmus prinzipiell von einer Ausgangsnumerierung der einzelnen Trägerinstanzen, in diesem Fall Knoten, ausgeht, und die einzelnen Knoten dadurch unterscheidbar werden. Dies führt in den meisten Fällen zu einer korrekten Lösung des Problems, in einigen ist dies jedoch nicht erwünscht.

Es ist jedoch möglich, diejenigen Einbettungen zu eliminieren, die für das Problem nicht von Bedeutung sind. Dies geschieht dadurch, daß man beim obigen Algorithmus eben nicht alle durch die Automorphismengruppe induzierten Einbettungen zusätzlich ausgibt, sondern diese Ausgabe unterdrückt. Dies würde in unserem Beispiel bedeuten, daß der linke Graph nur einmal in den rechten Graphen eingebettet werden kann, da sich alle übrigen Einbettungen durch die Symmetrien ergeben. Es ist jedoch auch möglich, nur diejenigen Einbettungen zu isolieren, die sich aus der Automorphismengruppe der Teilstruktur ergeben, jedoch alle Einbettungen, die sich durch die Automorphismengruppe der Teststruktur ergeben zu berücksichtigen. In diesem Fall würden sich 2 Einbettungen der Referenzstruktur ergeben.

Die Beurteilung der einzelnen Einbettungen sind also stark von dem betrachteten Problem abhängig.

Der Algorithmus ist jedoch so konzipiert, daß man bei jeder neu gefundenen Einbettung entscheiden kann, ob diese ausgegeben werden soll, oder eben nicht.

4.5. Komplexität

Eine allgemeine Aussage über die Komplexität der vorgestellten Algorithmen kann zunächst nicht getroffen werden. Wie bereits eingangs erwähnt, ist die Komplexität in hohem Maße von der diskreten Struktur bzw. ihrer Umsetzung bezüglich der vorgegebenen Schnittstellen abhängig. Für eine genauere

Komplexitätsbetrachtung beschränken wir uns zunächst auf den ersten Algorithmus zur Ähnlichkeits- bzw. Isomorphieerkennung.

Betrachten wir zunächst den (vereinfachten) Algorithmus zur Isomorphieerkennung, bestehend aus der Kombination von iterierter Klassifizierung und Backtracking:

```
Backtrack ( ISO, S, T )
{
    Ist S = T = ∅,
      dann gib ISO aus.
    Sonst
      Für ein i ∈ S
        Für alle j ∈ T
          Bilde ISO1 := ISO·(i → j).
          Ist ISO1 zulässig,
            dann bilde Backtrack (ISO1, S \{i},T \{j})
}
```

Wird durch die iterierte Klassifizierung keine Einschränkung der zulässigen Isomorphismen erreicht, d. h. sind alle Kombinationen der Trägerinstanzen für einen Isomorphismus denkbar, und wird innerhalb der iterierten Klassifizierung keine Verfeinerung der Trägerinstanzen vorgenommen, so müssen alle n! Möglichkeiten (bei einer Mächtigkeit der Trägermenge von n) überprüft werden. Der Aufwand ist in diesem Falle exponentiell und stellt gleichzeitig den „Worst Case" dar.

Eine Verringerung dieses Aufwandes ist also nur durch eine frühzeitige Erkennung der Zulässigkeit von Kombinationen bzw. durch eine Verfeinerung der Klassifizierung der Trägerinstanzen mit Hilfe der iterierten Klassifizierung zu erreichen.

Wie man am vorherigen Beispiel leicht nachvollziehen kann, reduziert sich die Komplexität in dem Umfang, wie die Anzahl der zur Beschreibung der diskreten Struktur verwendeten Eigenschaften zunimmt. Je mehr Informationen in den Schnittstellen der objektorientierten Darstellung hinterlegt werden, desto effizienter kann der Algorithmus arbeiten.

Es bleibt nun zu zeigen, daß der Algorithmus zur iterierten Klassifizierung nicht mit exponentiellem Aufwand abläuft, und somit durch geeignete Wahl von Diskretisierung und Schnittstellen eine Verbesserung gegenüber der reinen Backtrackinglösung - und damit dem Worst Case - zu erreichen ist.

Eugene Luks hat bereits mehrere Arbeiten zu dieser Problematik veröffentlicht ([LU1]). Er konnte damit zeigen, daß mit dieser Kombination von iterierter Klassifizierung und Backtracking Isomorphietests bei Graphen mit beschränktem Knotengrad mit polynomialem Aufwand durchgeführt werden können.

5. Graphen

Da die Vorläufer der vorgestellten Algorithmen ursprünglich ausschließlich für Anwendungen in der Graphentheorie eingesetzt wurden, wird in diesem Kapitel etwas genauer auf diesen Bereich eingegangen.

Die theoretischen Grundlagen wurden bereits erwähnt und im Verlaufe der Beschreibung der Algorithmen in den vorherigen Kapiteln bereits einige Schnittstellen entsprechend definiert. Beschränken wir uns im folgenden also auf die Definition der für die Algorithmen benötigten Schnittstellen.

5.1. Schnittstelle für Graphenisomorphie

Die nachfolgenden Schnittstellendefinitionen sind ein Vorschlag für eine schnelle und einfache Anwendung der zuvor beschriebenen theoretischen Grundlagen. Viele Erweiterungen, die einen effektiveren Ablauf der Algorithmen erlauben, sind im nachfolgenden Kapitel über die Anwendung in der Chemie zu finden, da molekulare Graphen (als bewertete Graphen) gegenüber den hier vorgestellten Graphen zwar viele Unterschiede aufweisen, einige Eigenschaften jedoch übernommen werden können.

Aufgrund der Definition der Graphen beschränken wir uns in diesem Kapitel auf die Betrachtung der Isomorphie von Graphen. Die Ähnlichkeit spielt bei dieser Art von Graphen keine Rolle.

5.1.1. Träger

Der Träger der diskreten Struktur Graph ist die Menge V der Knoten (vertices). Sei also G = (V,E) die diskrete Struktur GRAPH, dann repräsentiert V die Menge der Trägerinstanzen, und die Menge der Kanten E, eine zweistellige Relation $\rho_1 \subseteq V \times V$ auf V.

5.1.1.1. Implementation

Die für eine Klassifizierung relevante Eigenschaft eines Knotens ist sein Grad. Man definiere also ein Objekt KNOTEN,

Im Objekt GRAPH existiert dann ein Feld von Instanzen des Typs KNOTEN. Die Länge des Feldes entspricht der Anzahl der Knoten im Graphen. Die Operation „<“ auf einer Instanz KNOTEN ist dann wie folgt definiert:

```
operator KNOTEN(K1) < KNOTEN(K2)
  Falls K1.Grad < K2.Grad
    Gib WAHR zurück
  Gib FALSCH zurück
```

5.1.2. Methode rel

Welche Informationen können nun zur Verfeinerung der aktuellen Klassifizierung genutzt werden, wenn eine neue einelementige Klasse entstanden ist? Wie bereits zuvor erwähnt, ist die Konnektivität in diesem Fall die einzige zusätzliche Information im Graphen. Sei a_{ij} die Adjazenzmatrix des Graphen. Sei v_1 der zuletzt in die aktuelle Numerierung aufgenommene Knoten. Dann ist die Methode *rel* äquivalent zu folgender Funktion:

$$rel(\sigma, v_2) := \begin{cases} 1 & \text{falls } \{v_1, v_2\} =: e \in E \\ 0 & \text{sonst} \end{cases}$$

5.1.2.1. Implementation

Für die Implementation der *rel* - Funktion muß zuerst ein Objekt RE (rel - Ergebnis) definiert werden. Da dort jedoch lediglich die Information über die Konnektivität aufgenommen wird, ist dies sehr einfach:

```
REL - ERGEBNIS
Daten:
   Konnektivität
Funktionen:
   Operationen: „<,=,>"
```

Auch hier sind entsprechende Operatorfunktionen vorgesehen, um das Ergebnis zu vergleichen. Beispielsweise gilt:

```
operator RE (RE1) = RE (RE2)

Falls RE1.Konnektivität = RE2.Konnektivität
   Gib WAHR zurück
Gib FALSCH zurück
```

Die Implementation der *rel* -Funktion selbst ist dann trivial.

5.1.3. Methode Γ

Die Methode Γ zum Testen globaler Eigenschaften der aktuellen Klassifizierung wird zur Bestimmung der Isomorphie von einfachen Graphen ebenso wie die Methode *kl* nicht benötigt.

Mit derart definierten Schnittstellen läßt sich der Algorithmus zur Ähnlichkeitsbestimmung anwenden. Da alle Wertebereiche exakt übereinstimmen müssen, wird in diesem Fall sogar ein Isomorphietest durchgeführt.

5.2. Schnittstelle für Teil-/Subgraphenisomorphie

Will man den Algorithmus zur Teilstrukturähnlichkeit - bzw. isomorphie einsetzen, so müssen obige Schnittstellen modifiziert werden.

Dies liegt vor allem daran, daß der Knotengrad eines Atoms nicht mehr direkt als Kriterium zur Klassifizierung eingesetzt werden kann. Betrachtet man folgendes Teilgraphenproblem

dann können Knoten des Teilgraphen auch auf Knoten des zu testenden Graphen abgebildet werden, die einen gänzlich anderen Knotengrad besitzen. Aus diesem Grund kann keine Vorklassifizierung bezüglich des Knotengrades vorgenommen werden, da Knoten der Teilstruktur unter Umständen verschiedenen Klassen zugeordnet werden, obwohl sie bei einer Einbettung in die Teststruktur auf Knoten mit identischen Knotengraden abgebildet werden. Der Algorithmus ist jedoch so konzipiert, daß Knoten aus unterschiedlichen Ausgangsklassen nicht auf Knoten aus einer Klasse mit nur einem Klassifikationsmerkmal abgebildet werden können.

Betrachten wir zunächst den Fall der *Teilgraphenisomorphie*:

5.2.1. Träger

Die Träger der Struktur Graph werden wiederum durch die Implementation der Klasse KNOTEN repräsentiert. Die Klasse KNOTEN ist also wie zuvor implementiert.

5.2.1.1. Implementation

Da - wie eben beschrieben - eine Klassifizierung der Knoten nach ihrem Grad nicht zulässig ist, jedoch keine weiteren Merkmale zur Verfügung stehen, wird für den Vergleichsoperator „<„ eine „dummy" - Funktion hinterlegt, die für alle Knoten identische Werte liefert:

```
operator KNOTEN(K1) < KNOTEN(K2)
---
Gib UNDEFINIERT zurück
```

Man beachte, daß sich dadurch die Vorklassifizierung nicht von der Ausgangsklassifizierung unterscheidet.

5.2.2. Methode rel

Die Methode *rel* ist identisch zur Graphenisomorphie definiert und auch implementiert.

5.2.3. Methode Γ

Die Methode Γ - Funktion zum Testen globaler Eigenschaften der aktuellen Anordnung kann man für eine weitere Restriktion der Zuordnung nutzen. Legt man eine neue Zuordnung von einem Knoten A des Teilgraphen auf einen Knoten A' des zu testenden Graphen fest, so muß der Knotengrad von A' mindestens so groß sein wie der Knotengrad von A.

Diese Bedingung kann man sehr einfach mit Hilfe der Methode Γ auch bei der Graphenisomorphie verwenden.

Da die Methode Γ bei jeder neuen Zuordnung aufgerufen wird, genügt es, diesen Test nur für das jeweils letzte festgelegte Atom durchzuführen. Sie ist alsoäquivalent zu der Funktion:

$$\Gamma(\sigma) = \text{Knotengrad des Knotens } \sigma^{-1}(|\sigma|)$$

5.2.3.1. Implementation

Zuerst wird die Datenstruktur für das Ergebnis der Γ - Funktion definiert:

```
GE (Γ- Ergebnis)
---
Daten:
   Knotengrad
Funktionen:
   Operationen: „="
```

Der „=" - Operators lautet dann wie folgt:

```
operator GE (GE1) = GE (GE2)
---
Falls GE2 aus dem Teilgraphen Swap (GE1,GE2)
Falls GE1.Knotengrad <= GE2.Knotengrad
   Gib WAHR zurück
Gib FALSCH zurück
```

Die Γ - Funktion ist:

```
Γ( sigma )
max = |sigma|
ermittle i mit sigma[i]=max
GE.Knotengrad = Knoten[i].Grad
Gib GE zurück
```

Bei der *Subgraphenisomorphie* ist der bestehende Vergleich der REL - ERGEBNISSE nicht zulässig.

5.2.4. Methode rel

Erinnern wir uns an die Definition der Methode *rel* und ihre Aufgabe, die Klassen der aktuellen Klassifizierung zu verfeinern. Unter Berücksichtigung der Subgraphendefinition ist es nicht möglich, eine Verfeinerung der Klassen aufgrund der aktuellen Numerierung durchzuführen. Das einzige Merkmal, das hierfür zur Verfügung steht ist die Konnektivität. Da jedoch in dem Subgraphen und im zu testenden Graphen unterschiedliche Konnektivitäten zwischen den Knoten zugelassen werden, kann diese nicht genutzt werden.

Die Methode *rel* selbst liefert also für alle Numerierungen und alle Knoten identische Werte. Eine Verfeinerung ist damit nicht möglich.

5.2.4.1. Implementation

Die Implementation ist trivial und kann leicht in den Quelltexten nachvollzogen werden.

5.2.5. Methode Γ

Die Γ - Funktion übernimmt nun die Aufgabe, eine festgelegte Numerierung auf ihre Zulässigkeit zu testen. Dies geschieht, in dem bei ihrem Aufruf eine Liste erzeugt wird, in der die Konnektivitäten aller bereits festgelegten Knoten zum zuletzt festgelegten hinterlegt werden.

Durchläuft man nun die Liste aufsteigend, so müssen die Konnektivitäten des zu testenden Graphen mindestens so groß sein, wie die Konnektivitäten innerhalb des Subgraphen.

5.2.5.1. Implementation

Das Ergebnis der Gamma - Funktion hat dann die Form:

```
GE (Γ- Ergebnis)
Daten:
   Anzahl der festgelegten Knoten
   Feld der Konnektivitäten
   Herkunft (Sub- oder Testgraph)
Funktionen:
   Operationen: „="
```

Der Vergleichsoperator „=" wird definiert durch

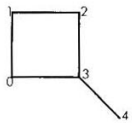
```
operator GE (GE1) = GE (GE2)
Falls GE2.Herkunft = Subgraph dann Swap(GE1,GE2)
Für alle Feldeinträge i
Falls GE1.Konnekt.[i] >GE2.Konnekt.[i]
      Gib FALSCH zurück
Gib WAHR zurück
```

Die Γ - Funktion selbst zur Ermittlung der Werte lautet:

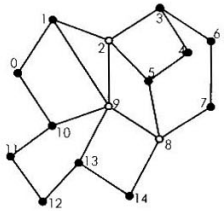
```
Γ ( sigma )
max = |sigma|
ermittle i mit sigma[i]=max
Für alle 0<=k<=max
Falls k!=i
    GE.Konnekt.[k]=Graph.Conn(k,i)
GE.Anzahl d. Knoten = max
Gib GE zurück
```

5.2.6. Beispiel

Es sollen alle möglichen Einbettungen des Teilgraphen



im Graphen



gefunden werden.

Unterschiedlichen Farben der Knoten zeigen einen unterschiedlichen Knotengrad an. Der Algorithmus (Teilstruktursuche) liefert dafür folgende Einbettungen:

/ 0 1 9 10 11 /	/ 1 0 10 9 8 /	/ 1 0 10 9 13 /	/ 2 3 4 5 8 /
/ 2 5 4 3 6 /	/ 2 5 8 9 10 /	/ 2 5 8 9 13 /	/ 2 9 8 5 4 /
/ 3 4 5 2 1 /	/ 3 4 5 2 9 /	/ 4 3 2 5 8 /	/ 4 5 2 3 6 /
/ 5 2 9 8 7 /	/ 5 2 9 8 14 /	/ 5 4 3 2 1 /	/ 5 4 3 2 9 /
/ 5 8 9 2 3 /	/ 8 5 2 9 10 /	/ 8 5 2 9 13 /	/ 8 9 2 5 4 /
/ 8 14 13 9 1 /	/ 8 14 13 9 2 /	/ 8 14 13 9 10 /	/ 9 1 0 10 11 /
/ 9 2 5 8 7 /	/ 9 2 5 8 14 /	/ 9 8 5 2 3 /	/ 9 8 14 13 12 /
/ 9 13 14 8 5 /	/ 9 13 14 8 7 /	/ 10 0 1 9 8 /	/ 10 0 1 9 13 /
/ 13 14 8 9 1 /	/ 13 14 8 9 2 /	/ 13 14 8 9 10 /	/ 14 8 9 13 12 /
/ 14 13 9 8 5 /	/ 14 13 9 8 7 /		

Führt man dagegen eine Subgraphensuche durch, so ergeben sich folgende Einbettungen:

/ 0 1 9 10 11 /	/ 0 10 9 1 2 /	/ 1 0 10 9 2 /	/ 1 0 10 9 8 /
/ 1 0 10 9 13 /	/ 2 3 4 5 8 /	/ 2 5 4 3 6 /	/ 2 5 8 9 1 /
/ 2 5 8 9 10 /	/ 2 5 8 9 13 /	/ 2 9 8 5 4 /	/ 3 4 5 2 1 /
/ 3 4 5 2 9 /	/ 4 3 2 5 8 /	/ 4 5 2 3 6 /	/ 5 2 9 8 7 /
/ 5 2 9 8 14 /	/ 5 4 3 2 1 /	/ 5 4 3 2 9 /	/ 5 8 9 2 1 /
/ 5 8 9 2 3 /	/ 8 5 2 9 1 /	/ 8 5 2 9 10 /	/ 8 5 2 9 13 /
/ 8 9 2 5 4 /	/ 8 14 13 9 1 /	/ 8 14 13 9 2 /	/ 8 14 13 9 10 /
/ 9 1 0 10 11 /	/ 9 2 5 8 7 /	/ 9 2 5 8 14 /	/ 9 8 5 2 1 /

/ 9 8 5 2 3 /	/ 9 8 14 13 12 /	/ 9 10 0 1 2 /	/ 9 13 14 8 5 /
/ 9 13 14 8 7 /	/ 10 0 1 9 2 /	/ 10 0 1 9 8 /	/ 10 0 1 9 13 /
/ 13 14 8 9 1 /	/ 13 14 8 9 2 /	/ 13 14 8 9 10 /	/ 14 8 9 13 12 /
/ 14 13 9 8 5 /	/ 14 13 9 8 7 /		

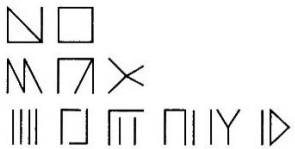
Die acht grau unterlegten sind durch die Subgraphendefinition bedingte, zusätzlich mögliche Einbettungen.

5.2.7. Beispiel

Die folgende Tabelle zeigt die Anzahlen der möglichen Einbettungen von Graphen mit vier Kanten in die Graphen mit fünf Kanten:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	0	4	4	2	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	8	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	120	24	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	6	8	0	4	2	0	0	0	0	2	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	12	0	0	0	24	0	0	0	0	24	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	2	4	0	0	4	0	0	0	0	2	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	12	24	0	0	0	24	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	8	0	8	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	16	0	0	8	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	32	64	48	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	768	1920

In Y - Richtung sind die Graphen mit vier Kanten aufgetragen, in X - Richtung diejenigen mit fünf Kanten. Die elf Graphen mit vier Kanten sind dabei:



Die Anzahl der Isomorphismen ist bei den Knoten mit vier Kanten:

1	2	3	4	5	6	7	8	9	10	11
2	8	24	2	12	2	12	4	8	16	384

und bei den Knoten mit fünf Kanten:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
4	4	2	2	2	10	120	6	8	4	2	2	16	12	2	48	4	12	48	4	4	48	16	16	96	3840

5.3. Schnittstellen zur kanonischen Numerierung

Bedingt durch den Zusammenhang zwischen den Algorithmen zur Ermittlung der Vollstrukturisomorphie und der kanonischen Numerierung ist klar, daß in beiden Fällen identische Schnittstellen verwendet werden können.

Zusätzlich muß jedoch noch bei der kanonischen Numerierung die Klassifikationsfunktion kl zur Bewertung der aktuellen Numerierung definiert werden:

5.3.1. Methode kl

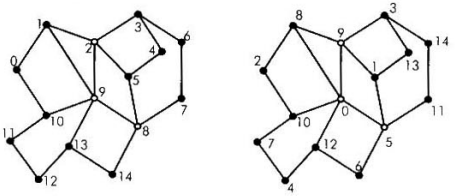
Die Methode kl zur Bewertung der aktuellen Numerierung, sowie die von ihr gelieferte Instanz der Klasse KL - ERGEBNIS wird aus Beispiel 3.2.3.6.4. übernommen. Dort wurden aus der Adjazenzmatrix Vektoren abgelesen, die bezüglich der lexikographischer Ordnung verglichen werden konnten.

5.3.1.1. Implementation

Die Implementation dieser Funktion wird in Abschnitt 6.2.4. nochmals ausführlich erklärt. Sie kann ohne Änderung für Graphen übernommen werden.

5.3.2. Beispiel

Für die kanonische Numerierung betrachten wir nochmals den Graphen aus Beispiel 5.2.6.:



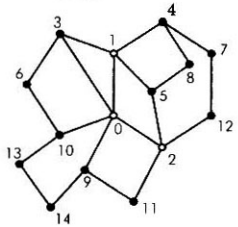
Das Ergebnis der kanonischen Numerierung lautet für den linken Graphen (Ausschnitt aus dem Logfile zum Programm):

```
Maximum canonical form reached:
Canval: Size(15)
Atoms: 9 - 2 - 8 - 1 - 3 - 5 - 0 - 6 - 4 - 13 - 10 - 14 - 7 - 11 - 12 -
Value:110110010001100000100000010000001100100000000100000100000100000010001000010000000
00000001000000000010001
```

Das Ergebnis für den rechten Graphen lautet:

```
Maximum canonical form reached:
Canval: Size(15)
Atoms: 0 - 9 - 5 - 8 - 3 - 1 - 2 - 14 - 13 - 12 - 10 - 6 - 11 - 7 - 4 -
Value:110110010001100000100000010000001100100000000100000100000100000010001000010000000
00000001000000000010001
```

Da der Graph eine triviale Automorphismengruppe besitzt, sind diese Numerierungen identisch:



6. Moleküle

Die folgenden Kapitel geben einen Überblick über die Anwendung der vorgestellten Algorithmen in einem Fachgebiet, in dem diskrete Strukturen bevorzugt zur Modellierung eingesetzt werden.

Es wird dabei weniger auf die Vollständigkeit der Diskretisierung chemischer Strukturen Wert gelegt als vielmehr auf das Aufzeigen der vielfältigen Möglichkeiten zur Darstellung von Strukturen, die mit diesem Algorithmus zur Verfügung stehen.

6.1. Chemische Strukturen

Wie bereits erwähnt, werden diskrete Strukturen, insbesondere Graphen, zur Modellierung chemischer Strukturen verwendet. Dies geschieht vor allem in der organischen Chemie, da die dort vertretenen kovalente Bindungen zwischen Atomen sehr gut durch Graphen repräsentiert werden kann (siehe auch [LK]. Obwohl im graphentheoretischen Molekülmodell beispielsweise keine quantenmechanischen Erkenntnisse wiedergeben werden können, gilt es deshalb in der organischen Chemie als bevorzugtes Modell in weiten Bereichen der Strukturaufklärung, Syntheseplanung und bei der Erforschung biologischer Eigenschaften von Molekülen mit Hilfe der Computerchemie. [KH]

6.1.1. Historie

Um die Vermutungen Alexander von Humbolds von 1797 über die Isomerie von Molekülen (vgl. [BH]) erklären zu können, suchte man nach einer geeigneten Darstellung. Gegen Mitte des 19. Jahrhunderts verwendete Alexander Crum Brown erstmalig eine Moleküldarstellung mit Hilfe von beschrifteten Knoten und Kanten. Molekulare Graphen wurden definiert und als konsequente Weiterentwicklung der Molekülmodelle von Couper, Loschmidt und Kékulé eingesetzt. Die folgenden Abbildungen geben einen Eindruck von der Entwicklung dieser Molekülmodelle.

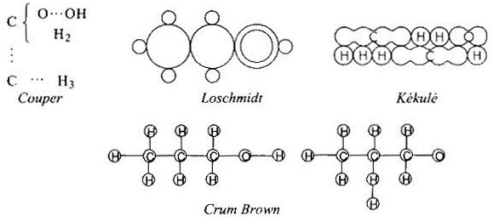


Abbildung 19: *Historische Molekülmodelle*

Wie man heute jedoch weiß, sind die Zusammenhänge in chemischen Strukturen sehr viel komplexer, als es dieses Modell zu vermitteln vermag. Dennoch dient es nach wie vor erfolgreich als Approximation und ist als solche bei einer großen Anzahl von Problemstellungen ausreichend. Hier zeigt sich insbesondere auch die große Bedeutung der Wahl der geeigneten Diskretisierung. Neben der graphentheoretischen Betrachtung von Molekülen haben sich auch andere Modelle etabliert, die für bestimmte Aufgaben besser geeignet sind. Auch die Möglichkeiten der Visualisierung von Informationen aus diesen Modellen sind natürlich erheblich besser als zur Zeit von Crum Brown. Die folgenden

Abbildungen geben davon einen Eindruck, was heute bereits auf Low End Pc's möglich ist.

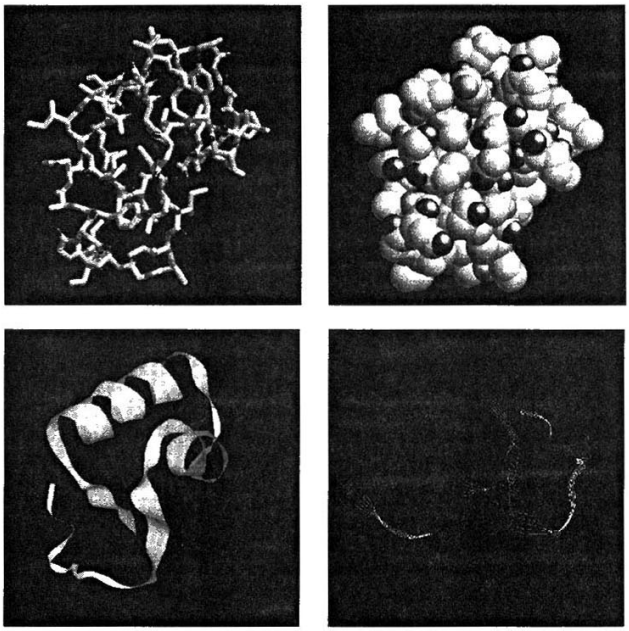


Abbildung 20: *Verschiedene Moleküldarstellungen in RASMOL.*[1]

Ein weiterer, großer Vorteil des graphentheoretischen Strukturmodells ist die große Anzahl von erprobten Verfahren, auf die zurückgegriffen werden kann (siehe auch [DJ]). Ein sehr effizienter Algorithmus zur Konstruktion von Graphen bzw. Molekülen wurde beispielsweise in das bereits mehrfach erwähnte Programmpaket **MOLGEN** integriert, das erfolgreich in der Industrie eingesetzt wird (siehe [MG1]-[MG5]). Anhand des graphentheoretischen Molekülmodells läßt sich auch das Isomorphieproblem sehr gut beschreiben.

6.1.2. Beschreibung chemischer Strukturen

Die gerade durch den Einsatz moderner Technologien in der chemischen Industrie eröffneten Möglichkeiten zur Erkennung und Konstruktion oder Synthetisierung chemischer Strukturen führen zu einer immer weiter ansteigenden Zahl von bekannten Molekülen.

In gleichem Maße stieg dabei auch das Interesse, diese Strukturen vollständig elektronisch zu erfassen, beispielsweise aus patentrechtlichen Gründen, und sie durch eine Vielfalt von Eigenschaften eindeutig

---

1. *RASMOL ist ein Public-Domain Programm, das unter verschiedenen Betriebssystemen die graphische Darstellung von Molekülen und Molekülinformationen erlaubt (siehe beispielsweise ftp://forte.mathematik.uni-bremen.de/pub/comp/visualization/rasmol).*

zu beschreiben, so daß ein schnelles Auffinden anhand dieser Eigenschaften gewährleistet ist.

Die nachfolgenden Kapitel versuchen, einige dieser Merkmale von Molekülen in ein diskretes Modell abzubilden. Darüber hinaus ist es jedoch dem Anwender der Algorithmen überlassen, die für ihn wichtigen Eigenschaften mit in die vorhandenen zu integrieren oder ein völlig neues diskretes Modell zu erstellen.

Um die im vorherigen Kapitel beschriebenen Klassifizierungsalgorithmen anwenden zu können, muß die betrachtete chemische Struktur diskret zu beschreiben sein. Das bedeutet, es müssen Merkmale gefunden werden, die einerseits diskrete Werte annehmen, jedoch andererseits das Objekt so umfassend beschreiben, daß es die für die geforderte Aufgabenstellung relevanten Klassifizierungsmerkmale enthält.

Der Isomorphiebegriff reduziert sich also auf die durch das diskrete Modell beschriebenen Struktureigenschaften.

Demzufolge existieren auch verschiedene Möglichkeiten zur Diskretisierung von Molekülen:

6.2. Diskretisierungen für topologische Identität

Das klassische diskrete Modell für chemische Strukturen berücksichtigte ausschließlich Eigenschaften, die zunächst nur eine Differenzierung der Moleküle auf topologischer Ebene erlauben. Heute werden jedoch auch dreidimensionale Struktureigenschafen ausgenutzt, um beispielsweise im Bereich der Arzneimittelforschung Resultate zu erzielen.

Solange man sich auf eine topologische Identität beschränkt, läßt sich mit der üblichen graphentheoretischen Beschreibung eines Moleküls ein diskretes Abbild der Struktur in die Mathematik finden. Toleranzen sind dort nicht erwünscht. Weitet man seine Betrachtungsweise jedoch in die dreidimensionale Darstellung aus, so kann dies - immer abhängig von den an den Isomorphiebegriff gestellten Ansprüchen - zu sehr komplizierten Abbildungen führen.

Grundvoraussetzung für eine erfolgreiche Identifikation der dreidimensionalen Struktur von Molekülen ist jedoch immer eine Übereinstimmung auf topologischer Ebene. Zu diesem Zwecke wurden die entsprechenden Schnittstellen zu den zuvor beschriebenen Algorithmen zunächst zur Identifizierung von Molekülen auf topologischer Ebene angepaßt.

6.2.1. Graphentheoretische Darstellung

Im klassischen, graphentheoretischen Molekülmodell werden die Atome eines Moleküls als Knoten aufgefaßt, und die Bindungen zwischen den Atomen als Kanten. Nun gibt es zwei Möglichkeiten, die Bindungsvielfachheit in diese Definition mit einzubinden. Zum einen durch den Übergang zu Multigraphen und zum anderen über eine entsprechende Kantenbewertung. Die letztere Möglichkeit ist sicherlich flexibler, wenn man beispielsweise aromatische Ringe oder tautomere Bindungen (siehe dazu [MF]) berücksichtigen möchte. In der Implementation entsprechen sich die beiden Alternativen annähernd.

Die Moleküle, die mit Hilfe dieses Modells dargestellt werden können, sind in der Regel organische Verbindungen. Nur dort läßt sich der Begriff der Bindung zwischen zwei Atomen näherungsweise beschreiben. Betrachtet man beispielsweise Metalle, so ist dort keine Bindung im obigen, klassischen Sinne vorhanden.

Eine besondere Bedeutung besitzen Wasserstoffatome in einem Molekül. Durch ihre univalente Bindungsmöglichkeit werden sie oft nicht explizit als Atom in das Modell aufgenommen. Stattdessen wird multivalenten Atomen die Anzahl der benachbarten Wasserstoffatome als Typmerkmal zugeordnet.

Somit ergibt sich die folgende Definition des Molekülgraphen als *ein* mögliches diskretes Modell:

6.2.1.1. Definition

Sei G = (V, E, B) ein bewerteter, ungerichteter Graph. Die Bewertungsfunktion $B = (B_V, B_E)$ sei definiert durch:

$W_E := \mathbb{N}$, $b_E(e) :=$ Bindungsvielfachheit

$W_V := \mathbb{P} \times \mathbb{N} \times \mathbb{N}$,

$b_V(v) :=$ (Atomtyp, Valenz, Anzahl benachbarter Wasserstoffe)

Wobei $\mathbb{P}$ alle Elementbezeichnungen des chemischen Periodensystems enthält ($\mathbb{P} := \{C, H, N, O, \ldots\}$). Die Knoten werden als Atome bezeichnet. Dann ist G = (V, E, B) ein *topologischer, molekularer Graph.*

6.2.1.2. Definition

- Seien $G^1 = (V^1, E^1, B^1)$ und $G^2 = (V^2, E^2, B^2)$ topologische, molekulare Graphen. $(\varphi, \kappa, \lambda)$ heißt Isomorphismus von G^1 auf G^2 genau dann, wenn $\varphi : V^1 \to V^2$ ein Isomorphismus von $H^1 = (V^1, E^1)$ auf $H^2 = (V^2, E^2)$ ist und $\kappa : W_V^1 \to W_V^2$, κ bijektiv und $\lambda : W_E^1 \to W_E^2$, λ bijektiv, und weiterhin gilt:
  - $\forall\, v \in V^1$ gilt: $\kappa(b_{V^1}(v)) = b_{V^2}(\varphi(v))$ und
  - $\forall\, \{v_1, v_2\} \in E^1$ gilt: $\lambda(b_{E^1}(\{v_1, v_2\})) = b_{E^2}(\{\varphi(v_1), \varphi(v_2)\})$
- G^1 und G^2 heißen isomorph ($G^1 \cong G^2$), wenn es einen Isomorphismus $(\varphi, \kappa, \lambda)$ von G^1 auf G^2 gibt.

6.2.2. Schnittstellen zur Vollstrukturisomorphie

Die vollständige Identität zweier Moleküle beruht auf topologischer Ebene ausschließlich auf der Identität der Adjazenzmatrizen und Atomtypen und Wasserstoffnachbarn der beiden Strukturen.

Die vorgestellten Algorithmen können alle diesbezüglichen Informationen nutzen, um die Komplexität des Isomorphieproblems zu vermindern. Dafür werden folgende Schnittstellen verwendet:

6.2.2.1. Träger

Die Träger der Struktur „molekularer Graph" sind die Atome. Diesen werden zunächst in einer beliebigen Reihenfolge feste Nummern zur eindeutigen Identifizierung zugeordnet (Ausgangsnumerierung).

6.2.2.1.1. Implementation

Für die Implementation des Objektes ATOM führen wir zunächst ein Objekt ATOMTYP ein:

ATOMTYP
Daten:
Name
Valenz
Funktionen:
Operatoren „<,=,>"

Da in der Regel mehrere Atome identische Atomtypen besitzen, muß man im Objekt ATOM nur einen Querverweis auf eine Instanz des entsprechenden Typs setzen, ohne die darin enthaltenen Informationen mehrfach speichern zu müssen.

Die Vergleichsoperatoren ergeben sich aus der natürlichen lexikographischen Ordnung auf den Attributen ($\mathbb{P} \times \mathbb{N}$).

Das Objekt Atom hat dann die folgende Gestalt:

```
ATOM : TRÄGER
Daten:
    Atomtyp
    Wasserstoffe
    Hybridisierung
Funktionen:
    Operatoren „<,=,>"
```

Die Belegung der Attribute Atomtyp und Wasserstoffe erfolgt analog zu der Bewertungsfunktion $b_v(v)$ aus Definition 6.2.1.1..

Beispielhaft sei die Implementation des Vergleichsoperators „=" aufgezeigt:

```
ATOM :: Operator = ( A1, A2 )
Falls A1.Atomtyp() = A2.Atomtyp()
  Falls A1.Wasserstoffe() = A2.Wasserstoffe()
    Falls A1.Hybridisierung() = A2.Hybridisierung()
      Gib „WAHR" zurück
Gib „FALSCH" zurück
```

Implizit wird hier also die Ordnung auf der Klasse Atomtyp verwendet. Das Attribut Hybridisierung ist wie folgt erklärt:

Betrachtet man einen molekularen Graphen, so kann man für jedes Atom eine sog. *Hybridisierung* ermitteln. Diese entspricht einer Mengenpartition der von diesem Atom ausgehenden Bindungen. Die folgende Tabelle zeigt mögliche Hybridisierungen für Atome bis zum Grad 4:

Atomgrad	**Hybridisierungen**
1	
2	
3	
4	

Damit erhalten einzelne Atome mit gleichem Typ und Namen ein zusätzliches Qualifikationsmerkmal. Zur Speicherung dieser zusätzlichen Eigenschaft genügt es meist, die Anzahl der Doppel- und Dreifachbindungen an einem Atom zu bestimmen, da in der Regel keine Atome mit Vierfachbindungen existieren.

Der nächste Schritt würde dann der „*Extended Connectivity*" im MORGAN - Algorithmus [HLM] entsprechen. Dort werden zusätzlich die Hybridisierungen aller benachbarten Atome als Klassifizierungsmerkmal jedes einzelnen Atoms berücksichtigt.

Die Summe aller Attribute (Atomtyp, Wasserstoffe, Hybridisierung) ergibt in der Regel bereits eine relativ gute Ausgangsklassifizierung außer für Sonderfälle (hochsymmetrische Moleküle), wie beispielsweise Fullerene (C_{60} H_{60}). Dort bringt die Vorklassifizierung nach diesen Attributen keine wesentliche Verfeinerung der Atommenge.

Dies sind jedoch nicht die einzigen lokalen Informationen, die man aus einem Molekularen Graphen zu diesem Zeitpunkt gewinnen kann.

6.2.2.1.2. Erweiterungen

In Anlehnung an die Hybridisierung ist auch zu erwägen, ob beispielsweise univalente Atome überhaupt in die Menge der Trägerinstanzen mit aufgenommen werden sollten. Stattdessen wird die Anzahl der benachbarten Wasserstoffe, Chloratome oder sonstiger univalenter Atome als Bestandteil der Atominstanz zu einem Attribut eines betrachteten multivalenten Atoms, wodurch sie bei einem Isomorphietest einander nicht mehr explizit zugeordnet werden müssen, jedoch durch ihr Auftreten als Attribut für die Klassifizierung verwendet werden. Durch die verminderte Ordnung der Trägermenge kann die Komplexität des Isomorphieproblems immer noch am effektivsten verringert werden, wie die nachfolgende Analyse zeigt.

Weitere Merkmale von Atomen wären beispielsweise, ob ein Atom Teil eines aromatischen Rings (vgl. [MF]) oder ein Stereozentrum (vgl. [WIE]) ist.

Grundsätzlich gilt, je mehr Informationen bereits an dieser Stelle berücksichtigt werden, desto kleiner werden die Klassen für die Ausgangsklassifizierung und desto flacher wird der Backtrackbaum im Algorithmus, was natürlich wiederum Einfluß auf die Ausführungsgeschwindigkeit der Algorithmen hat.

Im Anschluß an diesen Abschnitt wird anhand eines Beispiels der Einfluß der Vorklassifizierung auf die Geschwindigkeit des Algorithmus untersucht.

6.2.2.2. Methode rel

Die Isomorphie zweier molekularer Graphen setzt neben der Identität der Knoteninstanzen, die durch die Vorklassifizierung und durch den Algorithmus gewährleistet wird, auch die Identität der Kantenbewertung, d.h. der Konnektivität voraus.

Definiere nun eine Klasse rel - Ergebnis, deren einziges Attribut die Konnektivität ist. Die Vergleichsoperatoren entsprechen den Vergleichsoperatoren auf $\mathbb{N}$.

RE (rel - Ergebnis)
Daten: Konnektivität Funktionen: Operationen: „<,=,>"

Sei $b_E(e)$ die Kantenbewertungsfunktion aus Definition 6.2.1.1. ist. Sei σ die an die Methode *rel* übergebene aktuelle (partielle) Einbettung und v_1 die letzte der Permutation σ hinzugefügte Trägerinstanz. v_2 sei das zweite Argument der Methode *rel*, also die Tägerinstanz, die klassifiziert werden soll. Dann lautet eine mögliche Implementation von *rel*:

top. mol. Graph :: rel(σ, v_2)
Erzeuge Instanz R von REL - ERGEBNIS Setze R.Konnektivität := $b_E((v_1,v_2))$ Gebe R zurück

Beschränkt man sich nicht nur auf die Information über die Vielfachheit einer evtl. vorhandenen Bindung zwischen zwei Atomen der Trägermenge, so kann man eine weitere Verfeinerung erreichen:

6.2.2.2.1. Erweiterungen

Betrachtet man das Molekül als Ganzes, so kann man beispielsweise für zwei beliebige Atome die Anzahl und Länge aller Wege von Atom A nach Atom B bestimmen. Man muß hier jedoch einen Kompromiß bezüglich der Qualität dieser Information und der Dauer ihrer Berechnung finden. Es ist sinnlos, wirklich alle Wege und Weglängen zu berechnen, wenn der Algorithmus dadurch zwar wesentlich schneller abläuft, die Zeit, die für die Bestimmung der Wege und Längen aufgewendet werden mußte, jedoch so groß ist, daß insgesamt kein Geschwindigkeitsvorteil erzielt wurde.

Im Hinblick darauf, daß bei einer späteren, dreidimensionalen Betrachtung von Molekülen Tetraeder-

winkel eine entscheidende Rolle spielen, wird in der Implementation die Wege- und Weglängenbestimmung bis zu einer maximalen Länge von 4 Kanten durchgeführt.

6.2.2.2.2. Implementation

Für diese Implementation der *rel* - Funktion muß zuerst eine erweiterte Klasse RE (rel - Ergebnis) definiert werden. Dies geschieht wie folgt:

```
RE (rel - Ergebnis)
 Daten:
   Konnektivität
   Anzahl 3 - Wege (v1,v2)
   Anzahl 4 - Wege (v1,v2)
 Funktionen:
   Operationen: „<,=,>"
```

Die Implementation der Operatorfunktionen ist damit klar. Beispielhaft sei hier der Test RE1<RE2 definiert:

```
operator RE (RE1) < RE (RE2)
Falls RE1.Konnekt. < RE2.Konnekt.
  Gib WAHR zurück
Falls RE1.Konnekt. =RE2.Konnekt.
  Falls RE1.Anzahl 3-W.< RE2.Anzahl 3-W.
    Gib WAHR zurück
  Falls RE1.Anzahl 3-W. = RE2.Anzahl 3-W.
    Falls RE1.Anzahl 4-W.= RE2.Anzahl 4-W.
      Gib WAHR zurück
Gib FALSCH zurück
```

Die *rel* - Funktion selbst verwendet zur Ermittlung der benötigten Wegeanzahlen einen simplen graphentheoretischen Algorithmus, der alle Wege der Längen 3 und 4 zwischen zwei Knoten findet (vgl. [SDW]).

Im Falle der topologischen Identität sind bei einem Vergleich zweier Strukturen Toleranzen in den Werten von RE nicht erwünscht. Sie spielen erst bei der Betrachtung dreidimensionaler Strukturen eine Rolle.

6.2.2.3. Methode Γ

Die Γ - Funktion zum Testen globaler Eigenschaften der aktuellen Klassifizierung wird zur Bestimmung der topologischen Identität von Molekülgraphen nicht benötigt, da sich die Isomorphie ausschließlich durch lokale Eigenschaften beschreiben läßt.

Mit dieser Implementation der Schnittstelle kann eine effektive Vollstrukturisomorphie von topologischen molekularen Graphen durchgeführt werden.

6.2.2.3.1. Beispiel

Hochsymmetrische Moleküle sind eine Herausforderung für jeden Isomorphiealgorithmus. Durch die Symmetrie wird gerade das Backtracking erforderlich, wobei fast jeder neue Weg im Backtrackbaum zu einer zulässigen Lösung führt.

Die bezüglich Isomorphie zu testende Struktur sei Cyclohexan (C_6H_{12}).

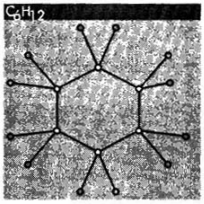


Abbildung 21: *Cyclohexan*

Dem Prinzip des Isomorphiealgorithmus folgend wird jede mögliche Einbettung des Cyclohexan in der zu testenden Struktur gefunden. Die resultierende Anzahl der möglichen 768 Überdeckungen der Vollstruktur ist dabei einfach zu erklären. Um das erste C - Atom zu plazieren, hat man 6 Möglichkeiten zur Auswahl. Unter Berücksichtigung der Konnektivität, die in der Methode *rel* ausgenutzt wird, gibt es für das zweite C - Atom nach dieser Festlegung jedoch nur noch 2 Möglichkeiten. Ist diese Entscheidung getroffen, so ist die Position für die übrigen 4 C - Atome bereits festgelegt. Zusätzlich gibt es jedoch bei jedem C - Atom noch zwei Möglichkeiten, die daran gebundenen beiden H - Atome zu identifizieren. Daraus ergeben sich also die $6 \cdot 2 \cdot (2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2) = 768$ Möglichkeiten. Die Tiefe des Backtrackbaumes ist dabei maximal 8 und damit im Verhältnis zu der Anzahl der betrachteten Atome bereits sehr groß. Zur Reduzierung der Backtracktiefe kann man die bereits zuvor erwähnte Erweiterung der Trägerattribute verwenden, und die H - Atome nicht explizit vom Algorithmus durchlaufen lassen.

Eine Unterscheidung der Wasserstoffatome ist in diesem topologischen Fall weder für die Chemie noch im Sinne eines schnellen Algorithmus von Interesse. Mathematisch gesehen bedeutet dies, daß die lokale Automorphismengruppe des Atoms bereits bekannt ist und nicht mehr explizit berechnet wird.

Damit wird einerseits eine Verfeinerung der Vorklassifizierung möglich, da Atomtypen jetzt zusätzlich anhand der gebundenen univalenten Atome zerlegt werden können. Andererseits wird eine unnötige Erweiterung des Backtrackbaumes durch zusätzliche Symmetrien verhindert.

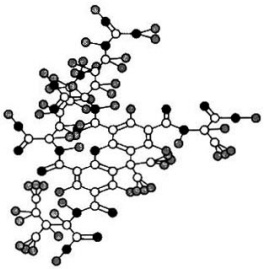
Im obigen Beispiel für C_6H_{12} (Cyclohexan) reduziert dieses Vorgehen die Anzahl der möglichen Einbettungen auf 12 (vorher 768). Desweiteren ist die maximale Tiefe des Backtrackbaumes jetzt nur noch 2 (vorher 8).

6.2.3. Analyse

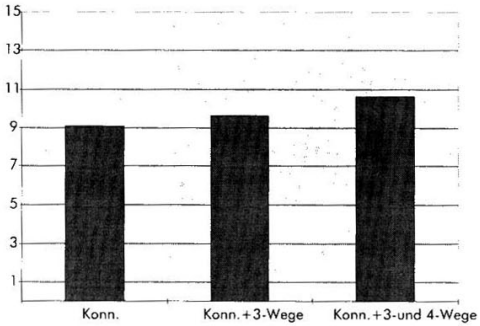
Die Qualität der Schnittstelle und des Algorithmus werden nun an verschiedenen Beispielen analysiert. Ziel dieses Abschnitts ist es, einen Eindruck zu vermitteln, wie bestimmte Änderungen im Aufbau einer Schnittstelle auf den Algorithmus wirken.

Verschiedene Teststrukturen wurden mit Nextor erzeugt, einem System zur Konstruktion kombinatorischer Bibliotheken (siehe auch [WI96]). Im Gegensatz zu MOLGEN haben die 1225 konstruierten Strukturen keine identische Bruttoformel und geben somit ein realistisches Beispiel für eine Datenbankanfrage. Die zu ermittelnde Struktur besitzt die Bruttoformel $C_{40}\,H_{54}\,N_8\,O_{13}$ und ist damit bereits

ein relativ großes Biomolekül.

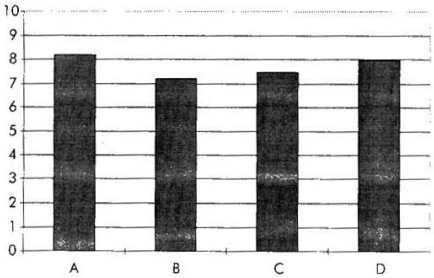


Es wurden mehrere Testläufe durchgeführt, einmal ausschließlich unter Berücksichtigung der Konnektivität in der *ref* -Funktion, einmal unter Berücksichtigung der Anzahl der 3 - Wege und unter Berücksichtigung der Anzahl der 4 - Wege. Das Resultat zeigt die folgende Übersicht:



Der Aufwand zur Berechnung dieser zusätzlichen Merkmale ist hier also größer als der dadurch erzielte Geschwindigkeitsvorteil. Dieses Verhalten tritt insbesondere bei kleineren Strukturen auf, ist jedoch auch von der Art des betrachteten Moleküls abhängig.

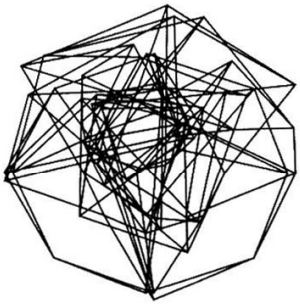
Betrachten wir noch die Auswirkung der Vorklassifizierung auf die Laufzeit des Algorithmus.



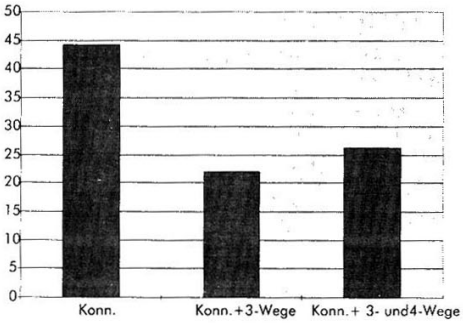
Im Fall A wird ausschließlich der Atomtyp aus den Trägerinstanzen zur Vorklassifizierung verwendet. Wie zu erwarten, wird dadurch die schlechteste Performance erreicht. Im Fall B wird die Anzahl der Einfachbindungen also zusätzliches Kriterium verwendet. Fall C berücksichtigt noch die Anzahl der Doppelbindungen, Fall D die Dreifachbindungen. Da in der getesteten Struktur und auch in keiner der Strukturen, in denen nach ihr gesucht wurde, eine Dreifachbindung vorlag, ist durch dieses Merkmal keine Beschleunigung des Verfahrens eingetreten, im Gegenteil, eine Verschlechterung durch den Aufwand für zusätzliche Tests ist bemerkbar. Die Anzahl der Doppelbindungen spielt in diesem Fall dann auch keine Rolle, da sie - wenn keine Dreifachbindungen auftreten - neben der Valenz und den Einfachbindungen nur redundante Informationen liefern.

In einem letzten Test wird nochmals auf die Bedeutung der Wasserstoffe hingewiesen. Während dem vorherigen Diagramm zufolge eine Vollstruktursuche ohne explizite Berücksichtigung von Wasserstoffen nur ca. 7,5 Sekunden benötigt, steigt die Rechenzeit auf über 30 min an, wenn man die Anzahl der Wasserstoffe nicht mehr als Klassifizierungsmerkmal in die Trägerinstanzen aufnimmt und stattdessen alle möglichen Einbettungen berechnet.

Ein weiteres Beispiel wurde mit Hilfe eines Graphengenerators konstruiert. Das folgende „Molekül" ist eigentlich ein regulärer Graph vom Grade 4 mit 80 Knoten:



Es existieren 1296 Einbettungen dieses Graphen in sich selbst, d.h. 1296 Automorphismen. Die Analyse der Laufzeiten ergibt folgendes Bild:



Erwartungsgemäß tritt eine deutliche Verbesserung des Laufzeitverhaltens bei der Hinzunahme der Anzahl der 3 - Wege ein. Die erweiterte Klassifizierung unter Verwendung der Anzahlen der 4 - Wege erhöht jedoch wieder leicht den Aufwand. Dies liegt einerseits an der Berechnung an sich, jedoch auch am gewählten Beispiel, da die Anzahl der 4 - Wege bei den meisten Knotenpaaren im obigen Graphen übereinstimmt.

6.2.4. Kanonische Numerierung

Die kanonische Numerierung kann prinzipiell mit den gleichen Implementationen der Schnittstellen vorgenommen werden, wie bei der Vollstrukturisomorphie. Zusätzlich wird jedoch noch die Methode *kl* benötigt, die eine Bewertung der aktuellen Numerierung der Trägermenge erlaubt.

6.2.4.1. Methode *kl*

Neben der bereits vorgestellten graphentheoretischen Bewertung der Trägermenge durch eine lexikographische Ordnung auf der Adjazenzmatrix besitzen wir jedoch noch zusätzliche Informationen, die zu einer feineren Bewertung führen.

Sei also σ die aktuelle partielle Numerierung, $|\sigma| = m$. Definiere nun eine Funkton $\mathcal{KL}$ wie folgt:

Für alle $1 \le i \le m$ und $1 \le j \le m$ durchlaufe die Adjazenzmatrix des molekularen Graphen a_{ij} nach dem bereits bekannten Schema:

$$a_{ij} := \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \cancel{1} & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ \cancel{0} & \cancel{1} & 0 & 1 & 0 & 0 & 0 & 1 \\ \cancel{0} & \cancel{0} & \cancel{1} & 0 & 1 & 0 & 1 & 0 \\ \cancel{0} & \cancel{1} & \cancel{0} & \cancel{1} & 0 & 1 & 0 & 0 \\ \cancel{1} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{1} & 0 & 0 & 1 \\ \cancel{0} & \cancel{0} & \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} & 0 & 1 \\ \cancel{1} & \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{1} & \cancel{1} & 0 \end{pmatrix}$$

Erstelle daraus einen Vektor, dessen Elemente nun nicht mehr ausschließlich die Bindungsvielfachheiten a_{ij} enthält, sondern auch die Anzahl der 3- und 4- Wege zwischen Knoten i und j, wie sie bereits bei der Methode *rel* eingesetzt werden.

$\mathcal{KL}(\sigma)$ ist damit ein $\frac{m(m-1)}{2}$ - Tupel, bestehend aus Tripeln:

$$\begin{aligned}\mathcal{KL}(\sigma) = ((&\text{Conn}(2,1), \text{3-Wege}(2,1), \text{4-Wege}(2,1)),\\ &(\text{Conn}(3,1), \text{3-Wege}(3,1), \text{4-Wege}(3,1)),\\ &(\text{Conn}(3,2), \text{3-Wege}(3,2), \text{4-Wege}(3,2)), \ldots)\end{aligned}$$

Die Ordnung auf dieser Menge wird auf natürliche Weise durch die lexikographische Ordnung induziert.

Repräsentiert man nun die rechte Seite der Funktion $\mathcal{KL}$ durch eine abstrakte Datenstruktur KL - ERGEBNIS, so kann man sehr einfach die Methode *kl* implementieren:

6.2.4.2. Implementation

Da die Ergebnisstruktur der $\mathcal{KL}$ - Funktion relativ komplex ist, und die nötigen Informationen (Konnektivität, Wegeanzahlen) bereits durch die Methode *rel* zur Verfügung gestellt werden, gibt man nicht den kompletten zuvor beschriebenen Ergebnisvektor zurück, sondern speichert lediglich die als Parameter übergebene Numerierung und einen Querverweis auf das Objekt Molekül, so daß in den für die *kl* - Funktion notwendigen Operatorfunktionen die benötigten Informationen direkt abgefragt werden können.

Dies hat weiterhin den Vorteil, daß bei einem Vergleich zweier Bewertungen nicht das gesamte Tupel berechnet werden muß, sondern immer nur schrittweise die nächsten Positionen. Wird bei einem lexikographischen Vergleich zweier Ergebnisvektoren bei Element i bereits festgestellt, daß Vektor 1 größer Vektor 2 ist, so müssen die restlichen Elemente des Ergebnisvektors nicht mehr verglichen und bei dieser Implementierung auch nicht berechnet werden.

Sei STRUKTURDATEN eine Klasse, die über die Methode Conn(A,B) Zugriff auf die Konnektivität zwischen den Atominstanzen A und B ermöglicht, sowie über die Methoden 3 - Wege (A,B) und 4 - Wege (A,B) eine Abfrage der Anzahl der Wege der Längen 3 bzw. 4 zwischen A und B ermöglicht.

Die Implementation der Ergebnisstruktur besitzt dann beispielsweise folgendes Aussehen:

KE (kl - Ergebnis)
Daten: Numerierung Strukturdaten Funktionen: Operationen: „<,=,>"

Die Implementation von $\mathcal{kl}$ lautet dann:

top. mol. Graph :: $\mathcal{kl}$ (σ)
Erzeuge eine Instanz KE der Klasse kl - Ergebnis Speichere σ in KE.Numerierung Speichere Verweis auf top.mol.Graph::Strukturdaten in KE.Strukturdaten Gebe KE zurück

Es wird also in diesem Fall lediglich die Numerierung geliefert und ein Verweis auf die Klasse, aus der die weiteren Strukturinformationen bezogen werden können. Die eigentliche Auswertung erfolgt in der Vergleichsfunktion des $\mathcal{kl}$ - Ergebnisses.

Eine mögliche Implementation der Operatorfunktion KE1>KE2 ist:

```
operator KE (KE1) > KE (KE2)

MIN = min (|KE1.Numerierung|, |KE2.Numerierung|)
Falls Min>1
  Für 1<= y <MIN
     Für 0<= x <= y
          A1 := KE1.Numerierung[x], B1 := KE1.Numerierung[y]
          A2 := KE2.Numerierung[x], B2 := KE2.Numerierung[y]
          Falls KE1.Strukturdaten.Conn( A1, B1 ) >
               KE2.Strukturdaten.Conn( A2, B2 )
                  Gib WAHR zurück
          Falls KE1.Strukturdaten.Conn( A1, B1 ) <
               KE2.Strukturdaten.Conn( A2, B2 )
                  Gib FALSCH zurück
          Falls KE1.Strukturdaten.3-Wege( A1, B1 ) >
               KE2.Strukturdaten.3-Wege( A2, B2 )
                  Gib WAHR zurück
          Falls KE1.Strukturdaten.3-Wege( A1, B1 ) <
               KE2.Strukturdaten.3-Wege( A2, B2 )
                  Gib FALSCH zurück
          Falls KE1.Strukturdaten.4-Wege( A1, B1 ) >
               KE2.Strukturdaten.4-Wege( A2, B2 )
                  Gib WAHR zurück
          Falls KE1.Strukturdaten.4-Wege( A1, B1 ) <
               KE2.Strukturdaten.4-Wege( A2, B2 )
                  Gib FALSCH zurück
Falls |KE1.Numerierung| = |KE2.Numerierung|
    Gib FALSCH zurück
Gib MÖGLICH zurück
```

„Gib MÖGLICH zurück" heißt in diesem Fall, daß eine Fortsetzung von KE2 durchaus größer sein kann.

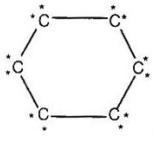
6.2.5. Substruktursuche

Die Problemstellung bei der Substruktursuche unterscheidet sich in bestimmten Teilen wesentlich von der Vollstruktursuche. Die lokalen Schnittstellen besitzen weniger trennende Merkmale, so daß eine Substruktursuche in der Regel langsamer als eine Vollstruktursuche ist. Die wesentlichen Funktionen im Klassifizierungsalgorithmus müssen dieses also berücksichtigen.

Während der Arbeit an MOLGEN entwickelten wir zwei Typen von Substrukturen: Geschlossene und offene Substrukturen. Der Unterschied liegt darin, daß bei geschlossenen Substrukturen nicht erlaubt ist, daß sich freie Valenzen der Substruktur zu einer zusätzlichen Bindung innerhalb der Substruktur verbinden. Die ist jedoch bei den sogenannten offenen Strukturen erlaubt. Diese Unterteilung ist nicht identisch mit der Betrachtung von Teil- und Subgraphen aus 5.2.. Ein Beispiel verdeutlicht die Situation:

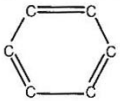
6.2.5.1. Beispiel

Gesucht sind alle Einbettungen der folgenden Struktur in einem Molekül:



* = freie Valenz

Definiert man diese Struktur als geschlossen, so wird sie in der folgenden Struktur nicht erkannt, da sich ja die freien Valenzen nicht innerhalb der Substruktur binden dürfen.



Das Problem dabei ist, daß bei offenen Substrukturen zusätzliche Freiheiten gegeben sind. Das bedeutet, bei der Suche nach offenen Substrukturen werden mindestens so viele gefunden, wie bei der Suche nach geschlossenen Substrukturen. Da bei offenen Substrukturen die Bindungsvielfachheit zwischen Atomen in der Substruktur, die noch nicht gebundene Valenzen besitzen, nicht festgelegt ist, kann eine Verfeinerung nach diesem Merkmal, wie in der Vollstruktursuche, nicht vorgenommen werden.

Es ergeben sich also die beiden folgenden, prinzipiell unterschiedlichen Schnittstellenbeschreibungen:

6.2.5.2. Geschlossene Substrukturen

Es werden wiederum topologische molekulare Graphen betrachtet. Die Definition der Schnittstellenfunktionen erfolgt durch:

6.2.5.2.1. Träger

Der Träger ist - identisch zur Vollstrukturisomorphie - eine Menge von Atominstanzen.

Die Implementation der Klasse ATOM kann analog erfolgen, obwohl in diesem Falle nicht alle Attribute berücksichtigt werden können:

Im Gegensatz zu den bei der Vollstruktursuche verwendeten Attributen ist es bei der Substruktursuche nicht möglich, eine Klassifizierung bezüglich der benachbarten Wasserstoffatome oder allgemeiner, bezüglich der Hybridisierung vorzunehmen. Sind an einem Atom der betrachteten Substruktur mehr als zwei freie Valenzen, so gibt es keinen eindeutigen Hybridisierungszustand mehr. Die Valenzen können in der Teststruktur entweder als Doppelbindung, oder als zwei einzelne Bindungen vorkommen. Dadurch bedingt ist es auch nicht möglich, die übrigen Atome, die evtl. keine freien Valenzen in der Substruktur besitzen, derart zu klassifizieren, da dadurch keine zulässige Ordnung auf der Menge aller Atome definiert ist, was jedoch Voraussetzung für eine Vorklassifizierung ist.

Die Implementation erfolgt also ähnlich der Vollstruktursuche unter Berücksichtigung der zur Verfügung stehenden Merkmale.

```
operator ATOM (A1) < ATOM (A2)
  Falls A1.Atomtyp < A2.Atomtyp
    Gib WAHR zurück
  Gib FALSCH zurück
```

6. 2. 5. 2. 1. 1. Erweiterungen

Als Erweiterungen sind alle Eigenschaften von Atomen zulässig, die nicht durch das Vorhandensein oder Fehlen von freien Valenzen beeinflußt werden. Informationen über umgebende Atome sind jedoch in der Regel immer von der Konnektivität abhängig und scheiden daher aus.

6.2.5.2.2. ref - Funktion

Die Methode *ref* und deren Ergebnis sind wie bei der Vollstruktursuche definiert. Die Implementation erfolgt analog zu 6.2.2.2.2.

6. 2. 5. 2. 2. 1. Erweiterungen

Sind weitere Informationen über die Substruktur bekannt, beispielsweise ob Bindungen in der Substruktur aromatisch sind, so können diese an dieser Stelle verwendet werden.

6.2.5.2.3. Methode Γ

Die Methode Γ zum Testen globaler Eigenschaften der aktuellen Klassifizierung wird zur Bestimmung der topologischen Identität von Molekülgraphen zunächst nicht benötigt, da sich die Isomorphie ausschließlich durch lokale Eigenschaften beschreiben läßt.

6. 2. 5. 2. 3. 1. Erweiterung

Eine Erweiterung der Methode Γ ergibt sich aus dem Problem, daß bei der Verwendung der bisherigen Schnittstelle zur Substruktursuche alle univalenten (hauptsächlich Wasserstoffatome) explizit im Algorithmus zugeordnet werden müssen. Dies liegt, wie bereits erwähnt, daran, daß es nicht möglich ist, Atome nach der Anzahl ihrer benachbarten univalenten Atome zu klassifizieren, da dies - bedingt durch die freien Valenzen - erst feststeht, wenn eine eindeutige Zuordnung eines Atoms der Substruktur auf ein Atom der zu testenden Struktur vorliegt.

Die Methode Γ wird aber - betrachtet man den Algorithmus zur Substruktursuche - immer dann aufgerufen, wenn eine solche Zuordnung zwischen einelementigen Klassen der aktuellen Klassifizierung vorgenommen wird.

Es genügt also, mit Γ zu testen, ob die Abbildung des neu festgelegten Atoms der Substruktur in der zu testenden Struktur mindestens so viele Wasserstoffe besitzt, wie das Substrukturatom. Dies genügt, da durch die Vorklassifizierung in Verbindung mit den Attributen der Atominstanzen bereits die Übereinstimmung der Valenz garantiert wird

Die Implementation erfolgt unter Verwendung des bereits definierten Objektes ATOM. Zuerst wird die Datenstruktur für das Ergebnis der Γ - Funktion beschrieben:

```
GE (gamma - Ergebnis)
Daten:
   Anzahl der Wasserstoffe
Funktionen:
   Operationen: „="
```

Die Γ - Funktion selbst lautet dann einfach:

```
top. mol. Graph :Γ(σ)
max =|σ|
ermittle i mit σ (i)=max
GE.Anzahl Wass. = ATOM[i].Wasserstoffe
Gib GE zurück
```

Die Definition des „=" - Operators ist dann wie folgt:

```
operator GE (GE1) = GE (GE2)
Falls GE2 aus Teilstruktur Swap (GE1,GE2)
Falls GE1.Anzahl Wass. <= GE2.Anzahl Wass.
   Gib WAHR zurück
Gib FALSCH zurück
```

Dabei ist „=" als Funktion „Ist zulässig" zu verstehen.

Die Methode Γ entspricht also einer Abbildung $\Gamma : \mathcal{E} \rightarrow \mathbf{N}$ mit:

$$\Gamma(\sigma) = \text{Anzahl der Wasserstoffe bei Atom } \sigma^{-1}(|\sigma|)$$

Diese - zugegebenermaßen - etwas künstlich wirkende Definition hat den folgenden Hintergrund: Da die Methode Γ in den Algorithmen bei *jeder* neuen Permutation eines Elementes der Trägermenge aufgerufen wird, d.h. jedesmal, wenn Atom A aus der Substruktur auf Atom A´ aus der zu testenden Struktur abgebildet wird, muß die Anzahl der Wasserstoffe auch nur für das zuletzt festgelegte Atom in der Numerierung σ überprüft werden. Das zuletzt hinzugefügte Atome ist jedoch das mit der Nummer $\sigma^{-1}(|\sigma|)$, da es durch $\sigma = \sigma \cdot (A \rightarrow |\sigma| + 1)$ umnumeriert wurde.

Um nun die Zulässigkeit einer neuen Abbildung $A \rightarrow A'$ bei nicht explizit berücksichtigten Wasserstoffen zu gewährleisten, muß $\Gamma(\sigma) \leq_G \Gamma(\sigma')$ gelten. σ ist die neue Numerierung der Substruktur und σ' die der zu testenden Struktur.

Mit diesen Schnittstellen und Erweiterungen können geschlossene Substrukturen effektiv ermittelt werden. Ein Beispiel folgt im Anschluß an den nächsten Abschnitt.

6.2.5.3. Offene Substrukturen

Offene Substrukturen setzen eine höhere Flexibilität bei der Bindungsdefinition voraus. Die Vorklassifizierung wird jedoch nicht davon betroffen.

6.2.5.3.1. Träger

Der Träger ist identisch wie bei geschlossenen Substrukturen durch die Klasse ATOM implementiert.

6.2.5.3.2. Methode rel

Die Methode *rel* besitzt bei der offenen Substruktursuche keine Aufgabe. Die Erklärung hierfür ist sehr einfach. Da die Methode *rel* zur Verfeinerung von Klassen eingesetzt wird, indem die *rel* - Ergebnisse aller Klassenelemente bezüglich eines neu festgelegten Atoms berechnet werden, ist eine diskrete Schranke notwendig, mit der man zwei Klassenelemente unterscheiden kann.

Bei der geschlossenen Substruktursuche ist dies einfach die Bindungsvielfachheit oder die Anzahl der Winkel von v_1 nach v_2. Da jedoch nach Definition der offenen Substrukturen auch neue Bindungen innerhalb der Struktur bestehen können, die Konnektivität also variabel ist, kann diesbezüglich keine Trennung der Klassenelemente vorgenommen werden.

Es findet also keine Verfeinerung nach der Festlegung einer neuen, einelementigen Klasse statt.

6.2.5.3.3. Implementation

Die Implementation der Methode *rel* erfolgt in diesem trivialen Fall so, daß eine Klasse RE definiert

wird, die keine Daten enthält. Die Operatoren auf diesem Objekt liefern immer identische Werte.

RE (rel - Ergebnis)
Daten: Funktionen: Operationen: „<,=,>"

operator RE (RE1) < RE (RE2)
Gib „UNDEFINIERT" zurück

Die Implementation der anderen Operatorfunktionen ist analog.

Die Implementation von *rel* selbst lautet dann:

top. mol. Graph :: rel(σ, v)
Erzeuge Instanz R von REL - ERGEBNIS Gebe R zurück

6.2.5.3.4. Methode Γ

Da also mit *rel* keine weitere Klassifizierung der Daten erreicht werden kann, muß die Methode Γ diese Aufgabe übernehmen, um nicht tatsächlich alle Kombinationen via Backtracking durchlaufen zu müssen.

Die Methode Γ wird immer dann aufgerufen, wenn eine neue Abbildung eines Elementes der Substruktur auf ein Element der zu testenden Struktur abgebildet wird.

Seien $v_1 \rightarrow w_1, \ldots, v_k \rightarrow w_k$ die bisher festgelegten partiellen Permutationen mit $v_i \in \mathcal{S}.S$ und $w_i \in \mathcal{T}.S$. Es werde die neue Permutation $v_{k+1} \rightarrow w_{k+1}$ hinzugefügt.

Berechne nun für alle $v_1,\ldots,v_k$ die tatsächliche Bindungsvielfachheit zwischen v_{k+1} und v_i, sowie die maximal mögliche. Führe dies analog für $w_1,\ldots,w_k$ und w_{k+1} aus.

Die maximal mögliche Bindungsvielfachheit berechnet sich aus der aktuellen zuzüglich dem Minimum der freien Valenzen der beiden beteiligten Atome.

Die Bindungsvielfachheit zwischen den Atomen w_i und w_j aus der zu testenden Struktur muß dann mindestens so groß sein wie die zwischen den Atomen v_i und v_j in der Substruktur, darf jedoch die maximal mögliche Anzahl nicht überschreiten.

6. 2. 5. 3. 4. 1. Implementation

Die Implementation erfolgt unter Verwendung des bereits definierten Objektes ATOM. Zuerst wird die Datenstruktur für das Ergebnis der Γ - Funktion:

GE (gamma - Ergebnis)
Daten: Feld für die Bindungsvielfachheit Feld für die Maximale Bindungsvielfachheit Feldlänge Funktionen: Operationen: „="

Die Definition des „=" - Operators ist dann wie folgt:

```
operator GE (GE1) = GE (GE2)
Falls GE2 aus Teilstruktur Swap (GE1,GE2)
Für alle Feldeinträge i
Falls GE1.Bindungsv.[i] <= GE2.Bindungsv.[i]
    Falls GE1.MaxBind[i] >= GE2.Bindungsv.[i]
    Gib WAHR zurück
Gib FALSCH zurück
```

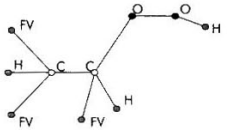
Der „=" - Operator überprüft also wieder die Zulässigkeit der Abbildungen. Die Γ - Funktion selbst zur Ermittlung der Werte lautet dann einfach:

```
top. mol. Graph :: Γ(σ)
max = |σ|
ermittle i mit σ(i)=max
Für alle 0<=k<=max
Falls k!=i
    GE.Bindungsv.[k]=Struktur.Conn(k,i)
    GE.MaxBind[k]=min(Atom[k].FreieValenzen,
        Atom[i].FreieValenzen)+GE.Bindungsv.[k]
Gib GE zurück
```

Wie aus den obigen Definitionen hervorgeht, ist die Suche nach offenen Substrukturen etwas aufwendiger als die Suche nach geschlossenen. Bereits bei kleinen Beispielen zeigt sich diese Problematik:

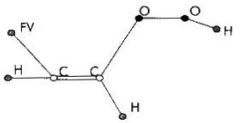
6.2.5.4. Beispiel 1

Mit MOLGEN wurden alle 13190 Isomere von $C_8H_{16}O_2$ erzeugt. Es soll nun die folgende Substruktur



sowohl als geschlossene als auch als offene Substruktur gesucht werden.

Man beachte, daß bei der Suche nach offenen Substrukturen auch Strukturen mit der folgenden Teilstruktur gefunden werden:



Im ersten Fall gibt es 488 Treffer, im zweiten Fall 505. Betrachtet man die Rechenzeiten, so zeigt sich dort bereits die zuvor angesprochene höhere Komplexität des zweiten Falles. Die Suche nach der geschlossener Substruktur benötigt im Schnitt 3,1 Millisekunden pro Test, wohingegen bei offener Struktursuche 3,4 Millisekunden benötigt werden.

Es kann also bei großen Mengen von betrachteten Strukturen und größeren Teilstrukturen durchaus sinnvoll (zeitsparend) sein, sich im voraus Gedanken über dieses Problem zu machen.

6.2.5.5. Beispiel 2

Ein weiteres Beispiel zeigt ein Verhalten des Algorithmus, wo geschlossene Substrukturen evtl. nicht zum gewünschten oder erwarteten Ergebnis führen:

Zur Bruttoformel C_6H_6 erzeugt MOLGEN ohne Angabe von Restriktionen genau 217 Isomere. Als Substruktur dient nun der Benzolring aus Beispiel 6.2.2.3.1. in der Form ohne vorgegebenen Doppelbindungen. Erwartungsgemäß enthält ihn keines der 217 Isomere, wenn er als geschlossene Substruktur deklariert wird. Als offene Substruktur gesucht, findet man neben dem erwarteten Benzolring noch weitere 51 Strukturen, von denen einige hier abgebildet sind:

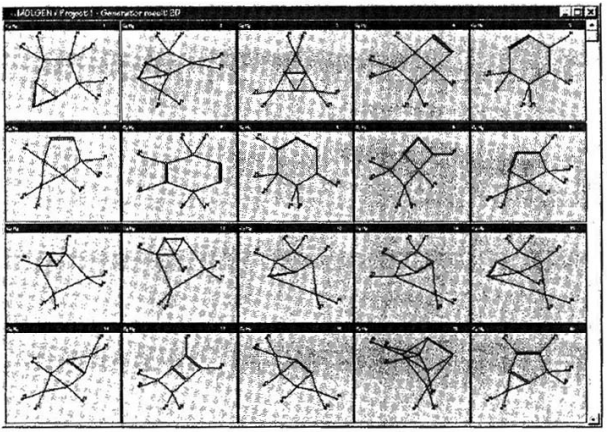


Abbildung 22: *Ergebnis der offenen Substruktursuche*

Darunter sind auch Strukturen, bei denen der vorgegebene Sechsring nicht sofort identifiziert werden kann, da sich die freien Valenzen aufgrund der vorgegebenen Definition natürlich nicht nur zu Doppelbindungen in der Substruktur schließen, sondern auch zusätzliche, gänzlich neue Bindungen erzeugen können.

Möchte man jedoch den vorgegebenen Sechsring quasi zur Ringsuche verwenden, ist beispielsweise eine Struktur wie

aus der vorherigen Ergebnismenge nicht erwünscht. Die Ringdefinition, die in der Chemie verwendet wird, geht davon aus, daß die Atome eines Rings zu den übrigen Ringatomen, außer zu den direkt benachbarten, nicht über eine direkte Kante verbunden sein dürfen. Insofern enthält die obige Struktur zwar Drei- und Vierringe, aber eben keinen Sechsring.

Es ist jedoch sehr einfach, die Strukturdefinition dahingehend abzuändern, daß diese Ringbedingung erfüllt ist. In der Methode Γ muß zusätzlich berücksichtigt werden, daß nur dann freie Valenzen zwischen Atomen der Substruktur gebunden werden dürfen, wenn bereits eine Kante zwischen den beiden Atomen existiert.

Die entsprechend abgeänderte Implementation der Operatorfunktion „=" auf GE hat die Form:

```
operator GE (GE1) = GE (GE2)
Falls GE2 aus Teilstruktur Swap (GE1,GE2)
Für alle Feldeinträge i
Falls GE1.Bindungsv.[i] <= GE2.Bindungsv.[i]
    Falls GE1.MaxBind[i] >= GE2.Bindungsv.[i]
        | Falls GE1.Bindungsv.[i] == 0 und GE2.Bindungsv.[i] >0
            Gib FALSCH zurück
    Gib WAHR zurück
Gib FALSCH zurück
```

Führt man die Suche mit dieser Definition nochmals aus, so findet man das gewünschte Ergebnis:

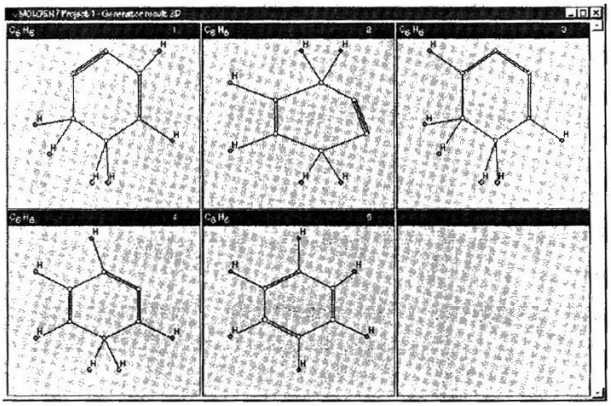


Abbildung 23: *Ringsuche mit offenen Substrukturen*

Das gleiche Ergebnis wird auch mit MOLGEN erzielt, wenn bei der Konstruktion der Strukturen bereits 6-Ringe als notwendige Bedingung angegeben werden.

Weitere Informationen zur 2D - Substruktursuche in Molekülen findet man auch in Abschnitt 7.1.. Dort werden einige spezielle Verfahren vorgestellt.

6.2.6. Schnittstellen zur Kanonischen Numerierung

Prinzipiell kann man für die kanonische Numerierung topologischer molekularer Strukturen die Schnittstellen zur Vollstrukturisomorphie verwenden. Zusätzlich wird jedoch noch die Methode kf zur Bewertung der aktuellen Numerierung benötigt. Die Implementation erfolgt analog zu den bei der Beschreibung des Algorithmus vorgestellten Beispielen anhand der Adjazenzmatrix des molekularen Graphen.

6.3. Diskretisierungen zur Erkennung räumlicher Ähnlichkeit

Ausgehend von einer topologisch korrekten Einbettung kann man nun verschiedene Kriterien für eine dreidimensionale Übereinstimmung der Molekülstrukturen festlegen. Ein erster allgemeiner Ansatz für nicht unbedingt chemische räumliche Strukturen soll eine kurze Einführung in die Problematik liefern,

bevor spezielle chemische Überlegungen mit in den Algorithmus einfließen.

Da zwei räumliche Plazierungen von Molekülen in der Regel nicht absolut identisch sind, darf nicht mehr die exakte Übereinstimmung der Koordinaten überprüft werden, sondern es müssen gewisse Toleranzen bei der Darstellung zugelassen werden. In diesem Fall verwendet man also den Begriff der Ähnlichkeitsabbildung, wie eingangs definiert.

6.3.1. Erster, allgemeiner Ansatz

Bei der räumlichen Plazierung von Molekülen werden für jedes Atom dreidimensionale Koordinaten berechnet. Dies kann mit Hilfe unterschiedlicher Methoden geschehen, eine davon ist in 9.1. beschrieben.

Die räumliche Darstellung eines Moleküls ist eindeutig durch die Abstände aller Atome zueinander beschreibbar. Stimmen alle Abstände bei zwei betrachteten Strukturen überein, sind die Strukturen identisch. Dabei sei natürlich eine topologische Identität vorausgesetzt.

Für die Überprüfung mit Hilfe des Isomorphiealgorithmus gibt es zwei Möglichkeiten.

- Finde alle topologischen Einbettungen und überprüfe dort die Abstände
- Benutze die Abstände bereits als zusätzliches Kriterium zur Verfeinerung während des Algorithmus

Bei kleinen Molekülen oder Molekülen mit geringer Symmetrie ist die erste Möglichkeit evtl. schneller. Da jedoch ein zusätzliches Merkmal bei der Verfeinerung eine Verringerung der Tiefe des Backtrackbaumes bewirkt, und eine Verkürzung das Backtracking die effektivste Beschleunigung des Verfahrens erlaubt, ist die zweite Möglichkeit vorzuziehen.

Dieser Ansatz wurde auch testweise implementiert und an verschiedenen (einfachen) chemischen Strukturen getestet.

Für kompliziertere Strukturen bzw. Substruktursuche ist er jedoch nicht geeignet, da die meisten Strukturen, die im chemischen Sinne isomorph zueinander sind, nicht unbedingt den sehr strikten Beschränkungen dieser Diskretisierung unterliegen.

6.3.1.1. Träger

Träger ist wiederum eine Menge von Instanzen des Typs ATOM. Die Klasse ATOMTYP wird übernommen. In der Klasse ATOM werden zusätzlich noch Variablen zur Speicherung der Koordinaten aufgenommen. Die Klasse ATOM hat dann die folgende Gestalt:

ATOM
Daten: Atomtyp Wasserstoffe Hybridisierung X,Y,Z
Funktionen: Operatoren „<,=,>"

Die Vergleichsoperatoren, nötig für die Vorklassifizierung der Atominstanzen, wird aus der topologischen Isomorphie übernommen. Es ist nicht sinnvoll, die Koordinaten zur Vorklassifizierung mit einzubeziehen, da zwei Moleküle in der Regel nicht in identischer räumlicher Ausrichtung vorliegen. Die Vorklassifizierung ist damit identisch mit der bei der topologischen Isomorphie verwendeten.

6.3.1.1.1. Erweiterungen

Zur Vorklassifizierung können alle lokalen Eigenschaften berücksichtigt werden. Dies wird zwar nicht von absoluten Koordinaten erfüllt, es ist jedoch denkbar, ähnlich der erweiterten Konnektivität bei der topologischen Isomorphie, die Abstände zu den benachbarten Atomen als zusätzliches Kriterium zur Unterscheidung der Atome heranzuziehen.

6.3.1.2. Methode rel

Sei G = (V, E, B) ein molekularer Graph. Sei:

- $b(\{v_1, v_2\})$ die Bindungsvielfachheit zwischen den Knoten v_1 und v_2.
- ${}^3N(\{v_1, v_2\})$ die Anzahl der Wege der Länge 3 von v_1 nach v_2.
- ${}^4N(\{v_1, v_2\})$ die Anzahl der Wege der Länge 4 von v_1 nach v_2.
- $d(\{v_1, v_2\})$ der euklidische Abstand zwischen den Knoten v_1 und v_2.

Definiere eine Klasse *rel* - Ergebnis der Gestalt:

```
RE (rel - Ergebnis)
Daten:
   Konnektivität
   Anzahl 3 - Wege (v1,v2)
   Anzahl 4 - Wege (v1,v2)
   Distanz (v1,v2)
Funktionen:
   Operationen: „<,="
```

Die Methode *rel* sei implementiert als:

```
Molekül :: rel (σ, v2)
Erzeuge Instanz R von REL - ERGEBNIS
Setze R.Konnektivität := b({v1, v2})
Setze R.Anzahl 3 - Wege (v1,v2) := 3N({v1, v2})
Setze R.Anzahl 4 - Wege (v1,v2) := 4N({v1, v2})
Setze Distanz (v1,v2) := d({v1, v2})
Gebe R zurück
```

v_1 sei wie zuvor definiert. Zum Vergleich der Werte der *rel* - Ergebnisse gibt man einen zulässigen Toleranzwert für den Abstand der beiden Atome vor:

Bei der Implementierung der Operatorfunktionen werden zwar bei der Konnektivität und den Winkelanzahlen nur identische Werte für die Vergleiche erlaubt, bei der Bewertung des Abstandes kann jetzt jedoch eine Toleranz vorgegeben werden. Üblicherweise verwendet man eine Prozentangabe, um die sich die Werte unterscheiden dürfen.

5 % erlaubte Toleranz heißen, daß zwei Abstände d_1 und d_2 als gleich anzusehen sind, wenn gilt:

$$|d_1 - d_2| < \frac{(|d_1| + |d_2|)}{2} 0.05$$

Die Implementation der Operatorfunktion RE1=RE2 lautet dann:

```
operator RE (RE1) =RE (RE2)
Falls RE1.Konnekt. != RE2.Konnekt.
   Gib FALSCH zurück
Falls RE1.Anzahl 3-W. != RE2.Anzahl 3-W.
   Gib FALSCH zurück
Falls RE1.Anzahl 4-W. != RE2.Anzahl 4-W
   Gib FALSCH zurück
Falls |d1 - d2| > (|d1| + |d2|)/2 0.05
   Gib FALSCH zurück
Gib WAHR zurück
```

Diese Implementation bietet zusätzlich die Möglichkeit, die Abstände zwischen direkt verbundenen

Atomen und nicht verbundenen Atomen unterschiedlich zu bewerten. Man könnte hiermit festlegen, daß beispielsweise die direkt gebunden Atomen strikteren Abstandsbedingungen genügen müssen als nicht gebundene. In der folgenden Implementation beispielsweise 10 % und 5%:

```
operator RE (RE1) =RE (RE2)
Falls RE1.Konnekt. != RE2.Konnekt.
   Gib FALSCH zurück
Falls RE1.Anzahl 3-W. != RE2.Anzahl 3-W.
   Gib FALSCH zurück
Falls RE1.Anzahl 4-W. != RE2.Anzahl 4-W
   Gib FALSCH zurück
Falls RE1.Konnekt. = RE2.Konnekt. = 1
   Falls |d1 - d2| > ((|d1| + |d2|)/2) 0.10
      Gib FALSCH zurück
   Gib WAHR zurück
sonst
   Falls |d1 - d2| > ((|d1| + |d2|)/2) 0.05
      Gib FALSCH zurück
   Gib WAHR zurück
```

6.3.1.2.1. Erweiterungen

Erweiterungen können analog zur topologischen Isomorphie vorgenommen werden.

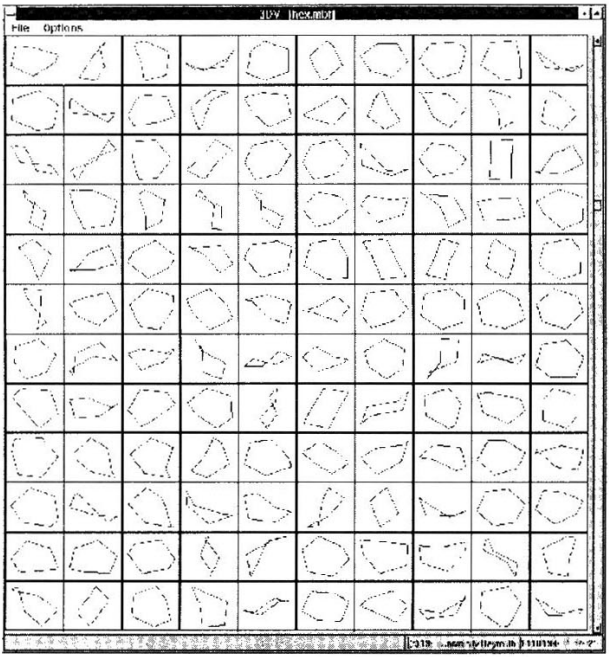
6.3.1.3. Γ - Funktion

Die Methode Γ zum Testen globaler Eigenschaften der aktuellen Klassifizierung wird bei dieser Diskretisierung nicht benötigt.

6.3.1.3.1. Beispiel

Als Beispiel wird wieder das Cyclohexan (C_6H_{12}) verwendet. Mit Hilfe einer Kraftfeldoptimierung nach dem MM2 - Modell (siehe auch 9.1.) für chemische Strukturen wurden 1.000 3-D Plazierungen von Cyclohexan berechnet. Einen kleinen Überblick über die Vielzahl der Darstellungen gibt das fol-

gende Bild:



Um eine Klassifizierung dieser Strukturen zu erreichen, ist es wünschenswert, alle im Sinne der vorherigen Definitionen ähnlichen Strukturen zu eliminieren. Dazu führt man den folgenden Basisalgorithmus aus:

Basisberechnung:

1. Nimm die erste Struktur in die Basis auf
2. Laufe durch alle vorhandenen Strukturen
3. Ist die aktuelle Struktur nicht ähnlich zu einer vorhandenen Basisstruktur entsprechend vorigem Algorithmus, so nimm sie mit in die Basis auf.

Die Verteilung der tausend Strukturen auf eine so berechnete Basis zeigt die folgende Tabelle:

Nummer	Energie	Anzahl Treffer
0	103.793	692
1	99.0856	239
2	405.001	31
3	296.864	6

Nummer	Energie	Anzahl Treffer
4	647.042	6
5	519.417	3
6	263.574	2
7	225.571	3
8	1963.56	2
9	279.434	2
10	628.404	1
11	845.966	2
12	1223.93	1
13	435.667	1
14	515.358	2
15	1992.06	1
16	1461.47	1
17	946.539	1
18	105.081	2
19	931.104	2
20	332.061	1

Es wurden insgesamt 1632 paarweise Vergleiche bei einer durchschnittlichen Geschwindigkeit von 0.0101103 Sekunden/Vergleich durchgeführt. Alle Strukturen sind ähnlich zu 21 Basisstrukturen. Die nachfolgende Tabelle zeigt eine Energiestatistik. Während der Energiewert in der obigen Tabelle ausschließlich den Energiewert der Basisstruktur zeigt, werden nachfolgend Minimum, Maximum und Durchschnitt der Energiewerte der zur Basisstruktur ähnlichen Moleküle aufgelistet. Basisstrukturen, die nur einen Treffer aufweisen, sind darin nicht berücksichtigt.

#	Energie	Treffer	Min	Max	Schnitt
1	103.793	692	103.544	276.176	103.828
2	99.0856	239	98.4754	99.8217	98.6877
3	405.001	31	198.035	972.832	299.195
4	296.864	6	109.585	1207.42	411.976
5	647.042	6	255.346	647.042	316.645
6	519.417	3	396.226	519.417	457.821
7	225.571	3	213.406	225.571	218.421
8	515.358	2	515.358	515.358	515.358
9	1963.56	2	1963.56	1963.56	1963.56
10	279.434	2	279.434	335.185	307.31
11	105.081	2	105.081	105.081	105.081
12	845.966	2	428.418	845.966	637.192

#	Energie	Treffer	Min	Max	Schnitt
13	931.104	2	931.104	931.104	931.104
14	263.574	2	263.574	268.258	265.916

Betrachtet man nun die Strukturen mit den meisten Treffern, beispielsweise mit dem 3D - Visualisierungstool 3DV (siehe 11.1.), so ergibt sich folgendes Bild:

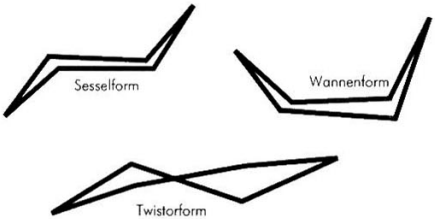


Abbildung 24: *Konformationen des Cyclohexan*

Mehr als 96% der berechneten dreidimensionalen Plazierungen sind ähnlich zu den drei Strukturen mit den meisten Treffern. Läßt man eine größere Differenz bei den molekülinternen Abständen zu, so wird sich diese Quote sicherlich noch weiter verbessern. Tatsächlich kommt Cyclohexan in drei verschiedenen räumlichen Konfigurationen vor. Man bezeichnet diese als Sessel-, Wannen- oder Twistorform. ([UD])

Zieht man die erzielten Energiewerte innerhalb der Strukturen zur Klassifizierung heran, so erhält man bei der Betrachtung des vollständigen Ausgangssatzes von 1000 Strukturen die folgende Energieverteilung:

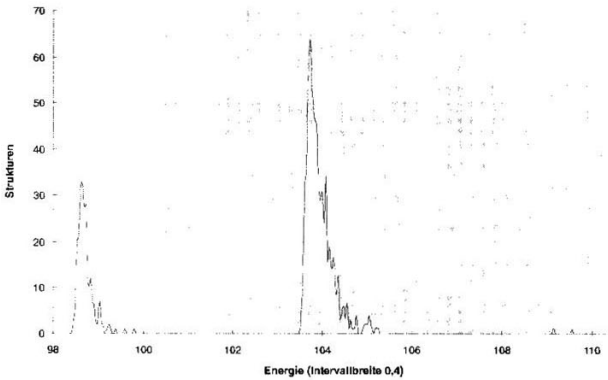


Abbildung 25: *Energieverteilung von Cyclohexan nach 1000 Optimierungen*

Das Diagramm zeigt den relevanten Ausschnitt aus dem Energiespektrum. Energiewerte größer als 110

sind - durch das Optimierungsprinzip bedingte - äußerst unwahrscheinliche dreidimensionale Repräsentanten der Struktur.

Es liegt die Vermutung nahe, daß Strukturen, die ähnliche Energiewerte aufweisen, auch im Sinne der vorherigen Definition ähnlich sind. Mehr zu diese Vermutung findet man später in Kapitel 8.4..

6.3.2. Zweiter, allgemeiner Ansatz

Der erste Ansatz, bei dem ausschließlich die Abstände innerhalb der Struktur berücksichtigt werden, eignet sich - wie bereits erwähnt - ausschließlich für kleine oder sehr starre Moleküle, da die durch die Bindungslängen vorgegebenen Restriktionen sehr stark sind.

Neben den Bindungslängen können jedoch auch die Bindungswinkel und auch Tetraederwinkel zur Klassifizierung der Moleküle herangezogen werden.

Die Bindungs- und Tetraederwinkeln besitzen einen relativ lokalen Charakter (bis zu 3 benachbarte Atome). Beziehungen zu weiter entfernten Atomen gehen also nicht explizit in die Klassifizierung mit ein. Natürlich kann man auch weiterhin eine Klassifizierung bezüglich den weiter entfernten Atomen durch eine Bewertung solcher Bindungslängen mit 50% oder mehr Toleranz vornehmen.

Durch den lokalen Charakter der Winkel ist es unter Umständen nicht möglich zwischen Molekülen zu differenzieren, bei denen Unterscheidungsmerkmale erst bei der Betrachtung von 5 oder mehr Atomen sichtbar sind.

Um solche Probleme zu vermeiden, wird später die Γ -Funktion entsprechend modifiziert.

6.3.2.1. Träger

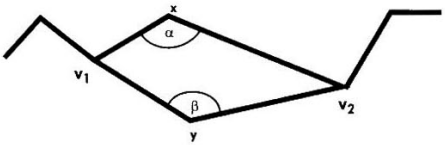
Analog zum ersten Ansatz (6.3.1.1.)

6.3.2.2. Methode rel

Als Erweiterung zum ersten Ansatz werden nun die Bindungs- und Tetraederwinkel mit in die Klassifizierung aufgenommen.

6.3.2.2.1. Bindungswinkel

Als Bindungswinkel bezeichnet man die Winkel, die durch eine Folge von drei Atomen zustande kommen. Zur Klassifizierung wird eine Liste aller möglichen Bindungswinkel zwischen zwei Knoten v_1 und v_2 erstellt:

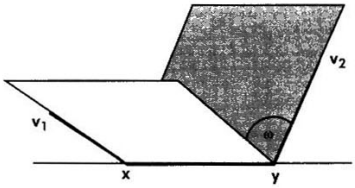


In obiger Abbildung werden also sowohl $\alpha = (v_1, x, v_2)$ als auch $\beta = (v_1, y, v_2)$ herangezogen. Diese Liste wird aufsteigend sortiert. Sei ${}^3L(\{v_1, v_2\})$ diese sortierte Liste aller Bindungswinkel von v_1 nach v_2. Die Bindungswinkel sind Bewertungen dreistelliger Relationen.

6.3.2.2.2. Tetraederwinkel

Als Tetraederwinkel bezeichnet man den Winkel, der entsteht wenn man in einer Folge von vier verbundenen Atomen den Winkel zwischen den durch die Knoten 1,2,3 und 2,3,4 aufgestellten Ebenen

mißt:



Aufgrund der Symmetrie gilt $\omega_{v_1, x, y, v_2} = \omega_{v_2, y, x, v_1}$. Es werden wieder alle Tetraederwinkel berücksichtigt, also alle Wege von v_1 nach v_2 über 2 zusätzliche Knoten. Die sortierte Liste aller Tetraederwinkel zwischen v_1 und v_2 bezeichnen wir als ${}^4L(\{v_1, v_2\})$.

Damit läßt sich *ref* wie folgt konstruieren:

Sei G = (V, E, B) ein molekularer Graph. Sei:

- $b(\{v_1, v_2\})$ die Bindungsvielfachheit zwischen den Knoten v_1 und v_2.
- ${}^3N(\{v_1, v_2\})$ die Anzahl der Wege der Länge 3 von v_1 nach v_2.
- ${}^3L(\{v_1, v_2\})$ die sortierte Liste der Bindungswinkel zwischen v_1 nach v_2.
- ${}^4N(\{v_1, v_2\})$ die Anzahl der Wege der Länge 4 von v_1 nach v_2.
- ${}^4L(\{v_1, v_2\})$ die sortierte Liste der Tetraederwinkel zwischen v_1 nach v_2.

Das Objekt RE (*ref* - Ergebnis) besitzt hier dann beispielsweise folgende Gestalt:

RE (rel - Ergebnis)
Daten: Konnektivität Anzahl 3 - Winkel (v_1,v_2) Liste der Bindungswinkel Anzahl 4 - Winkel (v_1,v_2) Liste der Tetraederwinkel
Funktionen: Operationen: „<, ="

Die dazugehörige Methode *ref*:

Molekül :: rel (σ, v_2)
Erzeuge Instanz R von REL - ERGEBNIS Setze R.Konnektivität := $b(\{v_1, v_2\})$ Setze R.Anzahl 3 - Wege (v_1,v_2) := ${}^3N(\{v_1, v_2\})$ Setze R.Liste der Bindungswinkel := ${}^3L(\{v_1, v_2\})$ Setze R.Anzahl 4 - Wege (v_1,v_2) := ${}^4N(\{v_1, v_2\})$ Setze R.Liste der Tetraederwinkel := ${}^4L(\{v_1, v_2\})$ Gebe R zurück

v_1 sei analog wie zuvor die zuletzt in die *ref* übergebenen Permutation aufgenommene Atominstanz. Zum Vergleich der *ref* - Ergebnisse gibt man zulässige Toleranzwerte für die Bindungs- und Tetraederwinkel vor, so daß auch nicht exakt identische Werte zu einer zulässigen Ähnlichkeitsabbildung führen können. Nach einigen Tests zeigt sich, daß bei Größenordnungen zwischen 10 und 25 Grad die besten

Ergebnisse erzielt werden. Dabei sollte die Einschränkung für die Tetraederwinkel weniger stark sein als bei den Bindungswinkeln.

Seien also die zulässigen Winkeldifferenzen δ_2 und δ_1 vorgegeben. Eine als Beispiel implementierte Operatorfunktion lautet dann:

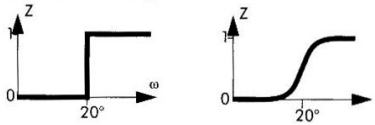
```
operator RE (RE1) = RE (RE2)
Falls RE1.Konnekt. != RE2.Konnekt.
   Gib FALSCH zurück
Falls RE1.Anzahl 3-W. != RE2.Anzahl 3-W.
   Gib FALSCH zurück
Falls RE1.Anzahl 4-W. != RE2.Anzahl 4-W
   Gib FALSCH zurück
Für alle  1 < i < RE1.Anzahl 3-W
  | Falls  |RE1.Bindungswinkel[i] – RE2.Bindungswinkel[i]| > δ1
  |        Gib FALSCH zurück
Für alle  1 < i < RE1.Anzahl 4-W
  | Falls|RE1.Tetraederwinkel[i] – RE2.Tetraederwinkel[i]| > 142δ2
  |        Gib FALSCH zurück
Gib WAHR zurück
```

6.3.2.2.3. Erweiterungen

Als Erweiterung dieser Definition kann man zusätzlich, wie im ersten Ansatz, die Bindungslängen mit in die Bewertung der Atommenge einbeziehen. Diese sollten dann jedoch mit relativ großen zulässigen Toleranzwerten verglichen werden.

Eine andere Möglichkeit ist, die zulässigen Winkeltoleranzen abhängig von der Entfernung der beiden Knoten zu sehen, so daß beispielsweise weiter entfernte Knoten weniger strikten Restriktionen unterliegen als dicht zusammenliegende. Hintergrund ist die Vorstellung, daß die Intramolekularen Wechselwirkungen bei weiter entfernten Atomen stark nachlassen.

Da durch die bloße Vorgabe eines zulässigen Winkelwertes eine starre Grenze gezogen wird und dadurch unter Umständen Knoten, die bezüglich des Referenzatoms lediglich eine Winkeldifferenz von nur einem Grad besitzen, in unterschiedliche Klassen fallen, ist es denkbar, eine „weichere" Grenze für die Winkelrestriktionen zu finden. Mit eben diesem Problem befaßt sich seit einiger Zeit die Theorie der *Fuzzy - Logic*. Dort versucht man strikte Übergänge, wie sie beispielsweise in diesem Fall geschaffen werden, durch fließende Übergänge zu ersetzen. Die nachfolgende Abbildung zeigt dabei eine mögliche Alternative zur strikten Winkeleinteilung:

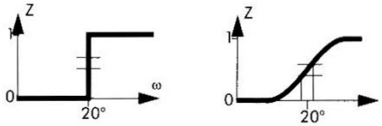


Dieser Idee liegt die Vorstellung zugrunde, daß Winkel von 19° und 21° im Grunde nahe beieinander liegen, jedoch Winkel von 16° und 24° bereits als sehr unterschiedlich klassifiziert werden. Um diesen - auch im Sprachgebrauch - fließenden Übergang zu beschreiben, benutzt man das zweite oben im Bild gezeigte Modell.

Auf der Y - Achse ist in diesem Fall die Zugehörigkeit zu einer bestimmten Klasse aufgetragen. Betrachtet man nun zwei Winkel ω_1 und ω_2, die im Rahmen der Methode *rel* ermittelt wurden (Bindungs- oder auch Tetraederwinkel) so sollen die dazugehörigen Atome nur dann in unterschiedliche Klassen eingeteilt werden, wenn sich die Zugehörigkeitswerte $Z(\omega_1)$ und $Z(\omega_2)$ um mehr als 0,2 unterscheiden.

Im ersten Bild werden dadurch Atome mit Winkeln von 19° und 21° getrennt. Im zweiten Bild dagegen

nicht:



Da man jedoch nur zwei Klassen zur Verfügung hat, auf die diese Knoten aufgeteilt werden können verwendet man als Referenz die geringere der beiden Gradzahlen. Bei einem diskreten Vergleich zwischen 19° und 21° Grad sind beide Atominstanzen in der Klasse für die Atominstanzen mit weniger als 20° Abstand. Wird der Knoten, dessen Bindungswinkel 21° beträgt, jedoch mit einem Knoten verglichen, dessen zugeordneter Winkel 15° ist, wird er für die Klasse mit der höheren Gradzahl klassifiziert.

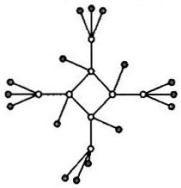
Man bemerke, daß dieser Ansatz nicht identisch dazu ist, gleich alle Atominstanzen mit weniger als 22° der gleichen Klasse zuzuordnen.

6.3.2.3. Methode Γ

Die Methode Γ - Funktion zum Testen globaler Eigenschaften der aktuellen Klassifizierung wird bei dieser Diskretisierung nicht unbedingt benötigt. Eine mögliche Implementation ist jedoch nachfolgend bei der Betrachtung des Fullerens näher ausgeführt.

6.3.2.4. Beispiel

In diesem Beispiel betrachten wir Tetramethylcyclobutan (C_8H_{16}):

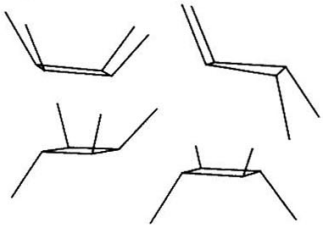


Es besitzt vier Stereoisomere und nach einer Klassifizierung der 500 dazu berechneten 3D - Plazierungen sollten deshalb auch mindestens vier Klassen erkannt werden.

Das Ergebnis der Klassifizierung in der bereits vorgestellten Tabellenform lautet:

#	Energie	Treffer	Min	Max	Schnitt
1	159.791	484	159.333	162.815	159.704
2	160.587	212	159.629	160.883	160.339
3	163.5	175	162.115	165.993	162.619
4	157.579	97	156.887	158.234	157.297
5	267.196	4	260.175	289.126	265.584
6	275.251	2	262.367	275.251	268.809
7	259.146	2	250.541	259.146	254.843
8	249.553	2	249.553	267.362	258.457

Die vier Strukturen mit den meisten Treffern sind:



Durch diese Beschreibung wurden also alle Stereoisomere korrekt erkannt. Nimmt man zur Klassifizierung die Wasserstoffe mit hinzu, so ergibt sich folgende Tabelle:

#	Energie	Treffer	Min	Max	Schnitt
1	159.791	484	159.333	162.815	159.704
2	160.587	212	159.629	160.883	160.339
3	163.5	175	162.115	165.993	162.619
4	157.579	97	156.887	158.234	157.297

Die Basiselemente mit nur zwei oder vier Treffern wurden durch die Berücksichtigung der Wasserstoffe entfernt, d.h. die Differenzen, die durch die zusätzliche Berücksichtigung der Wasserstoffe entstanden sind, waren zu groß. Tatsächlich ist es jedoch so, daß alle zu den vier Stereoisomeren ähnlichen Strukturen auch mit explizit verwendeten Wasserstoffen weiterhin dazu ähnlich sind.

Ein weiteres Molekül, das in letzter Zeit sehr oft betrachtet wurde dient als nächstes Beispiel. Links die zweidimensionale Plazierung eines Fulleren (C_{60}), bei der die Doppelbindungen sichtbar sind und rechts die dreidimensionale, optimierte Form:

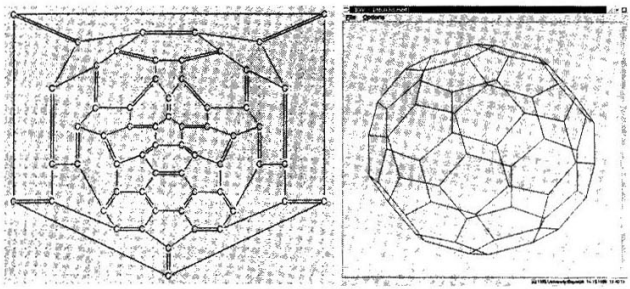


Abbildung 26: *Fulleren C_{60}*

Die Anzahl der Automorphismen des Fulleren ist bei der zuvor eingesetzten Diskretisierung 120. In diesem Fall wird jedoch auch erlaubt, das Innere des „Fußballes" nach Außen zu kehren, also eine Inversion der Struktur vorzunehmen. Soll die Orientierung der Knoten erhalten bleiben, muß eine

Erweiterung des Algorithmus vorgenommen werden. In der Methode Γ wird bei jedem neu hinzugefügten Atom die Orientierung bezüglich der bereits festgelegten Atome berechnet. Eine mögliche Implementation der entsprechenden Schnittstellen lautet:

Zuerst die Datenstruktur für das Ergebnis der Methode Γ :

```
GE (gamma - Ergebnis)
Daten:
   Orientierung (Positiv / Negativ)
Funktionen:
   Operationen: „="
```

Die Definition des „=" - Operators ist dann klar. Die Methode Γ selbst zur Ermittlung der Orientierung lautet dann:

```
Molekül ::Γ(σ)
Falls|σ| < 3
    GE.Orientierung = Positiv
max =|σ|
Falls det( Atom[1].Koord.,Atom[2].Koord.,
                                Atom[max] )>0
    GE.Orientierung = Positiv
sonst
    GE.Orientierung = Negativ
Gib GE zurück
```

6.3.3. Auswertungen

Dieses Verfahren wurde anhand verschiedener Vorgaben getestet. Einige Beispiele sind nachfolgend aufgeführt. Weitere Einsatzgebiet, die auch teilweise bereits realisiert wurden findet man noch in den Kapiteln 6.4. bis 6.9..

6.3.3.1. Cambridge Crystallographic Database

Um die Güte des Verfahrens beurteilen zu können, wurde ein Satz von 278 Beispielstrukturen aus der Cambridge Crystallographic Database für spezielle Auswertungen herangezogen. Die dort gespeicherten Moleküle besitzen eine experimentell ermittelte 3D - Referenzplazierung. Die Moleküle decken keine spezielle Substanzklasse ab und besitzen auch unterschiedliche Größen.

Einen kleinen Überblick bietet folgende Abbildung aus dem Programm 3DV.

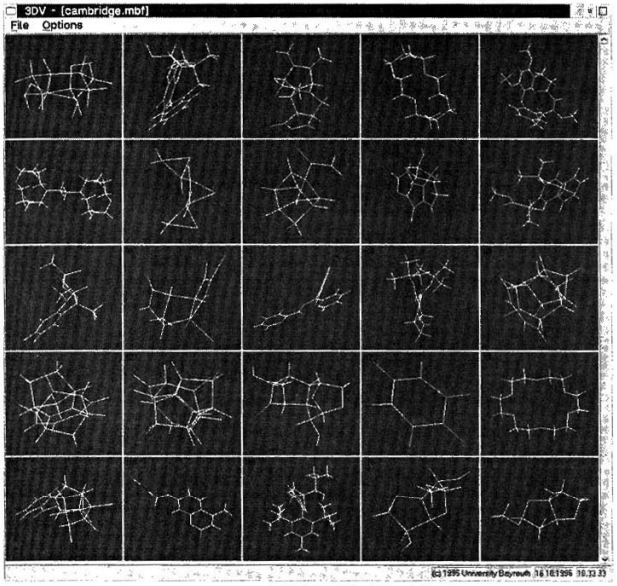


Abbildung 27: *Beispiele aus der Cambridge Datenbank*

Jede der 278 Strukturen wurde 500 mal mit Hilfe der MM2 - Methode 3D - plaziert. Die Auswertung ist also auch abhängig von der Qualität dieser Methode und des Optimierungsverfahrens selbst.

Eine Fragestellung war, wieviele der optimierten Strukturen ähnlich zu der jeweiligen Referenzstruktur sind.

Von den insgesamt 278 Strukturen wurde unter den jeweils 500 optimierten Strukturen in 141 Fällen, also bei gut 50% die Referenzstruktur *nicht* gefunden. Berücksichtigt man explizit die Wasserstoffe, dann ergibt sich sogar eine Quote von 69%. In beiden Fällen wurden sowohl bei den Bindungswinkeln wie auch bei den Tetraederwinkeln Toleranzen von 25° zugelassen. Ändert man diese Vorgaben beispielsweise auf 25° bei den Bindungswinkeln und 35° bei den Tetraederwinkeln, so ergeben sich Fehlerquoten von 39% bzw. 59%.

Eine weitere Fragestellung befaßt sich mit der Klassifizierung der Gesamtmenge der jeweils 500 optimierten Strukturen.

Mit Hilfe des Basisalgorithmus aus 6.2.2.3.1. wird dazu eine Menge von Basisstrukturen ermittelt. Dabei ist von Interesse, wieviele wesentlich unterschiedliche Basisstrukturen existieren und wieviele Moleküle dazu ähnlich sind. Idealerweise sollten wenigstens so viele Klassen erzeugt werden wie Stereoisomere zu der Referenzstruktur existieren. Da jedoch einige Strukturen bereits mehr als 1000 Stereoisomere aufweisen (siehe Tabelle), ist dies bei maximal 500 Optimierungsläufen natürlich nicht zu realisieren. Zusätzliche Klassen können dann noch durch unterschiedliche Konformationen der Stereoisomere entstehen oder durch fehlerhafte Plazierungen.

Die Basisberechnungen wurde sowohl unter expliziter Verwendung der Wasserstoff als auch ohne durchgeführt.

Als Beispiel betrachten wir die folgende Struktur:

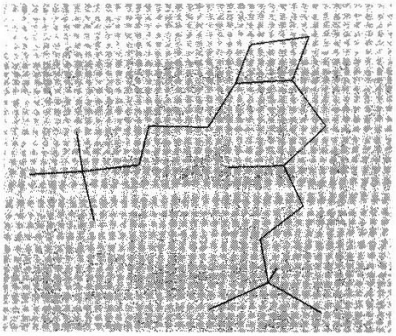


Abbildung 28: *Beispielstruktur cam_184 CCD*

Die Ergebnisse der Basisberechnungen ergeben folgende Tabellen: (linke Seite ohne, rechte mit Berücksichtigung der Wasserstoffe)

cam_184.mbf:

	Tests:	18272	Time:	96.339			Tests:	19764	Time:	216.151	
	SHits:	75	Stereo:	16			SHits:	83	Stereo:	16	
#	Energie	Treffer	Min	Max	Schnitt		Energie	Treffer	Min	Max	Schnitt
1	-1	1	1e+14	-1	-1		-1	1	1e+14	-1	-1
2	302.758	58	294.926	314.472	301.356		302.758	58	294.926	314.472	301.356
3	304.592	31	299.438	315.198	310.159		304.592	28	299.438	315.198	310.156
4	305.361	26	293.505	309.906	303.271		305.361	26	293.505	309.906	303.271
5	301.109	23	299.273	322.838	303.749		313.839	22	297.223	318.641	305.365
6	313.839	22	297.223	318.641	305.365		301.109	16	299.273	322.838	303.604
7	314.966	15	296.416	314.966	304.388		314.966	15	296.416	314.966	304.388
8	299.555	15	292.985	305.308	300.555		299.555	14	292.985	305.308	300.814
9	301.624	13	297.498	317.865	302.435		301.624	13	297.498	317.865	302.435
10	302.395	12	297.545	318.597	306.48		305.918	12	297.801	313.8	301.114
11	305.918	12	297.801	313.8	301.114		302.395	12	297.545	318.597	306.48
12	294.721	10	293.578	309.196	301.396		294.721	10	293.578	309.196	301.396
13	306.441	9	299.389	321.367	314.18		314.195	8	306.613	323.098	313.17
14	314.195	8	306.613	323.098	313.17		298.122	8	295.717	301.505	299.503
15	298.122	8	295.717	301.505	299.503		306.441	8	299.389	310.063	306.993
16	294.828	7	294.828	313.97	306.345		294.828	7	294.828	313.97	306.345
17	296.76	5	296.76	309.062	306.558		307.477	5	302.225	314.045	305.626
18	305.139	5	298.797	318.24	305.013		296.76	5	296.76	309.062	306.558

cam_184.mbf:											
19	296.041	5	296.041	313.877	299.676		296.041	5	296.041	313.877	299.676
20	311.579	5	304.704	311.579	307.517		301.433	5	301.433	307.819	304.785
21	307.477	5	302.225	314.045	305.626		311.579	5	304.704	311.579	307.517
22	321.894	5	300.893	321.894	311.697		311.212	5	296.201	319.001	312.364
23	300.232	5	297.32	322.235	306.107		321.894	5	300.893	321.894	311.697
24	305.453	4	302.152	305.453	303.326		307.9	5	302.04	307.9	304.435
25	918.224	4	907.781	918.224	911.52		305.139	5	298.797	318.24	305.013
26	325.207	4	304.79	325.207	315.622		918.224	4	907.781	918.224	911.52
27	302.662	4	297.582	315.155	308.291		305.743	4	301.819	311.599	307.305
28	304.285	4	300.054	304.285	301.753		305.453	4	302.152	305.453	303.326
29	315.004	4	297.585	319.883	312.635		300.232	4	297.32	322.235	304.395
30	291.027	4	291.027	297.767	292.24		302.662	4	297.582	315.155	308.291
31	310.311	4	298.79	310.311	302.385		302.48	4	292.967	302.48	297.256
32	301.433	4	301.433	306.408	304.042		304.048	4	295.952	310.171	301.676
33	302.48	4	292.967	302.48	297.256		315.004	4	297.585	319.883	312.635
34	304.048	4	295.952	310.171	301.676		304.285	4	300.054	304.285	301.753
35	311.212	4	296.201	314.248	308.104		312.825	4	299.117	312.825	301.948
36	310.472	4	300.213	312.243	308.794		310.472	4	300.213	312.243	308.794
37	312.825	4	299.117	312.825	301.948		291.027	4	291.027	297.767	292.24
38	308.174	3	302.625	308.174	304.135		314.009	3	314.009	317.28	314.96
39	310.083	3	310.083	319.651	317.006		319.849	3	305.406	321.948	317.288
40	297.162	3	297.162	314.279	308.453		308.174	3	302.625	308.174	304.135
41	307.9	3	302.04	307.9	304.811		310.083	3	310.083	319.651	317.006
42	308.357	3	305.063	314.105	310.407		316.601	3	304.137	317.393	310.567
43	300.224	3	291.955	300.224	294.771		300.224	3	291.955	300.224	294.771
44	314.009	3	314.009	317.28	314.96		308.357	3	305.063	314.105	310.407
45	946.685	3	946.685	1003.85	978.04		297.162	3	297.162	314.279	308.453
	DHits:	21					DHits:	22			

SHits bezeichnet die Menge der einelementigen Strukturklassen, DHits die Strukturklassen, die zwei Moleküle enthalten. Die Anzahl der Stereoisomere ist in diesem Fall 16. Der erste Energiewert entspricht der ersten, in die Strukturklasse aufgenommenen Struktur. Die übrigen Werte zeigen Minimum, Maximum und die durchschnittliche Energie an. Die Sortierung erfolgt nach der Anzahl der auf eine Basisklasse entfallenen Treffer.

Die Ergebnisse der übrigen Strukturen sind jederzeit verfügbar, sind jedoch zu umfangreich, um sie hier aufzuführen.

6.3.3.2. Überlegungen zur Transitivität

Wie bereits eingangs erwähnt, ist die Ähnlichkeit von Molekülen keine Äquivalenzrelation. Um dies am Beispiel zu verdeutlichen, verändern wir den Basisalgorithmus aus dem vorhergehenden Abschnitt.

Im vorhandenen Algorithmus wird eine Struktur als ähnlich zu einer Basisstruktur bezeichnet, wenn sie ähnlich zum ersten Vertreter aus dieser Klasse ist.

Aufgrund der nicht vorhandenen Transitivität ist es jedoch notwendig, die Ähnlichkeit bezüglich aller Strukturen zu überprüfen, die zu dieser Basisstruktur bis dahin als ähnliche erkannt worden sind.

Der modifizierte Basisalgorithmus lautet dann wie folgt:

Modifizierte Basisberechnung:

1. Nimm die erste Struktur in die Basis auf. Die Liste der dazu ähnlichen Strukturen ist leer.
2. Laufe durch alle vorhandenen Strukturen
3. Laufe durch alle Basisstrukturen
4. Ist die Struktur ähnlich zur betrachteten Basisstruktur und der Liste aller zu dieser ähnlichen Strukturen, nimm die Struktur in die Liste der dieser Basisstruktur ähnlichen Strukturen auf.
5. Wurde die Struktur noch nicht in eine Liste der zu einer Basisstruktur ähnlichen Strukturen aufgenommen, nimm die Struktur in die Basis auf. Die Liste der dazu ähnlichen Strukturen ist leer.

Um dies am Beispiel zu verdeutlichen, betrachten wir folgendes Molekül:

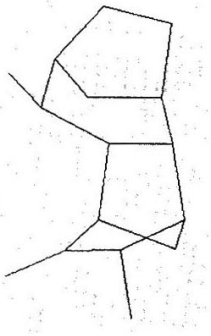


Abbildung 29: *Beispiel 1 für Transitivitätsproblem*

Die nachfolgende Tabelle beinhaltet im linken Teil das Ergebnis des bisherigen Basisalgorithmus und im rechten Teil das Ergebnis der Berechnung mit dem modifizierten Algorithmus:

cam_103.mbf:

	Time	35		Tests	17062
	SHits:	94			
#	Energie	Treffer	Min	Max	Schnitt
1	-1	13	207.4	211.869	208.684
2	221.77	29	219.22	223.925	222.127
3	208.408	29	207.279	221.952	207.998
4	210.4	26	208.849	218.502	210.113
5	218.618	21	217.564	317.413	230.345
6	212.094	21	209.829	308.59	212.069
7	209.184	20	208.407	209.89	208.964

Time	51		Tests	19149
SHits:	97			
Energie	Treffer	Min	Max	Schnitt
-1	2	209.877	209.877	209.877
221.77	26	219.22	223.925	222.19
208.408	26	207.279	221.952	208.023
210.4	26	208.849	218.502	210.113
212.094	21	209.829	308.59	212.069
218.618	20	217.564	220.118	218.084
209.184	20	208.407	209.89	208.964

cam_103.mbf:

8	209.12	19	206.13	213.899	208.934
9	207.329	16	206.238	213.751	207.24
10	211.886	14	209.333	212.951	211.943
11	207.074	14	206.937	212.498	207.339
12	215.949	13	214.361	216.273	215.445
13	214.086	13	211.624	304.764	213.142
14	208.61	13	207.784	214.668	209.266
15	218.591	13	217.317	321.492	244.083
16	216.962	11	216.228	217.951	217.25
17	208	11	207.449	211.661	207.96
18	224.419	9	223.928	225.326	224.856
19	223.702	8	223.518	230.272	223.94
20	226.786	8	225.138	227.462	225.955
21	227.046	7	226.985	228.369	227.948
22	224.627	7	224.01	225.464	224.966
23	227.522	7	227.445	229.059	228.302
24	225.038	6	224.266	225.73	225.007
25	223.87	5	223.155	224.577	223.81
26	226.523	5	226.488	226.745	226.638
27	223.91	4	223.516	223.91	223.603
28	224.784	4	224.784	225.449	225.21
29	225.59	4	225.59	225.891	225.815
30	338.237	3	337.336	341.711	339.749
31	337.359	3	334.186	339.462	337.617
32	315.365	3	312.733	315.365	313.809
33	412.94	3	316.493	412.94	340.683
34	683.092	3	681.752	683.092	682.284
35					

209.12	19	206.13	213.899	208.934
207.329	16	206.238	213.751	207.24
211.886	14	209.333	212.951	211.943
207.074	14	206.937	212.498	207.339
208.61	13	207.784	214.668	209.266
218.591	13	217.317	321.492	244.083
214.086	12	211.624	214.402	212.784
216.962	11	216.228	217.951	217.25
208	11	207.449	211.661	207.96
208.18	11	207.4	211.869	208.683
224.419	9	223.928	225.326	224.856
223.702	8	223.518	230.272	223.94
226.786	8	225.138	227.462	225.955
215.949	7	214.361	215.949	214.724
227.046	7	226.985	228.369	227.948
224.627	7	224.01	225.464	224.966
227.522	7	227.445	229.059	228.302
225.038	6	224.266	225.73	225.007
216.192	6	215.311	216.273	215.532
223.87	5	223.155	224.577	223.81
226.523	5	226.488	226.745	226.638
223.91	4	223.516	223.91	223.603
224.784	4	224.784	225.449	225.21
225.59	4	225.59	225.891	225.815
338.237	3	337.336	341.711	339.749
337.359	3	334.186	339.462	337.617
315.365	3	312.733	315.365	313.809
412.94	3	316.493	412.94	340.683

Wie man sieht, verändert sich die Basis. Neue Strukturen werden zu Basisstrukturen und die Anzahl der Treffer auf bereits vorhandene Strukturen verringert sich.

Basisstrukturen mit weniger als drei Treffern wurden aus Gründen der Übersicht weggelassen.

Anschaulich kann man das dadurch erklären, daß beim ersten Algorithmus alle Strukturen in die Basis aufgenommen werden, deren „Differenz" zur Referenzstruktur am Rande des Zulässigkeitsbereichs liegt. Zwei derartige Strukturen sind jedoch nicht zwangsläufig ähnlich zueinander. Der neue Basisalgorithmus überprüft dies und erkennt deshalb gegebenenfalls eine Struktur nicht als zu einer bestimmten Basis gehörig an.

Für einen ersten Überblick über die mindestens vorhandene Anzahl von Strukturklassen ist der erste Basisalgorithmus durchaus verwendbar. Für eine exakte Bestimmung aller Strukturklassen ist es jedoch unumgänglich, den obigen Basisalgorithmus anzuwenden.

Betrachten wir noch eine weitere Struktur zur Verdeutlichung des Transitivitätsproblems:

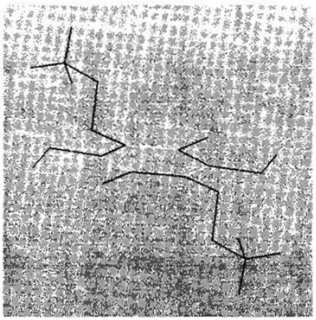


Abbildung 30: *Beispiel 2 für Transitivitätsproblem*

Wiederum enthält die Tabelle im linken Teil das Ergebnis des bisherigen Basisalgorithmus und im rechten Teil das Ergebnis der Berechnung mit dem modifizierten Algorithmus:

cam_230.mbf:

	Tests:	5979	Time:	31.516	
	SHits:	28	Stereo:	1	
#	Energie	Treffer	Min	Max	Schnitt
1	-1	1	1e+14	-1	-1
2	175.231	79	170.346	177.79	173.001
3	173.098	60	171.099	181.458	173.476
4	182.531	47	174.661	184.056	180.026
5	171.554	45	169.432	177.866	173.234
6	172.807	42	171.482	180.759	174.375
7	183.095	31	174.224	185.457	179.056
8	173.438	29	170.428	176.715	171.561
9	171.387	19	169.537	175.79	172.714
10	173.302	18	171.709	182.228	179.836
11	174.233	16	170.778	178.448	176.348
12	176.939	16	171.819	178.372	176.594
13	173.511	13	172.083	180.583	174.62
14	173.875	11	169.523	177.88	171.222
15	171.44	9	171.44	177.805	172.787
16	176.994	8	176.051	183.022	180.185
17	176.191	4	176.191	177.803	177.189
18	173.859	4	173.859	177.921	176.845
19	778.446	3	775.09	778.446	776.163
20	174.286	3	174.286	177.636	175.976
21	168.397	3	168.397	174.224	171.913

Tests:	8960	Time:	121.295	
SHits:	29	Stereo:	0	
Energie	Treffer	Min	Max	Schnitt
-1	1	1e+14	-1	-1
175.231	70	170.507	177.79	173.001
173.098	57	171.099	181.458	175.157
171.554	44	169.432	177.866	173.234
182.531	41	174.661	183.542	179.7
173.438	25	170.428	175.08	171.57
183.095	15	174.224	185.457	179.803
172.807	15	171.482	179.351	175.297
174.286	14	174.286	180.624	177.1
173.281	12	173.075	178.448	176.46
173.875	12	169.523	175.79	171.094
171.44	12	171.44	180.759	172.939
168.397	11	168.397	180.172	173.04
171.387	11	169.537	174.021	172.993
173.302	11	173.302	182.228	179.912
176.939	9	173.968	176.939	174.937
174.233	9	170.778	175.175	172.137
180.061	8	174.231	180.061	176.527
173.511	8	172.083	176.966	176.654
176.994	8	176.051	183.022	180.185
177.408	8	176.256	181.816	178.389

cam_230.mbf:

22	181.088	3	179.976	183.347	181.939
23	172.303	3	169.752	172.943	171.985
24	177.408	3	177.408	181.816	180.571
25	800.189	2	800.189	802.352	801.271
26	174.702	2	174.702	174.777	174.74
27					
28					
29					
30					
31					
32					
33					
34					
35					
36					
37					
38					
39					
40					
41					
42					
43					

172.38	6	170.346	173.373	172.307
174.034	6	171.538	174.812	173.207
173.505	5	172.165	173.992	173.216
180.88	5	178.046	184.056	182.648
172.303	4	169.752	173.148	171.255
176.466	4	175.095	176.979	176.293
181.088	4	173.708	183.347	180.233
174.797	4	173.088	180.583	174.92
778.446	3	775.09	778.446	776.163
174.702	3	174.702	176.715	175.243
170.465	3	170.465	172.837	171.817
171.747	3	170.787	171.747	171.258
176.59	3	172.048	176.59	173.333
172.88	3	171.654	173.467	172.867
800.189	2	800.189	802.352	801.271
174.224	2	172.516	174.224	173.37
175.918	2	175.918	178.372	177.145
171.73	2	171.73	171.75	171.74
173.167	2	173.167	173.639	173.403
179.146	2	179.146	182.023	180.584
173.811	2	173.811	176.039	174.925
173.859	2	173.859	175.215	174.537

Wie man sieht, werden die einzelnen Basisklassen noch wesentlich deutlicher in neue Klassen aufgeteilt, als im vorherigen Beispiel.

6.3.4. Kanonische Numerierung

Die Schnittstelle zur kanonischen Numerierung dreidimensionaler Moleküle ist wie auch bei der topologischen kanonischen Numerierung identisch zu der Schnittstelle für die Vollstrukturisomorphie.

Die Methode *kl*, die im topologischen Fall eine Abbildung auf der Adjazenzmatrix ist, muß um dreidimensionale Eigenschaften erweitert werden.

Dazu eignen sich wiederum alle bereits bei der Methode *rel* berechneten Werte. Sei KE (*kl* - Ergebnis) wie bei der topologischen kanonischen Numerierung definiert. Die Operatorfunktion auf dem Ergebnis

der Klassifikationsfunktion ist dann definiert durch:

```
operator KE (KE1) > KE (KE2)

MIN = min (|KE1.Numerierung|,|KE2.Numerierung|)
Falls Min>1
   Für 1 <= y <MIN
      Für 0<= x <= y
         A1 = KE1.Numerierung[x], B1 = KE1.Numerierung[y]
         A2 = KE2.Numerierung[x], B2 = KE2.Numerierung[y]
         Falls KE1.Strukturdaten.Conn( A1, B1 ) >
               KE2.Strukturdaten.Conn( A2, B2 )
                  Gib WAHR zurück
         Falls KE1.Strukturdaten.Conn( A1, B1 ) <
               KE2.Strukturdaten.Conn( A2, B2 )
                  Gib FALSCH zurück
         Falls KE1.Strukturdaten.3-Wege( A1, B1 ) >
               KE2.Strukturdaten.3-Wege( A2, B2 )
                  Gib WAHR zurück
         Falls KE1.Strukturdaten.3-Wege( A1, B1 ) <
               KE2.Strukturdaten.3-Wege( A2, B2 )
                  Gib FALSCH zurück
         Für alle 1<= i <= KE1.Strukturdaten.3-Wege( A1, B1 )
            Falls |KE1.Strukturdaten.Bindungswinkel( A1,B1 )[i] -
                  KE2.Strukturdaten.Bindungswinkel( A2,B2 )[i]|>25
                     Falls KE1.Strukturdaten.Bindungswinkel( A1,B1 )[i] >
                           KE2.Strukturdaten.Bindungswinkel( A2,B2 )[i]
                              Gib WAHR zurück
                     Sonst
                              Gib FALSCH zurück

         Analog für Tetraederwinkel

Falls |KE1.Numerierung| = |KE2.Numerierung|
   Gib FALSCH zurück
Gib MÖGLICH zurück
```

Ein einfaches Beispiel soll die Funktionsweise des Verfahrens beschreiben. Als Diskretisierung wurde der zweite zuvor beschriebene Ansatz verwendet:

6.3.4.1. Beispiel

Betrachten wir nochmals das Tetramethylcyclobuthan aus Abschnitt 6.3.2.4.. Die Atome von vier belie-

bigen Referenzstrukturen seien wie folgt numeriert:

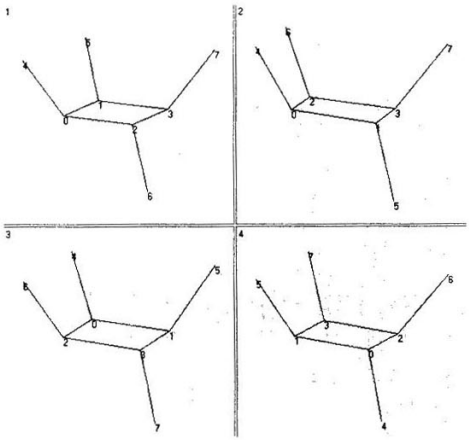


Abbildung 31: *Zwei Numerierungen von Tetramethylcyclobuthan*

Nach dem Durchlaufen des Algorithmus zur kanonischen Numerierung ergeben sich die folgenden Numerierungen:

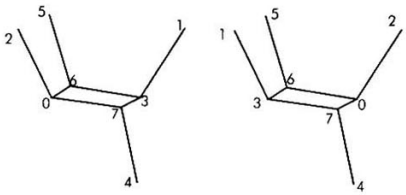


Abbildung 32: *Kanonische Numerierungen des Tetramethylcyclobuthan*

Die erste kanonische Numerierung wird bei Struktur 1 und 4 berechnet, die zweite bei Struktur 2 und 3.

Wie man sieht stimmen die beiden kanonischen Numerierungen nicht überein. Dies liegt daran, daß die verwendete Diskretisierung die Spiegelung, durch welche die beiden Numerierungen ineinander überführt werden können, nicht unterscheidet.

Die auf diese Art kanonisch numerierten Strukturen können nun mit dem Algorithmus aus Kapitel 4.2.5. sehr einfach auf Ähnlichkeit getestet werden.

Ändert man die zugrundeliegende Diskretisierung, so müssen auch die kanonischen Numerierungen neu berechnet werden, um eine Übereinstimmung gewährleisten zu können.

Die kanonische Numerierung ist in diesem Fall auch äußerst nützlich für eine RMS - Berechnung der beiden Strukturen, da sie bei der vorliegenden unterschiedlichen räumlichen Ausrichtung unter Verwendung der eindeutigen kanonischen Numerierung sehr einfach so ineinander gedreht werden können, daß ein minimaler Abstand (RMS) zwischen den Molekülen berechnet werden kann.

6.3.5. Substruktursuche

Die dreidimensionale Substruktursuche kann - analog zur topologischen Substruktursuche - mit den Schnittstellen der Vollstrukturisomorphie durchgeführt werden. Einziger Unterschied ist jedoch auch hier die spezielle Berücksichtigung der Konnektivitäten. Es können ebenso offene, geschlossene oder Ringsysteme gesucht werden.

6.3.5.1. Erweiterte Strategien

Ein Problem, das speziell bei der dreidimensionalen Substruktursuche auftaucht ist, daß eine frei plazierte Struktur in der Regel eine andere räumliche Gestalt annimmt als diejenige, die als Bestandteil eines größeren Moleküls den Wechselwirkungen mit den zusätzlichen Nachbaratomen ausgesetzt ist.

Da man jedoch gerade bei der Substruktursuche nicht weiß, welche benachbarten Atome vorhanden und damit welchen zusätzlichen Energien die Substruktur ausgesetzt ist, muß man die vorgegebenen Winkelrestriktionen entsprechend flexibel gestalten oder folgendes Verfahren anwenden:

- Führe eine 2D - Substruktursuche durch.
- Für jede Einbettung
- Verwirf die nicht beteiligten Atome
- Führe eine Nachoptimierung, ausgehend von den aktuellen Koordinaten, durch
- Führe den Algorithmus zur Isomorphieberechnung nach kanonischer Numerierung mit Original und optimierter Einbettung aus.

Damit kann eine erheblich bessere Trefferquote erzielt werden. Durch die zusätzliche Optimierung ist jedoch der Zeitaufwand für dieses Verfahren etwas größer.

Bei der Durchführung der Nachoptimierung ist darauf zu achten, daß die Konformation und Konstitution der entsprechenden Teilstruktur erhalten bleiben.

6.4. Einsatz bei der Clusteranalyse/Datenbanken

Die Klassifizierung von Molekülen nach unterschiedlichen Gesichtspunkten ist heute von großer Bedeutung in der chemischen Industrie.

Sind diese Merkmale mit Hilfe eines diskreten Modells beschreibbar, wie beispielsweise das Vorhandensein bestimmter Teilstrukturen, so eignen sich die vorgestellten Algorithmen in besonderem Maße für die Aufgaben deren Erkennung.

K. Varmuza hat bereits ein Programm zur Clusteranalyse vorgestellt [KV1/KV2]. In Anlehnung an die dort zur Clusterung herangezogenen Eigenschaften wurde unter anderem mit Hilfe der vorgestellten Algorithmen ein Programm erstellt, das diese Eigenschaften einer Struktur erkennt und sie in einem zur Speicherung in einer Datenbank geeigneten Format weitergibt.

Unter der Leitung von R. Laue ist in Bayreuth ein Datenbanksystem entstanden, das die unterschiedlichen Strukturmerkmale flexibel handhaben kann. Dies wird durch eine objektorientierte Struktur ermöglicht, die es erlaubt, auf diese Merkmale mit Hilfe der dafür am besten geeigneten Zugriffswege zuzugreifen. Beispielsweise werden binäre Informationen über einen sog. Superimposed Coding - Zugriffsweg verwaltet, wohingegen Informationen, die über Bereichssuchen angefordert werden, mit einem Multi-Index Verfahren gespeichert und abgefragt werden. Das folgende Diagramm zeigt eine vereinfachte Übersicht über die Struktur dieser modernen, objektorientierten Datenbank.

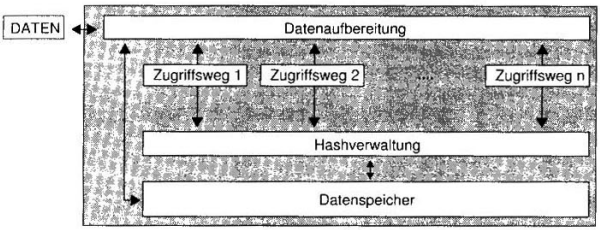


Abbildung 33: *Schematischer Aufbau der Datenbank*

Zur eigentlichen Clusteranalyse werden dann verschiedene chemische Strukturen anhand des Eigenschaftsvektors miteinander verglichen. Interessant ist auch die Betrachtung der Korrelationen zwischen verschiedenen Eigenschaften innerhalb eines Vektors einer Struktur.

Prinzipiell erfordert die Speicherung chemischer Strukturen in Datenbanken ein ähnliches Vorgehen wie bei anderen Daten. Ziel einer Datenbank ist neben der Sicherung und Aufbewahrung der Daten auch die Möglichkeit, diese bei Bedarf schnell wiederzufinden. Dazu verwendet man bei herkömmlichen Datentypen Indizes, welche eine geordnete Sicht auf die Daten - vergleichbar mit der Sortierung in einem Telefonbuch - erlauben. Sie erlauben direkten Zugriff auf Datenbereiche, die durch die Benutzeranfragen spezifiziert werden.

Analog werden Indizes auch bei chemischen Strukturen verwandt. Um beispielsweise eine Vollstruktursuche in einer Datenbank von mehreren Millionen Strukturen durchführen zu können, sollte man gewisse Merkmale vorgeben, die möglichst charakteristische Eigenschaften der Struktur beschreiben. So kann man beispielsweise sehr einfach anhand von sog. topologischen Indizes den Kreis der möglichen Kandidaten einschränken. Erst wenn diese primären Strukturmerkmale keine Unterscheidung mehr ermöglichen (beispielsweise ist der sog. Wiener - Index ([HW]) bei allen für eine Bruttoformel von MOLGEN generierten Strukturen identisch), sollte der Isomorphiealgorithmus eingesetzt werden.

Weitere Charakteristika bilden auch die eingangs erwähnten, bei der Strukturidentifizierung verwendeten Merkmale. Die Suche nach einer unbekannten Struktur im Reagenzglas ist hier bis zu einem gewis-

sen Punkt identisch mit der Suche nach einer Struktur in einer Datenbank oder, wenn man dieses Konzept erweitern möchte, identisch zur Konstruktion unbekannter Strukturen, welche vorgegebene Eigenschaften besitzen müssen. Bei letzterem handelt es sich um ein inverses Problem, das beispielsweise mit MOLGEN schon bis zu einem gewissen Grade gelöst werden kann, wo aber noch leistungsfähigere Ansätze in Arbeit befindlich sind [TG]. Diese Vorsortierung nach Strukturinvarianten bezeichnet man in der Chemie auch als Screening. Nach einem effizienten Screening bleiben ca. 1% der betrachteten Strukturen für eine intensivere Suche übrig. Diese müssen dann tatsächlich, vorzugsweise mit den vorgestellten Algorithmen, einzeln überprüft werden. Eine Kombination eines Invariantenvektors mit Hashing wurde beispielsweise bei [JG] vorgestellt.

Ein effizientes Screening enthält auch Invarianten, die nicht nur für die Vollstruktur, sondern auch Teilstruktursuche relevant sind. Dabei gibt es zwei mögliche Ansätze. Der erste, nicht so effiziente Ansatz schreibt eine feste Zahl von Teilstrukturen vor. Jede Struktur, die neu in die Datenbank eingespeichert wird, wird nun auf diese Teilstrukturen hin überprüft und entsprechende Indexinformationen werden gespeichert. Sucht man nun nach Strukturen in der Datenbank, die bestimmte Teilstrukturen enthalten, so wird die betrachtete Teilstruktur wiederum bezüglich dieses vorgegebenen Satzes von Teilstrukturen überprüft. Gesucht werden nun alle Strukturen, die mindestens die Anzahl der in der Ausgangsstruktur enthaltenen Anzahlen von Teilstrukturen enthält. Damit kann die Anzahl der tatsächlich zu testenden Strukturen bereits erheblich eingeschränkt werden.

Wesentlich aufwendiger, aber effektiver ist eine zweite Möglichkeit. Beim Einfügen einer Struktur in die Datenbank werden alle dort bereits vorhandenen Strukturen daraufhin überprüft, ob die neu hinzugekommene Struktur Teilstruktur von ihnen ist, oder umgekehrt. Ist keines von beiden der Fall, so wird die Struktur mit in den Indexteil aufgenommen. Bei einer Substruktursuche führt man zuerst eine Vollstruktursuche durch, was ja - wie gezeigt - effizienter möglich ist. Da ein Crossindex über alle Strukturen in der Datenbank vorhanden ist, kann man dann sofort alle Strukturen ermitteln, die wiederum diese - bereits in der Datenbank vorhandene - Struktur beinhalten. Der Aufwand für das Einfügen oder gar Löschen einer Struktur ist bei dieser Vorgehensweise natürlich enorm. In der Regel wird man dann so vorgehen, daß eine gewisse Anzahl von neu einzufügenden Strukturen angesammelt wird, bevor eine Neuorganisation des Index durchgeführt wird.

Beide Verfahren werden in der Praxis erfolgreich eingesetzt. Beide Verfahren ermöglichen zwar eine effiziente Reduktion der Kandidaten. Einen direkten Vergleich mit einem der vorgestellten Algorithmen können sie jedoch nicht vermeiden.

Weitere Informationen zu diesem Thema findet man beispielsweise in [JS].

6.5. Einsatz bei der Konformationsanalyse

Wie bereits im vorherigen Kapitel angesprochen, unterscheidet man prinzipiell in der Chemie zwischen Konstitutions-, Konfigurations- und Konformationsisomeren. Leider werden diese Begriffe jedoch nicht immer auf dieselbe Weise verwendet, so daß bei unterschiedlichen durchaus unterschiedliche Vorstellungen des Isomeriebegriffs finden.

Die Ermittlung der Konstitutionsisomere zu einer vorgegebenen Bruttoformel kann beispielsweise durch den bereits mehrfach angesprochenen Graphengenerator MOLGEN vorgenommen werden. Das Zusatzmodul MOLVIEW erlaubt darüber hinaus die Ermittlung aller Konfigurationsisomere, wobei nicht immer eine exakte graphische Repräsentation verfügbar ist. Mit Hilfe der Erkennung von Ähnlichkeit bei Molekülen, wie in Kapitel 6.1. vorgestellt, ist es nun möglich, bei einer geeigneten Anzahl von Kandidaten, mögliche Konformationsisomere zu finden.

De Clercq entwickelte das Programm SCA („Systematic Conformational Analysis") zur Konstruktion von Konformationen von Ringsystemen, die aus drei bis maximal sieben Atomen bestehen. (siehe dazu [DC1],[DC2],[DC3]) Als Eingabe dient eine beliebige Struktur, die diese Bedingungen erfüllt. Die

Ausgabe des Programmes sind mögliche Konformationsisomere dieser Struktur.

Der Vorteil des Programms ist eine schnelle Konstruktion schlüssiger 3D - Geometrien durch die Verbindung von einem regelbasierten System mit einer nachgeschalteten Energiebetrachtung, ohne daß dafür eine Kraftfeldoptimierung durchgeführt werden müßte. Als Einschränkung ist zweifellos die beschränkte Ringdimension zu sehen. Aufgrund dessen wurde nur ein Vergleichstest mit einer kleinen Struktur durchgeführt. Für Cyclohexan berechnete SCA 32 Konformationsisomere. Mit Hilfe des Basisalgorithmus aus 6.3.3.2. wurde dann eine Basis dieser 32 Konformationen ermittelt. Tatsächlich ergaben sich durch diese anschließende Berechnung die bekannte Sessel-, Wannen- und Twistorform des Cyclohexan als Basis für die restlichen Konformationsisomere. Alle übrigen Konformationen waren also aufgrund des vorgegebenen Modells dazu isomorph.

Durch den Einsatz des Algorithmus kann also eine effiziente Reduktion einer vorliegenden Strukturmenge dadurch erreicht werden, daß isomorphe Konformationsisomere identifiziert werden können, bzw. bei aus einem beliebigen Satz von dreidimensionalen Repräsentanten einer Konfiguration verschiedene Konformationen ermittelt werden können .

6.6. Bestimmung der Ähnlichkeit von Molekülen

Ein äußerst interessantes Problem, das auch für den folgenden Abschnitt von großer Bedeutung ist, ist die Frage nach einem Maß für die Ähnlichkeit von Molekülen. Kennt man beispielsweise ein Molekül, das nachweislich eine bestimmte Wirkung hervorruft, so beschäftigt man sich mit der Frage, ob eventuell ein ähnliches Molekül, das evtl. leichter (kostengünstiger) zu synthetisieren ist existiert, oder eine Struktur, die bei gleicher Dosis einen wesentlich größeren Wirkungsgrad besitzt.

Kann man die Ähnlichkeit von Molekülen durch ein diskretes Maß bestimmen, so ist es auch möglich, mit Hilfe der vorliegenden Algorithmen, ähnliche Strukturen zu suchen. Dies kann aufgrund bereits bestehender Datenbanken erfolgen, wie in 6.4. aber auch auf gänzlich neue Strukturen, wie sie beispielsweise mit MOLGEN konstruiert werden können, ausgedehnt werden. MOLGEN verhält sich dann quasi wie eine „virtuelle" Datenbank.

In der Literatur werden vielfältige Anstrengungen unternommen, die Ähnlichkeit von chemischen Strukturen zu definieren. Verwiesen sei beispielsweise hier auf [DB] .

6.7. Einsatz bei der Spektrenvorhersage

Bei der Spektrenvorhersage geht es darum, für eine vorgegebene molekulare Struktur, Aussagen über das von ihr zu erwartende Spektrum zu treffen.

In der klassischen Chemie verwendet man zur Bestimmung einer unbekannten chemischen Substanz mit experimentellen Hilfsmitteln die sogenannte Spektroskopie. Grob vereinfacht versetzt man die betrachtete Struktur durch verschiedene Verfahren in einen Zustand, daß sie Energie freigibt. Aufgrund der Intensität und der Lage im Energiespektrum kann man spezifische Aussagen über die betrachtete Struktur und ihre Bestandteile treffen. Es existieren verschiedene Formen der Spektroskopie:

6.7.1. Massenspektroskopie

Die Massenspektroskopie und -spektrometrie wird als Routinemethode zur Substanzermittlung verwendet. Zur weiteren Analyse werden sie oft mit anderen Methoden, beispielsweise der Gaschromatographie gekoppelt. Bei der Massenspektroskopie werden Ionen entsprechend ihrem Masse/Ladungsverhältnis aufgetrennt und mit unterschiedlichen Verfahren registriert. Anhand der Bruchstücke können Rückschlüsse auf die Ausgangsstruktur gezogen werden.

6.7.2. IR-Spektroskopie

Bei der Infrarotspektroskopie wird den Molekülen Energie in Form von IR - Strahlung zugeführt. Dies bewirkt eine Schwingung der Atome innerhalb des Moleküls, deren Energie im Infrarotbereich (760nm - 500 μm) gemessen werden kann. Man unterscheidet dabei Valenz- und Deformationsschwingungen. Die Infrarotspektroskopie wird zur qualitativen und quantitativen Analyse und Konstitutionsermittlung bei Naturstoffen und Zwischenprodukten verwendet. Sie wird bei der Betriebskontrolle von Gemischen und Reinchemikalien und allgemein bei der Routineanalyse eingesetzt.

Weitere Informationen drüber und über andere Methode findet man in [HMZ]. Diese Spektren dienen in der Regel als Grundlage zur Aufklärung unbekannter chemischer Strukturen. In einem nächsten Schritt werden die Informationen, die das Spektrum liefert genutzt, um daraus mögliche Kandidaten zu konstruieren, beispielsweise mit dem Strukturgenerator MOLGEN.

Zur Verifizierung der generierten Strukturen wird dann ein künstliches Spektrum generiert und mit dem Originalspektrum verglichen.

Den typischen Arbeitsablauf zeigt das folgende Diagramm:

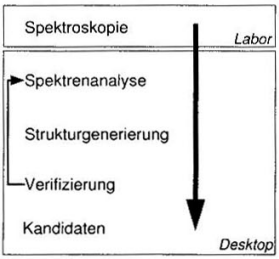


Abbildung 34: *Arbeitszyklus in der Strukturaufklärung*

Betrachtet man diesen Arbeitsablauf, wird die Intention der Industrie verständlich, in immer größeren Maße einen Weg zur automatische Strukturaufklärung zu suchen. Eine Stunde Arbeit im Spektroskopielabor ist heute mit wesentlich höherem Aufwand und Kosten verbunden als die Arbeit am Desktop. Dies setzt natürlich voraus, daß der Chemiker flexibel in der Auswahl seiner Hilfsmittel ist.

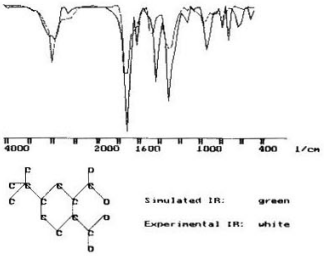
Zur Bestimmung der Spektren ist es notwendig, die Teilstrukturen in der Ergebnisstruktur zu bestimmen, von denen bekannt ist, daß sie spezifische Peaks im Spektrum hervorrufen. Weiteres Hilfsmittel ist dabei auch die Ermittlung sog. Retentionsindizes, die ebenfalls mit Hilfe der Substruktursuche in Molekülen ermittelt wird. Dies ist effektiv mit den vorgestellten Algorithmen möglich. Eine diesbezügliche Verwendung im laufenden Verbundprojekt mit der Universität Jena, gefördert von der DFG ist bereits vorbereitet worden.

Der Vergleich der berechneten Spektren liefert dann Aussagen über den Ähnlichkeitsgrad der berechneten Struktur mit der unbekannten Ausgangsstruktur.

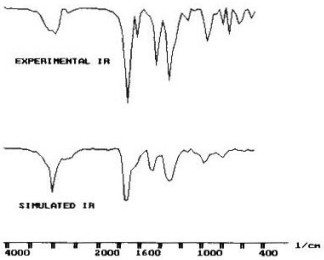
Ein Programm, das die Spektrenvorhersage für IR - Spektren realisiert, ist unter der Leitung von R. Herges an der Universität Erlangen entstanden.

SMIRRNA sucht dazu in der vorgegebenen Struktur nach einer festen Anzahl von bekannten Substrukturen. Dabei wird nicht nur das Auftreten einer Struktur, sondern auch die Häufigkeit des Auftretens berücksichtigt. Der sich daraus ergebende Substrukturvektor dient als Eingabe für ein neuronales Netz. Dieses wurde durch aufgeklärte Strukturen und die dazugehörigen tatsächlichen Spektren trainiert. Die Spektren wurden dafür in eine feste Anzahl von Intervallen über einen vorgegebenen Spektralbereich diskretisiert. Wird nun ein Substrukturvektor vorgegeben, so liefert das neuronale Netz als Ausgabe die

Intensitäten für die vorgegebenen Spektralbereiche, in denen das Netz trainiert wurde. Das folgende Bild ist eine Ausgabe des Programmes SMIRRNA. Das Spektrum zeigt überlagert das Original und das berechnete Spektrum der unteren Struktur.



Im nächsten Bild werden die beiden Spektren zur Übersicht getrennt gezeigt:



Ein komplettes Paket zur Strukturaufklärung, das nach diesem Konzept vorgeht, wird von der Firma Chemical Concepts (Weinheim) unter dem Namen SpecInfo vertrieben (siehe auch [SPM]). Bestandteil von SpecInfo ist auch MOLGEN, das in Bayreuth entwickelt wurde.

6.8. Dockingprobleme in der Pharmazie

Eine über die reine Strukturisomorphie bei Molekülen hinausgehende Anwendung der vorgestellten Algorithmen ist auf dem Gebiet der Pharmazie denkbar.

Ein Arzneimittel muß, um eine biologische Wirkung entfalten zu können, am Wirkungsort in Wechselwirkung mit einem biologischen Makromolekül treten. Diese Ligand - Rezeptor - Wechselwirkungen sind Grundlage aller Wirkungsanalysen in der modernen Arzneimittelforschung. Die Erforschung der Vorgänge, die diese Wechselwirkungen verursachen, sind wesentlich für die Entwicklung neuer, wirksamerer oder verträglicher Medikamente. Die moderne Gentechnologie und Molecular Modelling bieten hier Möglichkeiten, die noch vor wenigen Jahren undenkbar gewesen wären. Nach der Identifizierung der an einer biologischen Reaktion beteiligten Moleküle erfolgt die Ermittlung der Struktur der Stoffe, auch im Bezug auf ihre dreidimensionale Gestalt. Erst dann ist es möglich, synthetische Strukturen zu erstellen und gemäß der Zielsetzung zu verfeinern.

An dieser Stelle kann der Computer eingesetzt werden, um eine Auswahl oder Entwurf für passende

Liganden zu ermitteln.

Der Name Dockingproblem oder Schlüssel - Schloß - Prinzip beruht auf der Tatsache, daß nur ganz bestimmte Strukturen die gewünschten Wechselwirkungen hervorrufen. Bereits kleinste Veränderungen der beteiligten Moleküle können zu einem Verlust der biologischen Wirkung oder schlimmer, zur gegenteiligen Reaktion führen. Der Wirkstoff verhält sich praktisch wie ein Schlüssel gegenüber dem Rezeptor. Der Wirkstoff wird quasi in das beteiligte Biomolekül eingebettet, so daß eine entsprechende geometrische Struktur zusätzliche Grundvoraussetzung für eine mögliche Wirkung ist, daher der Name Dockingproblem.

Die folgenden Abbildungen zeigen ein Biomolekül mit einer entsprechend darin eingebetteten Struktur.

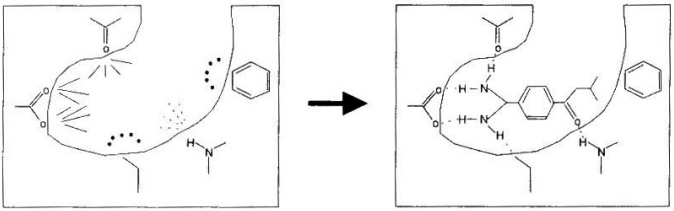


Abbildung 35: *Die Abbildung zeigt eine Bindungstasche und einen darin eingebetteten Liganden. Die durchgehenden Linien im oberen Diagramm zeigen H-Donorgruppen, die gestrichelten H-Akzeptoratome. Die Punkte stehen für hydrophobe Gruppen. (Quelle: [HKH])*

Neben einer geeigneten Geometrie ist es notwendig, diejenigen Teile einer chemischen Struktur zu erkennen, bei denen es mit hoher Wahrscheinlichkeit zu einer Wechselwirkung mit potentiellen Wirkstoffen kommt. Diese Aktivitätszentren sind beispielsweise H-Brücken, Donor- oder Akzeptorgruppen, lipophil aromatische oder aliphatische Reste und hydrophobe Gruppen, sowie Gruppen mit entsprechender Ladungsstruktur. Sind diese Zentren bekannt, so kann man daraus wiederum ihre dreidimensionale Lage bestimmen.

Der zweite Teil des Problems besteht nun darin, einen geeigneten Partner für diese Struktureigenschaften zu finden, der die gegenteiligen Eigenschaften aufweist und in einer in dem zur Verfügung stehenden Raum stabilen Form auftritt.

Um dieses Problem mit den Hilfsmitteln dieser Arbeit zu lösen, benötigt man ein entsprechendes mathematisches Modell.

Ein naheliegendes Modell bietet sich aus den bereits vorangegangenen Kapiteln an. In einem ersten Schritt, beim Auffinden der Aktivitätszentren, kann man bei solchen Zentren, die sich durch die Existenz einer bestimmten Substruktur ergeben, die entsprechende zwei- oder dreidimensionale Substruktursuche verwenden. Da alle möglichen derartigen Zentren wichtig sein könnten, müssen alle Einbettungen gefunden werden. Es genügt nicht, die Existenz eines Zentrums im Biomolekül zu ermitteln. Die Substruktursuche kann vermutlich auf topologischer Ebene durchgeführt werden, es sei denn die räumliche Anordnung einer Struktur ist ausschlaggebend für eine der zuvor aufgeführten Aktivitäten, was jedoch in der Regel nicht der Fall ist. Die dreidimensionale Lage der Einbettung kann dann aus den Koordinaten der Teststruktur übernommen werden. Können mehrere Substrukturen identische Aktivitäten hervorrufen, müssen alle in Frage kommenden Strukturen getestet werden.

Für jede der errechneten Einbettung der Substrukturen wird die geometrische Lage und Aktivität gespeichert.

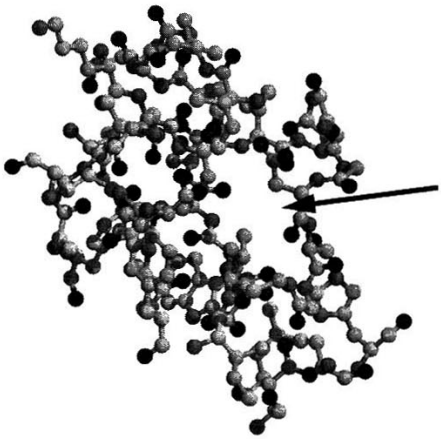
Die Summe aller daraus resultierenden Informationen ist ausschlaggebend für ein darauf aufbauendes, weiteres diskretes Molekülmodell. In diesem Modell besteht das Molekül nicht mehr aus Atomen und Bindungen, sondern aus Aktivitätszentren und Abständen. Natürlich handelt es sich dabei um eine

grobe Vereinfachung, was jedoch in diesem Fall durchaus gewollt ist. Denn ein Ziel ist, möglichst viele Kandidaten zu finden, die bezüglich diese Molekülmodells übereinstimmen, um potentielle Wirkstoffe zur Hemmung der Aktivitäten erzeugen.

Diese Modellierung kann sehr einfach in das vorhandene Modell integriert werden, indem man das bestehende Atom - Bindungsmodell von MOLGEN benutzt und die Aktivitätszentren quasi mit Atomnamen kennzeichnet. Statt des Periodensystems verwendet man dann eine Menge möglicher Aktivitätszentrenbezeichnungen. Die Atome besitzen natürlich dreidimensionale Koordinaten; Bindungen sollten zwischen allen Atomen eingefügt werden. Es wäre auch denkbar, Bindungen zwischen Aktivitätszentren nur dann aufzubauen, wenn die Entfernung zwischen den Zentren innerhalb eines vorgegebenen Bereichs liegt. Dabei ist zu berücksichtigen, daß der entstehende Bindungsgraph zusammenhängen ist.

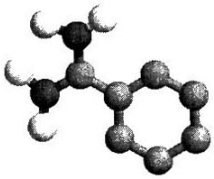
Die Suche nach geeigneten Kandidaten kann dann sowohl mit Hilfe vorhandener Daten durchgeführt werden, als auch mit Unterstützung durch einen entsprechenden Generator, wie beispielsweise MOLGEN.

Die nachfolgende Abbildung zeigt das Molekül Trypsin



In dem mit einem Pfeil bezeichneten Bereich ist eine Tasche der Struktur erkennbar. Mit LUDI wurden die aktiven Zentren um diese Tasche berechnet:

Die Kugeln stellen dabei keine Atome dar, sondern klassifizieren Eigenschaften der Tasche im Trypsin. Eine nachfolgende Struktursuche in einer Datenbank unter Berücksichtigung dieser vorgegebenen Merkmale ergab die folgende, mögliche Lösungsstruktur:



6.9. Einsatz in der Kombinatorischen Chemie

Die Chemieindustrie beschäftigt sich in letzter Zeit zunehmend mit der sogenannten kombinatorischen Chemie. Der Ausgangspunkt für die Entstehung der kombinatorischen Chemie liegt in den Erfahrungen, die bei der De- Novo- Entwicklung von Wirkstoffen gewonnen wurden. Dort stellte man sehr schnell fest, daß es nur zu einem unbefriedigenden Maße möglich ist, neue Strukturen mit vorgegebenen Eigenschaften zu entwerfen und ihr Verhalten zu simulieren.

Man besann sich also wieder auf klassische Syntheseverfahren und versuchte Wirkstoffe im Labor zu generieren. Die Synthese von Stoffen mit Hilfe herkömmlicher Verfahren ist jedoch sehr aufwendig und nimmt sehr viel Zeit in Anspruch.

Die ursprünglich Idee bei der kombinatorischen Chemie war nun, möglichst schnell eine möglichst große Anzahl von Strukturen zu erzeugen, indem man ausgehend von einigen Grundkomponenten Mischungen erzeugt und diese Mischungen auf ihre Aktivität bezüglich eines bestimmten chemischen Vorganges testet. Wurde eine gewünschte Reaktion ausgelöst, so muß man mit Unterstützung erfahrener Analytiker die aktive Struktur ermitteln.

Da dies jedoch wiederum äußerst aufwendig ist, erzeugt man inzwischen durch Kombinationen von gezielten Mischungen eine relativ überschaubare Anzahl von möglichen Kombinationen von Ausgangsstrukturen bzw. Fragmenten.

Durch die Mischung der Elemente können Überlagerungen von molekularen Aktivitäten einerseits eine gewünschte Wirkung hervorrufen, die bei einer einzelnen Struktur dann nicht mehr gegeben ist, andererseits können diese Überlagerungen auch tatsächlich im Detail vorhandene Aktivitäten auch verhindern. Eine genaue Klassifizierung des Verhaltens des Gemisches ist also nicht möglich.

Zur Analyse der so entstandenen Substanzen werden sogenannte kombinatorische Bibliotheken benötigt. Die Mathematik liefert hier sehr effiziente Hilfsmittel zur erschöpfenden Konstruktion dieser vielfältigen Strukturen. Dabei wird sowohl die kanonische Numerierung als auch die Isomorphieerkennung von Strukturen verwendet. Die Arbeit von Wieland [WI96] beschäftigt sich mit der Erzeugung derartiger Bibliotheken. Weitere Informationen zu diesem Thema finden sich auch in [BKK].

7. Andere Verfahren

Die in dieser Arbeit vorgestellten Verfahren sind natürlich nicht die einzigen, die bisher entwickelt wurden. Jedoch besitzen die herkömmlichen Verfahren nicht die flexibilität des objektorientierten Ansatzes.

Da die Entwicklung der Graphentheorie zu einem Großteil in der Betrachtung chemischer Strukturen begründet liegt, ist es nicht verwunderlich, daß auf diesem Gebiet eine große Anzahl guter Algorithmen entstanden ist. Die folgenden Abschnitte geben einen kurzen Überblick über einige dieser Verfahren:

7.1. 2D - Substruktursuche bei Molekülen

7.1.1. HOSE - Code

Bei BASF wird die Substruktursuche mit Hilfe des Hose - Codes durchgeführt. Der Hose - Code beschreibt Umgebungen von Atomen, ähnlich der erweiterten Konnektivität aus dem Morgan - Algorithmus. Ermittelt man diese erweiterte Konnektivität nicht nur für die direkt benachbarten Atome, sondern solange auch für weiter entfernte Atome, bis alle Atome des Moleküls miteinbezogen wurden, und ordnet man diese Informationen je nach Entfernung vom betrachteten Atom, so kann man quasi durch eine lexikographische Ordnung auf den Umgebungsmerkmalen der unterschiedlichen Entfernungen von jedem Atom aus das gesamte Molekül beschreiben. Eine Substruktursuche geht dann so vor sich, daß für das zu suchende Molekül ebenfalls für alle betrachteten Atome diese Entfernungsmerkmale der unterschiedlichen Tiefe festgelegt werden. Durch Sortierung und nachfolgenden Vergleich kann man dann direkt die evtl. vorhandene Einbettung bestimmen.

Der Nachteil dieses Verfahrens ist der große Speicherbedarf, der zur Berechnung alle HOSE - Codes aller Schalentiefen benötigt wird.

Die Schnittstellen zum vorliegenden Algorithmus zur Substruktursuche können so modifiziert werden, daß der Algorithmus wie oben abläuft. Dazu ist es notwendig, alle HOSE Codes eines Atoms mit der *dom* - Funktion zur Verfügung zu stellen und eine entsprechende Vergleichsfunktion zu implementieren. Gibt die Vorklassifizierung noch keine vollständige Zerlegung der Atommenge an, sind mehrere Einbettungen der Substruktur möglich. Diese werden dann über das Backtracking sofort geliefert.

Eine Differenzierung zwischen geschlossenen und offenen Substrukturen sind im Hose - Code nicht vorgesehen. Entsprechende Erweiterungen müßten, falls überhaupt möglich, noch vorgenommen werden.

7.2. 3D - Klassifizierung von Molekülen

In den letzten Jahren gewinnen Systeme zur 3D - Struktur- und Substruktursuche zunehmend an Bedeutung. Die Strukturdaten stammen dabei entweder aus Experimenten oder sie wurden mit speziellen Programmen berechnet. Der größte Teil der experimentell aufgeklärten und veröffentlichten Strukturen wird in der Cambridge Structural Database [CS] gesammelt. Daneben existieren jedoch auch einige andere Quellen, wie beispielsweise die Brookhaven Protein Data Bank oder die NBS Crystal Data base.

Das Cambridge Crystallographic Data Center wurde bereits 1965 aufgebaut, um 3D - Strukturen zu katalogisieren. Es ist inzwischen die weltweit größte Datenbank für verifizierte, experimentelle Ergebnisse in der Chemie. Schon 1990 wurden der Datenbasis jährlich etwa 8500 zusätzliche Strukturen hinzugefügt.

Verschiedene Model - Builder wurden für die Konstruktion von 3D - Strukturen, ausgehend von der

zweidimensionalen Connection Table geschrieben. Der wohl bekannteste Vertreter dieser Art ist wohl Concord bzw. Corrina [SG]. Aufgrund seiner hohen Geschwindigkeit wurde es zur Konvertierung vieler vorhandener 2D - Datenbanken in 3D genutzt.

7.2.1. 3D - Substruktursuche

Das für die 2D - Substruktursuche verwendete graphentheoretische Modell dient weiterhin als Grundlage für die Struktursuche. Zusätzlich werden geometrische Restriktionen eingeführt, die eine Unterscheidung der Strukturen auf räumlicher Ebene erlauben.

Bei großen Datenmengen ist wiederum ein effektives Screening erforderlich, um die Anzahl der auf Isomorphie oder Teilisomorphie zu testenden Strukturen zu minimieren. Eine Reihe von effektiven Unterscheidungsmerkmalen liefert beispielsweise der von Varmuza verwendete Vektor zur Clusteranalyse. Die BASIC (Basel Information Center) [GK] Gruppe von Unternehmen stellte ebenfalls eine Liste von Fragmenten zusammen, die später Basis für das Suchsystem des CAS ONLINE von STN International [DF] wurden. Inzwischen umfaßt diese Liste bereits einige tausend Screening - Eigenschaften, die eine effektive Reduktion der Datenbasis bei Suchanfragen erlauben.

7.3. Kanonische Numerierung von Molekülen

In verschiedene Ansätzen wurde bereits die Fragestellung einer kanonischen Numerierung von Molekülen erläutert. Darunter ist der wohl bekannteste der Morgan-Algorithmus [HLM], der vom Chemical Abstract Service verwendet wird. Darüber hinaus gibt es zu diesem Problem noch Arbeiten von Randic [MR], Hendrickson/Toczko [HT] und die aus dem Dendral Projekt [LBFL] stammenden Ergebnisse. Weitere Veröffentlichungen stammen von [KP], [SM], [SU] und [JG]. Wie man sieht, ist die kanonische Numerierung von Strukturen Gegenstand vieler Bemühungen. Letztendlich beruhen alle Algorithmen auf einem ähnlichen Grundprinzip. Die Moleküle werden als Graphen dargestellt und es wird versucht, anhand augezeichneter Eigenschaften eine eindeutige Darstellung zu finden. Die Unterschiede der Verfahren beschränken sich in der Regel ausschließlich auf die Art der verwendeten Eigenschaften. Morgan hat beispielsweise bei seinen Überlegungen noch keine stereochemischen Eigenschaften einbezogen, da diese zur damaligen Zeit irrelevant erschienen. Wipke/Dyott versuchen in [WD] den Algorithmus um diese Komponente zu erweitern. Weit verbreitet ist bei der Wahl der Eigenschaften auch der Einsatz der sog. Topologischen Indizes. ([RH])

Die verwendeten Eigenschaften sind in der Regel lokale Eigenschaften, d. h. Eigenschaften, die bei der vorliegenden Molekülschnittstelle durch die *dom* - Funktion dargestellt werden. Eine darüber hinausgehende Beziehung vergleichbar der *ref* - Funktion findet man beispielsweise als Erweiterung der Connection-Table in [KP]. Eine Klassifizierung bezüglich globaler Eigenschaften, wie sie beispielsweise bei der Berücksichtigung bereits bekannter Isomorphismen in Kapitel 4.4. gezeigt wird, ist nicht vorhanden.

Der Vorteil des vorliegenden Algorithmus liegt also vor allem in der Möglichkeit, relativ schnell neu charakteristische Eigenschaften molekularer Strukturen berücksichtigen zu können. Darüber hinaus bietet er die Möglichkeit Eigenschaften zu implementieren, die nicht ausschließlich auf Atom by Atom Beziehungen beruhen.

Durch den Einsatz der iterierten Klassifizierung wird eine zusätzliche, frühzeitige Einschränkung der Kandidatenmenge erreicht, so daß gegenüber klassischen Verfahren, vor allem bei großen Strukturen ein erheblicher Geschwindigkeitsvorteil zu erwarten ist.

8. Bibliographie

[AH] Andreas Heuer: *„Objektorientierte Datenbanken - Konzepte, Modelle, Systeme"*, Addison-Wesley Bonn (1993)

[AH87] S. Abiteboul and R. Hull, IFO: *„A formal semantic database model"*, ACM Trans. Database Systems 12, 525-565 (1987)

[AK] A. Kirk: *„Efficiency considerations in the canonical labelling of graphs"*, Technical report TR-CS-85-05, Computer Science Department, Australian National University (1985)

[BG] H. Bandemer, S. Gottwald: *„Einführung in Fuzzy - Methoden"*, Verlag Harry Deutsch, Frankfurt a.M. (1990), 73

[BH] K. Bruhns: *„Alexander von Humbold"*, Bibliographie in drei Bänden, Brockhaus (1872)

[BJL] Th. Beth, D. Jungnickel, H. Lenz: *„Design theory"*, B.I.-Wissenschaftsverlag, Zürich (1985)

[BKLW] A. Betten, A. Kerber, A. Kohnert, R. Laue, A. Wassermann: *„The discovery of simple 7 - designs with automorphism group* $PTL(2,32)$ " Proc. AAECC-11 (11th International Symposium on Applied Algebra, Algebraic Algorithms and Error - Correcting Codes), Paris 1995, LN in Comp. Sc. 948 (131-145), Springer Berlin (1995)

[BKK] H.-J. Böhm, G. Klebe, H. Kubinyi: *„Wirkstoffdesign - Der Weg zum Arzneimittel"*, Spektrum Akademischer Verlag (1996)

[BS] B. Stroustrup: *„Die C++ Programmiersprache"*, Addison Wesley (1992), 2. Auflage

[CL] U. Clausen: *„Objektorientiertes Programmieren"*, Springer Verlag Berlin (1993)

[CS] F. H. Allen, J. E. Davies, J. J. Galloy, O. Johnson, O. Kennard, C. Macrae, E. M Mitchell, G. F. Mitchell, J. M. Smith, D. G. Watson: *„The development of version 3 and 4 of the Cambridge Structural Database System"*, J. Chem. Inf. Comp. Science 31, 187 (1991)

[CY] Peter Coad, Edward Yourdon: *„Objektorientierte Analyse"*, Prentice-Hall New York (1994)

[DB] Denis M. Bayada: *„Algorithms for the Similarity of Molecules"*, PhD thesis at the University of Leeds, 2/1994

[DB2] IBM Corporation: *„DATABASE 2 OS/2, SQL Reference"*, Document Number S62G-3667-00, Part Number 62G3667, Copyright 1993

[DC1] P. J. De Clercq: *„Systematic Conformational Analysis. A Microcomputer Method for the Semiquantitative Evaluation of Polycyclic Systems Containing Five-, Six- and Seven-membered Rings"*, Tetrahedron Vol. 40 (1984) 19, 3729-3738

[DC2] P. J. De Clercq: *„Systematic Conformational Analysis. Torsion Constraint Evaluation in Cyclic Systems"*, Tetrahedron Vol. 37 (1981) 24, 4277-4286

[DC3] P. J. De Clercq: *„Systematic Conformational Analysis. General Method Rapid Evaluation. Its Application do Hydroazulene System"*, J. Org. Chem. 46, (1981) 667-675

[DF] P. G. Dittmar, N. A. Farmer, W. Fisanick, R. C. Haines, J. Mockus: *„The CAS Online Search System. I. General System Design and Selection, Generation and Use of Search Screens"*, J. Chem. Inf. COmp. Science 23, 93 (1983)

[DJ] Dieter Jungnickel: *„Graphen, Netwerke und Algorithmen"*, BI-Wissenschaftsverlag (1994)

[DK] Donald E. Knuth: *„The Art Of Computer Programming - Volume 3 / Sorting and Searching"*, Addison - Wesley Series in Computer Science and Information Processing, Addison Wesley (1973)

[DK2] Donald E. Knuth: *„Estimating the eficiency of Backtrack Programs"*, Mathematics of Com-

putations, Vol. 29, No. 129 (1975), p. 121-136

[GK] W. Graf, H. K. Kaindl, H. Kneiss, R. Warszawski: *„The third BASIC fragment search dictionary"*, J. Chem. Inf. Comp. Science 22,177 (1982)

[GKL] R. Grund, A. Kerber, R. Laue: *„Construction of discrete structures, especially of chemical isomers."*, to appear in Discrete Applied Mathematics.

[GRVI] Stéphane Grumbach, Victor Vianu: *„Tractable Query Languages for Complex Object Databases"*, Journal of Comp. and System Sciences 51, 149-167 (1995)

[GT] Gottfried Tinhofer: *„Methoden der angewandten Graphentheorie"*, Springer Verlag, Wien (1976)

[HA] Marshall Hall, Jr: *„The theory of groups"*, Chelsea Publiching Company, New York (1976) second edition

[HLM] Morgan, H. L.: *„The Generation of a unique dscription for chemical structures - A technique developed at Chemical Abstracts Service"*, J. Chem. Doc. (1965), 5, 107-113

[HKI] Hugo Kubinyi: *„Der Schlüssel zum Schloß - I. Grundlagen der Arzneimittelentwicklung"*, Pharmazie in unserer Zeit / 23. Jahrgang (1994/3)

[HKII] Hugo Kubinyi: *„Der Schlüssel zum Schloß - II. Hansch-Analyse, 3D-QSAR und De novo-Design"*, Pharmazie in unserer Zeit / 23. Jahrgang (1994/5)

[HM] Horst Müller: *„Diskrete Algebraische Strukturen I und II"*, Arbeitsberichte des Instituts für Informatik, FAU Erlangen - Nürnberg, Band 6/5 und Band 8/2 (1987/79)

[HMZ] M. Hesse, H. Meier, B. Zeeh: *„Spektroskopische Methoden in der organischen Chemie"*, Georg Thieme Verlag, Stuttgart (1991)

[HT] Hendrickson, J. B.; Toczko, A. G.: *„Unique numbering and cataloguing of molecular structures"*, J. Chem. Inf. Comput. Sci. (1983), 23, 171-177

[HO] Wolfram Hofmann: *„Algorithmen zur automatischen Unterstrukturerkennung"*, Diplomarbeit Prof. Laue 4/1993

[HS] Erika Horn, Wolfgang Schubert: *„Objektorientierte Software-Konstruktion - Grundlagen, Modelle, Methoden, Beispiele"*, Hanser München (1993)

[HST] Harald S. Stone: *„Discrete Mathematical Structures and their Applications"*, Science Research Associates, Inc. USA (1973)

[HUPP] Bertram Huppert: *„Endliche Gruppen I"*, Springer Verlag Heidleberg (1976)

[HW] Wiener, H. : *„Structural determination of paraffin boiling points"*, J. Am. Chem. Soc. (1947), 69, 17-20

[IG] W. D. Ihlenfeld, J. Gasteiger: *"Der Einsatz von Hashcodes zur Erkennung der strukturellen Ähnlichkeit von Molekülen"*, Software-Entwicklung in der Chemie, Springer-Verlag Berlin Heidelberg 1988

[JG] Jochum, C., Gasteiger, J. : *„Canonical numbering and constitutional symmetry"*, J. Chem. Inf. Comp. Sci. (1977), 17, 113-117

[JL] J. S. Leon: *„Permutation group algorithms based on partitions"*, J. Symbolic Computation 21, 533-583 (1991)

[JM] Jürgen Müller: *„Algorithmen zur Isomorphieerkennung bei mathematischer Darstellung chemischer Strukturen"*, Dipomarbeit an der Universität Bayreuth, Prof. Dr. R. Laue (1994)

[JS] J. K. Seydel: *„QSAR and strategies in the design of bioactive compounds"*, VCH Publishers New York (1985)

[KE] A. Kerber: *„Algebraic combinatorics via finite group actions"*, BI-Wissenschaftsverlag

(1991)

[KH] L. B. Kier, L. H. Hall: *„Molecular Connectivity in Structure-Activity Analysis"*, Research Studies Press, Chichester, 1986

[KK] A. Kerber, A. Kohnert, A. Lascoux: *„SYMMETRICA, an object oriented computer-algebra system for the symmetric group"*, J. Symbolic Computation 14, 195-203 (1992)

[KP] V. Kvasnicka, J. Pospichal : *„Canonical indexing and the constructive enumeration of molecular graphs"*, J. Chem. Inf. Comp. Sci. (1990) 30

[KV1] K. Varmuza, H. Scsibrany: *„Common substructures in groups of compounds exhibiting similar mass spectra"*, Fresenius J. Anal. Chem. 344 (1992) 220-222

[KV2] K. Varmuza, H. Scsibrany: *„ToSiM: PC-Software for the investigation of topological similarities in molecules"*, Software development in Chemistry, Vol. 8 (1994)

[KY] G. J. Klir, B. Yuan *„Fuzzy Sets and Fuzzy Logic - Theory and Applications"*, Prentice Hall, New Jersey (1995)

[LA1] R. Laue: *„Construction of Combinatorial Objects"*, Bayreuther Mathematische Schriften 43 (1993)

[LBFL] Lindsay, R.K. and Buchanan, B.G. and Feigenbaum, E.A. and Lederberg, J.: *„Applications of Artificial Intelligence for Organic Chemistry: The Dendral Project"*, McGraw-Hill, New York (1980)

[LK] H. P. Latscha, H. A. Klein: *„Organische Chemie - Basiswissen II"*, Springer Verlag Berlin, Heidelberg (1990)

[LU1] Eugene M. Luks: *"Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time"*, Journal of Computer and System Science 25 (1986), 42-65

[LU2] Eugene M. Luks: *"Permutation Groups an Polynomial-Time Computation"*, Deparment of Computer and Information Science, University of Oregon, 2/1993

[MEY] K. Meyberg: *„Algebra"*, Hanser (1975)

[MF] M. Fischer: *„Algorithmen zur Erkennung von Aromatizitäts- und Tautomerie-Überlappungen in organischen Strukturen"*, Dipomarbeit an der Universität Bayreuth, Prof. Dr. R. Laue, in Zusammenarbeit mit IBM Heidelberg (1996)

[MG1] C. Benecke, R. Grund, R. Hohberger, R. Laue, A. Kerber, Th.Wieland: *„Chemical Education via MOLGEN"*, J. Chem. Edu. 72 (1995), 403 - 406

[MG2] C. Benecke, R. Grund, R. Hohberger, R. Laue, A. Kerber, Th.Wieland: *„MOLGEN+, a generator of connectivity isomers and stereoisomers for molecular structure elucidation"*, Anal. Chim. Acta 314 (1995), 141 - 147

[MG3] C. Benecke, R. Grund, R. Hohberger, R. Laue, A. Kerber, Th.Wieland: *„Lehren, Lernen und Forschen mit MOLGEN"*, CLB Chemie in Labor und Biotechnik 44/12 (1993) 636-638

[MG4] C. Benecke, R. Grund, R. Hohberger, R. Laue, A. Kerber, Th.Wieland: *„MOLGEN, a Computer Algebra System for the Generation of Molecular Graphs"*, Computer Algebra in Science and Engineering, World Scientific, Singapore (1995), 260 - 272

[MG5] C. Benecke, R. Grund, R. Hohberger, R. Laue, A. Kerber, Th.Wieland: *„Die Veranschaulichung von Molekülen im Chemieunterricht mit Hilfe des Programmsystems MOLGEN+"*, MNU - Der mathem. - naturw. Unterricht 48 (1995), 156 - 163

[MK] B. D. McKay: *„Computing automorphisms and canonical labelling of graphs"*, Proc. Intern. Conf. on Combinatorial Theory, Lecture Notes in Mathematics Nr. 686, Springer Verlag Berlin (1977)

[MR] M. Randic: *„On canonical numbering of atoms in a molecule and graph isomorphism"*, J. Chem. Inf. Comput. Sci. (1977) 17, 171-180

[PK] P. Krief, J. Plaice: *„Prototyping with Objects"*, Prentice Hall, London (1996)

[PS] S. L. Pfleeger, D. W. Straight: *„Introduction to discrete Structures"*, John Wiley & Sons, New York (1985)

[RH] Rouvray, D.H.: *„Topological Indices as chemical behaviour descriptors"*, Congr. Numerantium (1985), 49, 161-179

[SB] J. Stoer, R. Burlisch: *„Numerische Mathematik 2"*, Springer Verlag Berlin (1990)

[SCH] Mathias Schimmel: *„Algorithmen zur Strukturerkennung, insbesondere bei Molekülen"*, Dipomarbeit an der Universität Bayreuth, Prof. Dr. R. Laue (1993)

[SDW] Robert Sedgewick: *„Algorithmen in C++"*, Addison-Wesley Bonn (1992), 2. Auflage

[SG] J. Sadowski, J. Gasteiger: *„From Atoms and Bonds to three - dimensional coordinates: Automatic Model Builders"*, Chem. Rev. 93, 2576 (1993)

[SM] Shelley, C. A., Munk M. J.: *„An approach to the assignment of canonival connection tables and topological symmetry perceptions"*, J. Chem. Inf. Comp. Sci. (1979), 19, 247-250

[SPM] Chemical Concepts GmbH, 69469 Weinheim: *„SpecInfo 3 manual"*

[SU] Schubert, W., Ugi, I.: *„Constitutional symmetry and unique descriptions of molecules"*, J. Am. Chem. Soc. (1978), 100, 37-41

[TG] Th. Grüner: *„Ein neuer Ansatz zur rekursiven Erzeugung von schlichten Graphen"*, Diplomarbeit am Lehrstuhl II für Mathematik, Prof. Laue, (1995)

[UD] I. Ugi, J. Dugundjl, R. Kopp, D. Marquarding: *„Perspectives in Theoretical Stereochemistry"*, Lecture Notes in Chemistry 36, Springer New York (1984).

[VAC] IBM Corporation: *„IBM VisualAge C++ for OS/2 - Language Reference Version 3.0"*, Document Number S25H-6963-00, Part Number 25H6963

[VK] A. Kerber: *„Diskrete Strukturen"*, Vorlesung Univ. Bayreuth

[VL1] R. Laue: *„Graphentheoretische Algorithmen"*, Vorlesung Univ. Bayreuth

[VL2] R. Laue: *„Diskrete Strukturen"*, Vorlesung Univ. Bayreuth

[WA] W. Alt: *„Approximationstheorie"*, Vorlesung Univ. Bayreuth

[WD] Wipke, W. T., Dyott, T. M.: *„Stereochemical unique naming algorithm"*, J. Chem. Inf. Comp. Sci. (1977) 17, 113-117

[WH] W. Hofmann: *„Algebraische Algorithmen zur automatischen Unterstrukturerkennung"*, Dipomarbeit an der Universität Bayreuth, Prof. Dr. R. Laue (1993)

[WIE] T. Wieland: *„Computerunterstützte Berechnung von Stereoisomeren"*, Dipomarbeit an der Universität Bayreuth, Prof. Dr. A. Kerber, Prof. Dr. R. Laue (1994)

[WI96] T. Wieland: *„Konstruktionsalgorithmen bei molekularen Graphen und deren Anwendung"*, Dissertation an der Universität Bayreuth, Prof. Dr. A. Kerber, Prof. Dr. R. Laue (1996)

[WWW] R. Wirfs-Brock, B. Wilkerson, L. Wiener: *„Objektorientiertes Software-Design"*, Hanser / Prentice Hall Coedition (1993)