Modellierung geometrischer Constraints für CAD-Anwendungen

Roland Berling, Chun Du, Walter Hower, Manfred Rosendahl
Institut für Informatik, Universität Koblenz-Landau, Rheinau 1, 56075 Koblenz
{berling,chun,walter,ros}@informatik.uni-koblenz.de

Zusammenfassung

Die Modellbildung mit Hilfe informatischer Methoden erfordert bei CAD-Anwendungen wie auch bei anderen Anwendungen sowohl die Abbildung von Strukturen der Objekte des Gegenstandsbereiches als auch die Beschreibung von Beziehungen der Strukturkomponenten untereinander. Als Modellierungsmittel für den letzteren Aspekt haben sich in verschiedenen Anwendungen Constraints und Constraint-Systeme als geeignet erwiesen. Der vorliegende Text gibt einen Überblick über Constraint-Modellierung für CAD-Anwendungen mit einem Schwerpunkt auf der Beschreibung von Beziehungen geometrischer Art. Es werden drei Ansätze und ihre Anwendungsmöglichkeiten vorgestellt. Zwei von ihnen dienen im besonderen der Modellierung geometrischer Invarianten; Constraint-Modell und Verfahren sind hier zur Berechnung einer Lösung ausgelegt. Der dritte Ansatz hat seinen Ursprung in allgemeinen Untersuchungen zu *Constraint-Satisfaction* in der KI, er ist adaptiert worden für Probleme im CAD, die eine Suche nach allen Lösungen in endlichen Lösungsräumen erfordern.

1 Einführung

Die Modellbildung mit Hilfe informatischer Methoden erfordert bei CAD-Anwendungen sowohl die Abbildung von Strukturen der Objekte des Gegenstandsbereiches als auch die Beschreibung von Beziehungen der Strukturkomponenten untereinander. Als Modellierungsmittel für den letzteren Aspekt haben sich in verschiedenen Anwendungen Constraints und Constraint-Systeme als geeignet erwiesen. Die Entwicklung von CAD-Systemen erfordert in der Modellbildung die Berücksichtigung der verschiedenen Aufgaben, die ein solches System zu erfüllen hat. Der Entwurf eines technischen Objekts kann in Phasen strukturiert werden. Das dabei entstehende Modell des technischen Objekts wird als eine Verbindung von Teilmodellen beschrieben, die die verschiedenen Aspekte des Objekts vom abstrakten Funktionsprinzip über seine Form bis hin zu fertigungstechnischen Daten repräsentieren. Aus den Phasen des Entwurfsvorgangs und den Teilmodellen heraus leiten sich unterschiedliche Anforderungen an die Modellierung von Beziehungen zwischen Teilmodellen und zwischen Komponenten innerhalb von Teilmodellen ab. Konsequenterweise kommen verschiedene, problembezogen-gestaltete *Constraint-Modelling*- und *Constraint-Satisfaction*-Techniken zum Einsatz. Der vorliegende Text stellt drei Ansätze vor. Zwei Ansätze sind Vorschläge zur Modellierung gleichungsbasierter Relationen, der dritte Ansatz arbeitet mit Ungleichungen und Lösungsintervallen. Die folgenden Abschnitte der Einleitung explizieren die unterschiedlichen Anforderungen.

1.1 Constraints für erweiterte Geometriemodelle

Das zentrale Modell in einem CAD-System ist das Geometriemodell. Eine Konstruktion ist dabei nicht nur durch die Menge der sie beschreibenden geometrischen Elemente spezifiziert, sondern insbesondere auch durch geometrische und nichtgeometrische Beziehungen zwischen diesen Elementen. Nichtgeometrische Beziehungen sind Abhängigkeiten zwischen Dimensionen (Längen und Winkeln) und anderen Größen, wie z.B. Kräften, die auf das Objekt wirken. Beide Arten von Invarianten modellieren Beziehungen zwischen Geometrie- und anderen Teilmodellen.

Geometrische und nichtgeometrische Beziehungen können als Constraints in Form der bekannten Gleichungen (Beispiel: Tangentialität) oder durch Konstruktionsbeziehungen auf den Parametern der geometrischen Elemente beschrieben werden ([BeRo92]). Das Constraint-Satisfaction-Problem ist dann definiert als die Aufgabe, aus einer gegebenen Menge von singulären Werten für einige Variablen mit Hilfe des Constraint-Systems und eines geeigneten Verfahrens singuläre Werte für die restlichen Variablen zu berechnen. Die Wahl des Verfahren ist abhängig von der Phase des Entwurfsprozesses.

In der konzeptuellen Entwurfsphase muß das Constraint-System in allen Variablen evaluierbar sein, da auch Änderungen von nichtgeometrischen Größen durch Änderungen der Geometrie untersucht werden. Ein gängiges Verfahren zur Berechnung einer konsistenten Belegung ist die Berechnung einer Zuordnung von Constraints zu noch nicht gebundenen Variablen durch Graph-Matching-Verfahren. Zur Evaluierung der Constraints und zur Auflösung von Kreisen im Graphen werden numerische Methoden benutzt, da geometrische wie auch nichtgeometrische Beziehungen im allgemeinen nichtlinear sind. Referenzen sind hier: [LiGo82, SeGo87, ChSc89] und [BeDuRo91, BeRo92].

In der Variantenerzeugung sind die Werte für nichtgeometrische Größen und Dimensionen bekannt und konsistent, der geometrische Ort der Konstruktion ist dagegen zu berechnen. Es werden hier oft spezielle Verfahren eingesetzt, die geometrische Konstruktionen identifizieren und effizient berechnen können, um ein numerisches Lösen von Gleichungssystemen zu vermeiden ([AlMaRi91, ArWa91, Owen91, Roll91, VeScRo92]).

In den Kapiteln 2 und 3 werden zwei Ansätze vorgestellt, die in diese Klasse von Constraint-Modellen für CAD-Anwendungen fallen. Der erste Ansatz: *konstruktive Modellierung* beschreibt die geometrischen und nichtgeometrischen Beziehungen durch die Konstruktionsoperationen, die bei Aufbau und Änderung einer Geometrie

benutzt werden. Das Resultat ist eine Abbildung, die eine Konstruktion auf der Basis einer Menge von Parametern vollständig beschreibt. Die Änderbarkeit abhängiger Variablen bleibt dabei durch die Evaluierbarkeit von Umkehrabbildungen erhalten. Die nachträgliche Definierbarkeit von Constraints auf einer bestehenden Abbildung erlaubt auch die Beschreibung von Konstruktionen, die rein konstruktiv nicht definierbar sind. Der Ansatz ist wegen der effizienten Berechenbarkeit von Varianten durch neue Werte für die Parameter besonders für späte Phasen des Entwurfsprozesses geeignet. Er ist in dem prototypischen CAD-System RelCAD realisiert worden. Referenz ist [DuRoBe93, BeDuRo91]. Der zweite Ansatz: *deklarative Modellierung* beschreibt geometrische Beziehungen durch eine Reihe von Basis-Constraints auf Punkten und Dimensionen als Variablen. Im Gegensatz zum ersten handelt es sich hier um eine deklarative Beschreibung der Beziehungen. Der resultierende Constraint-Graph wird mit Hilfe von lokaler Propagierung evaluiert. Die spezifische Art der Modellierung erlaubt es, daß bei der Evaluierung zur Berechnung der Geometrie auf Konstruktionsoperationen zurückgegriffen werden kann, und somit die Verwendung numerischer Methoden weitestgehend vermieden werden kann. Die Technik ist aus dem ersten Ansatz heraus entwickelt worden und eignet sich insbesondere für die frühen Phasen des Entwurfsprozesses. Referenz ist [BeRo93].

1.2 Constraints zur Suche in Lösungsräumen

Neben der Repräsentation geometrischer und nichtgeometrischer Beziehungen können Constraints noch für andere Aufgaben im CAD verwendet werden. In den frühen Phasen des Entwurfsprozesses aber auch in der Analyse und Simulation von Konstruktionen ist die Berechnung von Konfigurationen in einem durch Randbedingungen festgelegten Lösungsraum eine Aufgabe für CAD-Systeme. Ein Beispiel ist die Gestaltung eines Büroraumes bei bekannter Raumdimensionierung und bekannter Anzahl von Ausstattungsstücken. Gesucht sind die möglichen Positionen der Raumausstattung unter einer Menge gegebener Einschränkungen. Das Constraint-Satisfaction-Problem ist dann definiert als die Aufgabe, bei einer gegebenen Menge von Variablen mit endlichen Wertebereichen und einer Menge auf den Variablen definierter Constraints, alle global konsistenten Belegungen der Variablen zu berechnen ([Mack92, Freu78]). In der Anwendung der Methoden für CAD-Systeme ist es daher erforderlich, für die dortigen Problemstellungen geeignete qualitative Abstraktionen zu finden, die endliche Wertebereiche und damit endliche Lösungsräume garantieren.

In Kapitel 4 wird ein Ansatz vorgestellt, der diese Art von Constraint-Modellierung unterstützt. Der Algorithmus macht in einer ersten Phase eine Art Vorverarbeitung, welche die Wertebereiche der Variablen bereits vorfiltert (falls höher-stellige Constraints vorliegen); die darauffolgende zweite Phase führt die tatsächliche Berechnung der globalen Konsistenz durch. Es wird eine beispielhafte Anwendung auf ein Problem der Verteilung geometrischer Objekte innerhalb einer anderen Geometrie vorgestellt. Referenz ist [HoRoBe93]. Die Verwendung von Constraints zur Modellierung von Konfigurationsproblemen ist möglicherweise auch dazu geeignet, die Verwendung von Toleranzbereichen für geometrische Elemente mit der Modellierung von geometrischen und nichtgeometrischen Beziehungen zu verbinden.

2 Konstruktive Modellierung: RelCAD

2.1 Das geometrische Modell

Zur Verwaltung der geometrischen Zusammenhänge in einem geometrischen Modell müssen für die einzelnen geometrischen Modell-Primitive (GMP) nicht nur die Koordinaten gehalten werden, sondern auch die Abhängigkeiten zu anderen Modellelementen. Für die Benutzung eines Modell-Primitivs durch ein anderes sind jedoch nur dessen Koordinaten notwendig. Beispiel: Zur Konstruktion einer Tangente zwischen zwei Kreisen sind nur die Koordinaten der Kreise, nicht jedoch ihre Konstruktionsart maßgebend. Im Sinne einer objektorientierten Realisierung bietet es sich daher an, für die geometrischen Elemente zunächst Basisklassen (absolute GMP) einzuführen und aus diesen die bedingten, d.h. abhängigen geometrischen Modell-Primitive (conditioned GMP) abzuleiten.

Das geometrische Modell von RelCAD kennt zunächst fünf Basisklassen mit folgenden Daten:

value
: reeller Wert

point
: x-, y-Koordinate

line
: Anfangs-, Endpunkt

circle
: Mittelpunkt, Radius

arc
: Mittelpunkt, Radius, Anfangs-, Endwinkel

Die absoluten geometrischen Primitive bestimmen die geometrische Form eines Elements, enthalten jedoch noch keine Abhängigkeiten.

Um die Abhängigkeiten darzustellen, werden von den absoluten Klassen die bedingten Klassen abgeleitet, die die *conditioned GMP* beschreiben. Aus der Klasse `value` wird abgeleitet:

value_1P
: Abgeleitet aus den Koordinaten eines Punktes:
x-, y-Koordinate, Abstand oder Winkel zum Nullpunkt

value_2P
: Abgeleitet aus den Koordinaten zweier Punkte:
Abstand, Winkel

value_line, value_circle, value_arc
: Abgeleitet aus den Daten eines referierten Objekts:
Länge, Radius (z.B.)

exp
: Ausdruck, aus Operator und Operanden vom Typ value

Aus der Klasse `point` wird abgeleitet:

pointref
: Koordinaten sind Instanzen von

value oder von aus
value abgeleiteten Klassen.

pointrel
Koordinaten bestimmen sich durch einen Bezugspunkt und Offsets für x- und y-Koordinaten.

pointitem
Koordinaten sind definierte Punkte anderer Objekte.

pointinters
Punkt ist Schnittpunkt zweier Objekte.

pointlot, pointtangente
Punkt ist als Lot oder Tangentialpunkt auf bzw. an andere Objekte gegeben.

Von `line` wird abgeleitet:

line_2o
Linie ist durch zwei Referenzobjekte bestimmt: Tangente von Punkt an Kreis, Tangente an zwei Kreise (z.B.)

Von `circle` wird abgeleitet:

circle_co
Kreis ist definiert durch Mittelpunkt und Referenzobjekt, das den Radius bestimmt.

circle_r2o
Kreis ist durch Radius und zwei Tangentialobjekte bestimmt.

circle_3o
Kreis ist durch drei Tangentialobjekte bestimmt.

Von `arc` werden ähnliche Klassen abgeleitet wie von `circle`. Abbildung 2.1 zeigt ein Beispiel für die Beschreibung von Geometrie und Beziehungen durch Instanzen der Klassen:

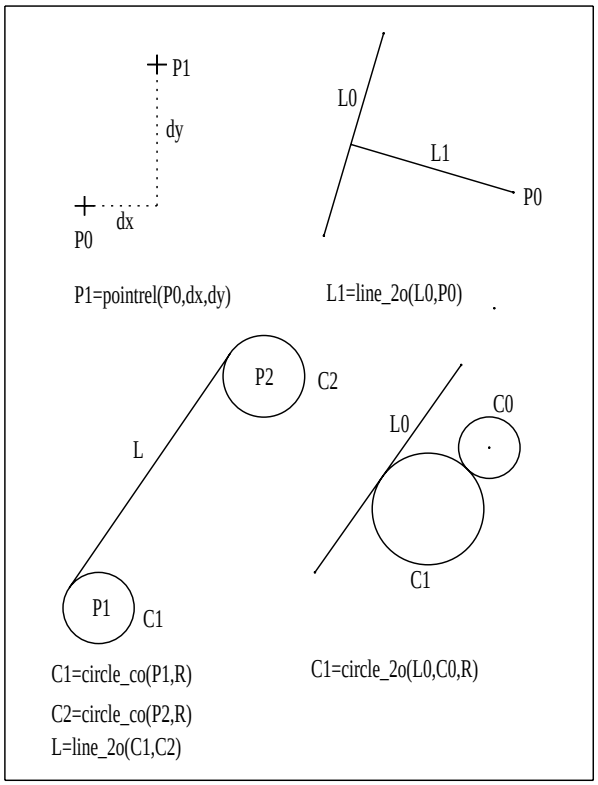


Abbildung 2.1. Geometrisches Modell

Da in manchen Fällen mehrere Lösungen existieren, beinhalten diese abgeleiteten Klassen noch ein Feld: `alter`, das in kanonischer Form die zu entwickelnde Lösung beschreibt. Dieses Feld wird von der Schnittstelle aufgrund der Auswahl des Benutzers belegt.

Die in den abgeleiteten Klassen referierten Objekte werden *support GMP* genannt. Von diesen *support GMP* wird jedoch nur jeweils die Information benutzt, die bereits in der entsprechenden absoluten Klasse enthalten ist. D. h. für die Berechnung eines `value_2P` spielt es keine Rolle von welchem Typ die referierten Punkte sind.

Die GMP einer geometrischen Konstruktion und ihre Bezüge untereinander bilden einen azyklischen Graphen (DAG) wie in Abbildung 2.2 gezeigt.

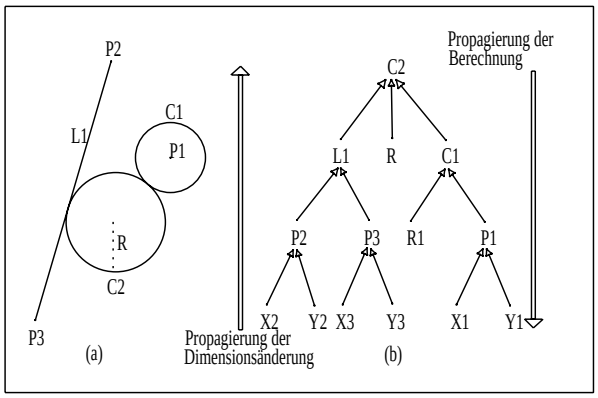


Abbildung 2.2. DAG zum geometrischen Modell

Um ein Element der Konstruktion zu berechnen existiert für jede Klasse eine Methode *compute*. Diese fordert zunächst die *support GMP* auf sich zu berechnen und berechnet aus den Werten der entsprechenden absoluten Klassen die Werte des Elements (z. B. Schnittpunkt zwischen zwei Linien). Die Blätter dieses DAG sind Instanzen der Basisklassen, da diese keine weiteren Bezüge haben. Sie werden *absolute Objekte* genannt und bilden die Parameter der Konstruktion. Aus Gründen der vereinfachten Implementation werden alle Konstruktionen auf absolute Werte zurückgeführt, d. h. absolute Punkte werden durch `pointref` mit Bezug auf zwei absolute Werte dargestellt.

2.2 Varianten einer Konstruktion

Durch Ändern der Werte an den Blättern (Parameter) und anschließender Neuberechnung kann die Konstruktion variiert werden.

In vielen Fällen wird jedoch gewünscht, daß ein oder mehrere abhängige Objekte vorgegebene Werte erreichen. Diese Berechnung kann nicht direkt ausgeführt werden, vielmehr werden in einem Iterationsverfahren freigegebene Parameter so geändert, daß die abhängigen Objekte die Zielwerte erreichen. Die Änderung der Koordinaten des Mittelpunkts des Kreises C2 in Abbildung 2.2 erfordert beispielsweise dieses Vorgehen.

Es sind nicht alle geometrischen Konstruktionen sequentiell konstruierbar bzw. nicht ohne komplizierte Hilfskonstruktionen.

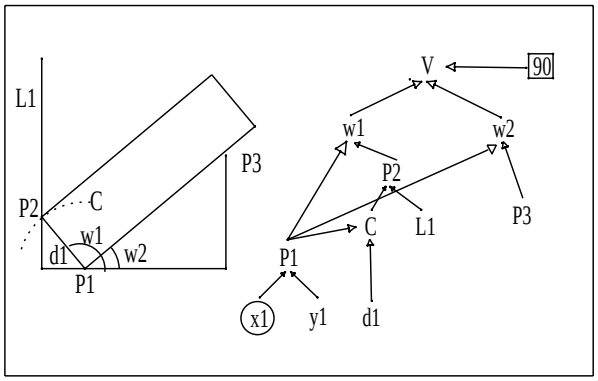


Abbildung 2.3. Nicht-sequentielle Konstruktion

Das Beispiel in Abbildung 2.3 zeigt ein Rechteck, das bei vorgegebener Größe in einer vorgegebenen Wanne plaziert werden soll. Sequentiell bestimmt werden kann eine Konstruktion, bei der der Winkel $v = w1 - w2$ nicht rechtwinklig sein muß. Der Punkt $P1$ kann dann auf der waagerechten Linie verschoben werden. Gesucht wird nun die x-Koordinate von $P1$, bei der der Winkel v den Wert 90° erreicht. Die Klasse `late-constraint` ermöglicht diese nachträgliche Spezifikation einer Beziehung zwischen Parameter und abhängigem `value`-Objekt.

Objekte der Klasse `late-constraint` halten folgende Information:

- Abhängige `value`-Objekte
- Zielwerte für diese Objekte
- Parameter, deren Werte geändert werden sollen, um die Zielwerte zu erreichen (*maintaining variable*).

Durch ein `late-constraint` mit *n* Werten verliert eine Konstruktion *n* Freiheitsgrade, da *n* Parameter nicht mehr frei gewählt werden können, sondern zur Erfüllung des `late-constraint` benutzt werden müssen. Da Zielwerte für abhängige Objekte in einem `late-constraint` jedoch frei gewählt werden können, können die Objekte, falls erwünscht, auch als Parameter benutzt werden. Im Beispiel könnte auch für einen Winkel von z. B. 80° eine Lösung gefunden werden. Für dieses Vorgehen ist es natürlich erforderlich, daß eine Änderung von Parameterwerten eine Änderung des abhängigen Objekts bewirkt. Eine notwendige Voraussetzung hierzu ist, daß im DAG eine Verbindung von der Variablen zum Wert besteht.

Mit Hilfe der `late-constraint` können alle Konstruktionen, die durch symmetrische Gleichungssysteme beschrieben werden können, auch durch GMP in Verbindung mit Objekten der Klasse `late-constraint` beschrieben werden. Das zur Lösung benötigte Gleichungssystem enthält die minimale Anzahl von Variablen, die zur Beschreibung erforderlich sind.

2.3 Verallgemeinertes Segmentkonzept

Das RelCAD–Modell mit seiner Möglichkeit der Beschreibung geometrischer Beziehungen kann zur Definition eines verallgemeinerten Segmentkonzepts (d. h. der Verwendbarkeit einer Konstruktion als Teil einer anderen Konstruktion) genutzt werden. Gängigerweise können bei normalen Segmenten Bezugskoordinate des Referenzpunktes, Ausrichtung und der Größenfaktor in *X* und *Y* bei der Instanziierung gewählt werden. Die Anzahl und der Typ der aktuellen Parameter ist im System fest vorgegeben.

Das Konzept im RelCAD–System hat als Vorbild das Prozedurkonzept der Programmiersprachen.

In einer geometrischen Konstruktion existieren als Blätter absolute GMP (d. h. Werte, Punkte, Linien, Kreise, Bögen). Diese absoluten GMP bilden die formalen Parameter des Segmentes.

Bei einer Instanz dieses Segmentes werden die formalen Parameter durch aktuelle Parameter ersetzt. Die aktuellen Parameter sind GMP aus der Konstruktion in die das Segment eingesetzt wird. Der Typ des aktuellen Parameters muß identisch oder abgeleitet vom Typ des formalen Parameters sein. Segmente können durchaus Segmente als lokale Elemente oder auch als Parameter enthalten.

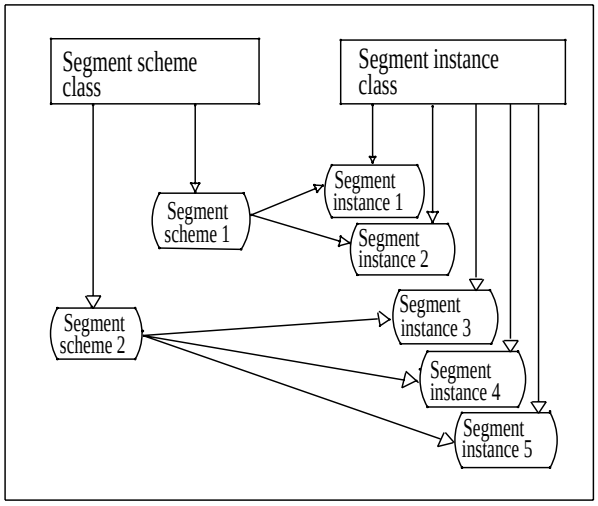


Abbildung 2.4. Segmentschema und Segmentinstanz

Eine Segmentdefinition ist ein Vertreter der Klasse *Segmentschema* analog zur Prozedurdeklaration. Die Benutzung eines Segmentes erfolgt durch die Klasse *Segmentinstanz*. Die Berechnung der lokalen Elemente einer Segmentinstanz kann ohne einen Stack erfolgen, da in dieser Anwendung rekursive Segmente, d. h. Segemente, die Segmentinstanzen vom gleichen Typ als Elemente enthalten nicht erlaubt und im allgemeinen auch nicht notwendig sind.

Die lokalen Elemente eines Segmentes können sowohl Bezüge auf Parameter und andere lokale Elemente als auch auf globale Elemente außerhalb des Segments enthalten, analog zu Programmiersprachen.

Im Gegensatz zu Prozeduren in Programmiersprachen ist es hier jedoch durchaus sinnvoll, auch von außen Bezüge auf lokale Elemente eines Segments zu erlauben, z. B. um zwei lokale Elemente in verschiedenen Segmenten durch eine Linie zu verbinden. Um dies zu ermöglichen, wird von jeder der absoluten GMP ein *substitute GMP* abgeleitet. Das *substitute GMP* enthält einen Pfad, der den Weg über die Segmentinstanzen zu dem lokalen Element beschreibt. Bei der Neuberechnung eines *substitute GMP* wird die im Pfad angegebene Segmentinstanz berechnet. Die Werte des lokalen Elementes werden dann für den Außenbezug übernommen.

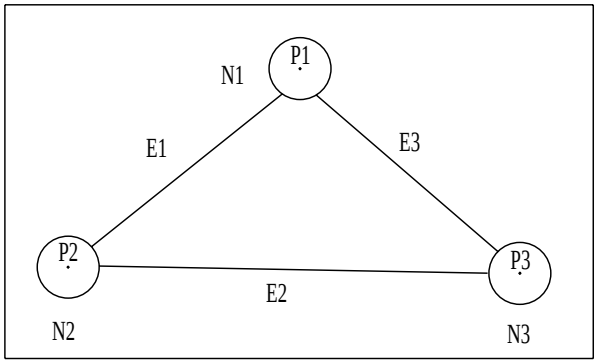


Abbildung 2.5. Beispiel für ein Segement

Segmentdefinition zum Beispiel in Abbildung 2.5:

```
segment:  node (P: point)
    value:  R;     {absolut}
    circlemr:  C (P,R);
segment:  edge (N1,N2:  node)
    C1:  circle_inst (N1,C);
         {substitute GMP}
    C2:  circle_inst (N2,C);
         {substitute GMP}
    L: line_2o (C1,C2);
segment:  trigraph (P1,P2,P3:  point)
    N1:  node (P1);
    N2:  node (P2);
    N3:  node (P3);
    E1:  edge (N1,N2);
    E2:  edge (N2,N3);
    E3:  edge (N3,N1);
```

Das Segment `node` enthält einen Kreis, dessen Radius aus dem lokalen Element `R` bestimmt wird und so für alle Exemplare von `node` gleich ist. Im Segment `edge` werden die Kreise in den beiden als Parameter übergebenen Segmenten `node` zur Konstruktion der Linie referenziert: `line_2o (C1,C2)`. Die korrekte Referenz erfolgt über die Pfadbeschreibung der substitute GMP.

Falls von außen keine Bezüge auf lokale Elemente existieren, können Segmentdefinitionen geändert werden, solange die Schnittstelle, d. h. die formalen Parameter gleichbleiben. Im obigen Beispiel könnte die Definition von `edge` geändert werden. Bei einer Änderung von `node` müßte jedoch der Kreis `C` erhalten bleiben, da in `edge` ein Bezug hierauf existiert.

3 Deklarative Modellierung

Im Gegensatz zum konstruktiven Ansatz beschreibt die deklarative Modellierung geometrische und nichtgeometrische Beziehungen durch Mengen von Gleichungen. Geometrische Beziehungen zwischen Linien, Kreisen und Bögen als den Elementen einer Konstruktion werden dabei auf wenige Basisrelationen zwischen Punkten und Dimensionen zurückgeführt.

3.1 Constraint-Modell

Das geometrische Modell konstituiert sich aus der Beschreibung der geometrischen Elemente und aus der Beschreibung geometrischer und nichtgeometrischer Beziehungen, die zwischen diesen Elementen wirken. Die geometrischen Elemente sind Punkt, Linie, Kreis und Bogen, die über die Parameter: Punkt und Radius definierbar sind:

$line(P_1, P_2)$
$circle(P, r)$
$arc(P_1, P_2, M, r)$

Zur Beschreibung aller relevanten geometrischen Beziehungen werden drei Typen von Basis-Constraints, *dimensionale Relationen* genannt, benötigt. Die Basis-Constraints wirken auf Parameter geometrischer Elemente und definieren Dimensionen: Abstände und Winkel:
Abstand d zwischen 2 Punkten P_1, P_2:

$dpp(P_1, P_2, d)$

Abstand d zwischen einem Punkt P und einer Geraden durch 2 Punkte P_a, P_e:

$dpl(P, P_a, P_e, d)$

Winkel α zwischen zwei Geraden durch die Punkte P_1, P_2 und P_3, P_4:

$ang(P_1, P_2, P_3, P_4, \alpha)$

Constraints zwischen geometrischen Elementen sind durch einen oder mehrere dieser Basis-Constraints ausdrückbar. Mit der Definition eines Bogens sind beispielsweise zwei Constraints verbunden, die den Abstand der Endpunkte zum Mittelpunkt auf die Größe des Radius festlegen:

$dpp(P_1, M, r), dpp(P_2, M, r)$

Abbildung 3.1 gibt ein Beispiel für eine Tangentenbeziehung zwischen Geraden und Kreisen.

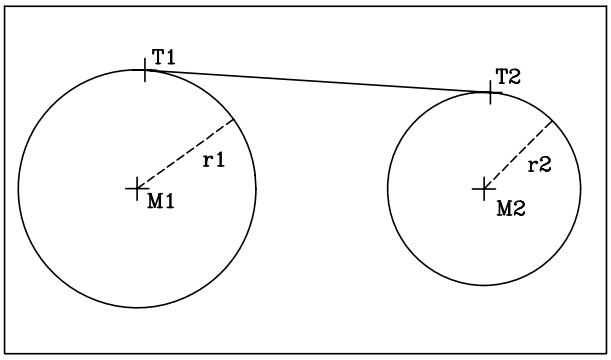


Abbildung 3.1. Gerade tangential an zwei Kreise

Das geometrische Modell ist durch folgende Mengen von Constraints und geometrischen Elementen gegeben:

$dpp(M_1, T_1, r_1)$
$dpp(M_2, T_2, r_2)$
$dpl(M_1, T_1, T_2, r_1)$
$dpl(M_2, T_2, T_1, r_2)$

$circle(M_1, r_1)$
$circle(M_2, r_2)$
$line(T_1, T_2)$

Nichtgeometrische Beziehungen können mit gleichen Mitteln modelliert werden. Es sind Relationen zwischen Dimensions-Variablen und anderen skalaren Größen. Zur Definition werden die betroffenen Dimensionsvariablen durch entsprechende Basis-Constraints dargestellt, bevor eine passende *skalare* Relation die nichtgeometrische Beziehung beschreibt. Soll beispielsweise in Abbildung 3.1

die Länge der Linie in Beziehung stehen zur Summe der Radien, so wird die Constraint-Menge um folgende Relationen erweitert:

$dpp(T_1, T_2, d)$
$r(d, r_1, r_2, f)$ $\{d = (r_1 + r_2) * f\}$

Zur Vermeidung iterativer Methoden bei der Evaluierung einzelner nichtgeometrischer Constraints müssen für jeden Constraint Funktionen definiert sein, die jede Variable aus den anderen berechnen können.

3.2 Constraint-Graph

Ein Constraint-System wird durch einen ungerichteten, bipartiten Graphen $G_C(O, C, E)$ beschrieben. Die Knotenmengen $O = P \cup D$ und C repräsentieren die Punkte (P), Dimensionen bzw. skalaren Größen (D) und die Menge der Constraints (C). Die Kanten in E drücken die Constraint-Bindungen an die Variablenknoten in O aus. Abbildung 3.2 zeigt den Constraint-Graphen zur Konstruktion aus Abbildung 3.1.

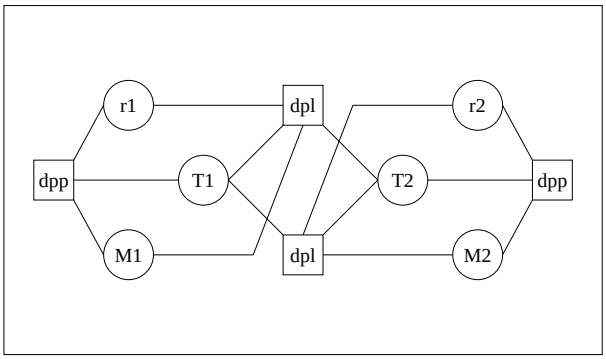


Abbildung 3.2. Ein Constraint-Graph

Die Verwendung dimensionaler Relationen zur Beschreibung geometrischer Constraints hat mehrere Vorteile. Die Beschränkung auf wenige Basis-Constraints hat beispielsweise zur Folge, das Implementierungen der Evaluierung besser optimierbar sind. Der eigentliche Vorteil liegt aber darin, daß bei der Verwendung von lokaler Propagierung als Constraint-Satisfaction-Verfahren weniger Kreise im Graphen entstehen. Punkte werden im Graphen durch einzelne Knoten repräsentiert. Sie brauchen zu ihrer Berechnung zwei Constraints, da sie zweidimensionale Werte sind. Die Evaluierung einer dimensionalen Relation in einem ihrer Punkt-Attribute besitzt als Lösung eine Geometrie: im zweidimensionalen Fall eine Gerade oder einen Kreis. Punkte sind also als Schnittpunkte zwischen bekannten Geometrien berechenbar. Eine Verwendung anderer Constraint-Gleichungen zur Beschreibung geometrischer Beziehungen erfordert eine Verteilung der x- und y-Koordinaten eines Punktes auf verschiedene Knoten im Graphen. Bei Verwendung von lokaler Propagierung führte dies zu einer größeren Anzahl gerichteter Kreise im Graphen, die mit iterativen Methoden zu berechnen wären, da es sich bei den Constraints um nichtlineare Gleichungen handelt.

3.3 Constraint Satisfaction

Der Constraint-Graph repräsentiert bei einer nicht gebundenen Variablenmenge ein unterbestimmtes Gleichungssystem. Mit jeder Änderung einer Konstruktion wird eine Teilmenge der Variablen mit Werten belegt und der Constraint-Satisfaction-Prozeß gestartet. Aufgabe eines Algorithmus ist es, das Gleichungssystem in mehreren Schritten in ein vollständig bestimmtes System zu überführen und eine Zerlegung dieses Systems in eine Anzahl untereinander unabhängiger Teilmengen von Gleichungen zu berechnen. Unter Umständen müssen dabei bei weiterhin unterbestimmten Systemen weitere Werte "von außen" angefordert werden. Lokale Propagierung und Freiheitsgrad-Analyse sind die Techniken, auf denen der Constraint-Satisfaction-Prozeß aufbaut.

3.3.1 Lokale Propagierung

Durch den Constraint-Satisfaction-Prozeß wird das unterbestimmte Gleichungssystem in ein bestimmtes überführt. Parallel dazu wird eine Zerlegung des Systems in eine Sequenz von Gleichungen und kleineren Gleichungssystemen berechnet. Die Berechnung dieser Zerlegung entspricht der Berechnung einer Kantenorientierung auf dem ungerichteten Graphen, die definiert, welche Constraints welche Variablen berechnen sollen. Die Reihenfolge der Evaluierung ist durch die topologische Sortierung der Knoten[1] des gerichteten Graphen definiert. Gerichtete Kreise repräsentieren Mengen von Constraints, die als Gleichungssysteme auszuwerten sind. Da die Berechnung nichtlinearer Gleichungssysteme iterative Methoden erfordert, die bekanntermaßen nicht global konvergent sind, ist das Ziel der Kantenorientierung einen gerichteten, nach Möglichkeit kreisfreien Graphen zu berechnen.

Der orientierte Graph muß die folgenden Bedingungen bezüglich der Kanteninzidenzen erfüllen:

$\forall\, v \in P : \delta^-(v) = 2$
$\forall\, v \in D : \delta^-(v) = 1$
$\forall\, v \in C : \delta^+(v) = 1$

Die Kantenrichtung zeigt die Richtung der Evaluierung von Constraints bzw. die Richtung der Verbreitung von Variablenwerten an. Jeder Punkt benötigt zwei Constraints zu seiner Bestimmung, jede skalare Größe einen Constraint. Ist der Wert einer Variablen bestimmt, wird er durch Orientierung der mit dem Knoten inzidenten ungerichteten Kanten zu den adjazenten Constraints propagiert.

Jedes Constraint-System besitzt einen Freiheitsgrad, der ausdrückt, wieviele Werte von Variablen nicht durch Constraints berechenbar sind, sondern "von außen" festgelegt werden müssen. Der Freiheitsgrad (DoF[2]) berechnet sich wie folgt:

$\forall p \in P : DoF(p) = 2 - \delta^-(p)$
$\forall d \in D : DoF(d) = 1 - \delta^-(d)$
$\forall N \subseteq O : DoF(N) = \sum_{\forall v \in N} DoF(v)$

Zur Berechnung des Freiheitsgrads eines Constraint-Graphen wird für einem Constraint-Knoten ebenfalls ein Freiheitsgrad definiert:

$\forall c \in C : DoF(c) = \delta^+(c) - 1$
$\forall U \subseteq C : DoF(U) = \sum_{\forall v \in U} DoF(v)$

Somit ist:

[1] genaugenommen durch die topologische Sortierung der starken Komponenten (siehe 3.3.2)

[2] Degree of Freedom

$\forall G_C(O, C, E)$:
$DoF(G_C) = DoF(O) + DoF(C)$

Der Algorithmus, der die lokale Propagierung bekannter Werte implementiert, benutzt den Freiheitsgrad für die Auswahl der zu propagierenden Werte.

solve (G_C)
  if $DoF(G_C) > 0$ **then**
    get $N \subset O_v : DoF(N) \leq DoF(G_C)$[3]
    propagate N
      `--durch Orientierung der Kanten`
      `--Resultat: Menge` H `von ungerichteten Teilgraphen von` G_C
    for all $g \in H$ **do**
      solve (g)
  else
    compute_matching (G_C)
      `--Resultat: Graph mit gerichteten Kreisen`
    evaluate (G_C)
      `--Berechnung durch iterative Methoden`

Mit jeder Iteration von $solve()$ werden freie Variablen in den N_i gebunden und deren Werte propagiert. Die Auswahl der Variablen wird durch den Benutzer festgelegt, sie wird durch den verbliebenen Freiheitsgrad gesteuert. Die Operation **propagate** realisiert die Kantenorientierung. Soweit neue Variablen dadurch bestimmt werden, wird die Berechnung der Werte hier durchgeführt. Bei Vorliegen von mehreren Lösungswerten wird der passende Wert durch Heuristiken bestimmt. In **propagate** wird auch eine Konsistenzprüfung vorgenommen, die frühestmöglich die Constraints identifiziert, die bezüglich propagierter und berechneter Werte inkonsistent und somit nicht lösbar sind.

Ist $DoF(G_C)$ auf 0 reduziert, so können trotzdem noch ungerichtete Teilgraphen existieren, die nicht mehr kreisfrei orientierbar sind. Die Operationen **compute_matching** und **evaluate** realisieren Verfahren zur Evaluierung von Graphen G_C mit $DoF(G_C) = 0$. Sie sind in 3.3.2 näher beschrieben.

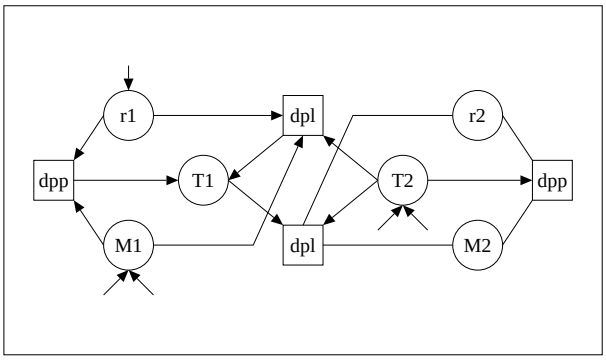


Abbildung 3.3. Lokale Propagierung

Die Abbildung 3.3 zeigt den Graphen aus Abbildung 3.2 mit Werten für $N_1 = \{M_1, r_1, T_2\}$ nach einem ersten Propagierungsschritt. Für den verbleibenden Teilgraphen gilt $DoF = 1$. Mit $N_2 = \{r_2\}$ folgt der zweite und letzte Propagierungsschritt (Abbildung 3.4).

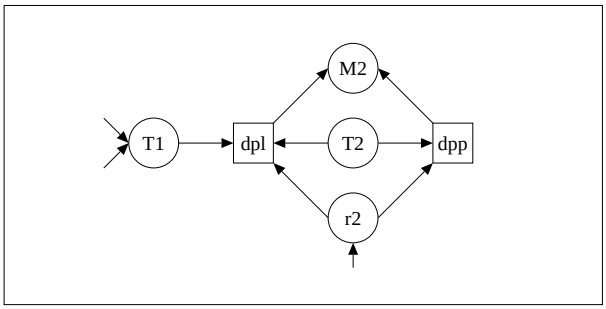


Abbildung 3.4. Lokale Propagierung fortgesetzt

Die Propagierungsschleife terminiert, wenn alle Kanten orientiert sind oder die Propagierung zu ungerichteten Teilgraphen führt, die ein DoF von 0 haben.

Zum oben aufgeführten Algorithmus gibt es nach einer ersten Propagierung bekannter Werte für die weitere Reduktion des Graphen eine Alternative: Propagierung von Freiheitsgraden. Falls die Anzahl ungerichteter Kanten an einem Variablen-Knoten kleiner oder gleich seinem Freiheitsgrad ist, können die Kanten zur Variable hin orientiert werden. Im Beispiel aus Abbildung 3.3 führt die Propagierung von Freiheitsgraden zu folgendem Vorgehen. Nach der Propagierung von N_1 werden die Freiheitsgrade und die Anzahl der inzidenten, ungerichteten Kanten der verbliebenen Knoten bestimmt. Der Knoten M_2 wird zur Propagierung ausgewählt, da $\delta(M_2) \leq DoF(M_2)$. Das Verfahren resultiert im gleichen Graphen wie in Abbildung 3.4 dargestellt. Das Verfahren hat gegenüber einer weiteren Propagierung bekannter Werte den Vorteil, daß im allgemeinen kleinere Restgraphen mit einem Freiheitsgrad von 0 übrigbleiben.

Aus der Konstruktionsdefinition und aus Konstruktionsänderungen liegen vor jeder Anwendung eines Constraint-Satisfaction-Verfahrens konstante und bekannte Werte für eine Teilmenge der Knoten vor. Eine initiale Propagierung bekannter Werte ist daher in jedem Fall nötig, da der Graph um diese Konstanten reduziert werden muß, bevor eine Propagierung von verbliebenen Freiheitsgraden stattfinden kann.

3.3.2 Auflösung von Kreisen im Graphen

Ungerichtete Teilgraphen mit einem DoF von 0 sind nicht kreisfrei orientierbar. Ihre Entstehung ist abhängig von der Wahl der Variablen für die N-Mengen. Abbildung 3.5 zeigt beispielsweise den nicht kreisfrei orientierbaren Graphen, der bei Wahl von M_2 für N im zweiten Schritt des oben genannten Beispiels entsteht. Für M_2 ist im Beispiel eine Koordinate gegeben, die allerdings nicht propagiert werden kann. Der Restgraph über M_2, r_2 besitzt einen Freiheitsgrad von 0 und ist nicht kreisfrei orientierbar, da M_2 entweder über dpp oder über dpl bestimmt werden muß.

[3] O_v ist die Menge der Variablen in G_C, deren Werte noch unbekannt sind.

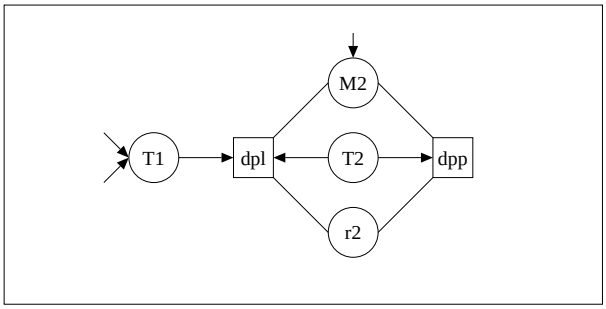


Abbildung 3.5. Propagierung die zum Kreis führt.

Es ist nachweisbar, daß gerichtete Kreise auf verbliebene, ungerichtete Teilgraphen mit $DoF = 0$ beschränkt sind, also nicht durch die bereits orientierten Graphen führen können. Dem Beweis liegt zugrunde, daß die lokale Propagierung Kanten immer in Richtung des noch nicht orientierten Graphen ausrichtet. Der Rand der ungerichteten Teilgraphen zu den gerichteten Graphen besteht aus Kanten die zum ungerichteten hin orientiert sind.

Zur Evaluierung der Constraints in den Teilgraphen mit $DoF = 0$ wird das im folgenden beschriebene Verfahren eingesetzt. Auf dem verbliebenen ungerichteten Graphen wird eine Orientierung der Kanten durch Berechnung eines maximalen Matchings erreicht ([McHu90]). Mit Hilfe einer Berechnung und topologischen Sortierung der starken Komponenten können die unter dieser Kantenorientierung unabhängigen Gleichungssysteme identifiziert werden. Jede starke Komponente entspricht einem simultan zu lösenden Gleichungssystem. Die topologische Ordnung definiert die Reihenfolge, in der sie zu lösen sind. Für die Berechnung der Gleichungssysteme muß eine numerisch-iterative Methode verwendet werden, beispielsweise Newton-Raphson. Anwendungen graphbasierter Techniken dieser Art für die Zerlegung und Lösung von Gleichungssystemen finden sich auch in [SeGo87, SoBuEr86].

3.4 Abschließende Bemerkungen zu konstruktiver und deklarativer Modellierung

Das hier vorgestellte Modell ist wie auch der Ansatz aus dem vorigen Kapitel in erster Linie zur Modellierung geometrischer Relationen konzipiert. Während dort eine integrierte Modellierung von Objekten und Beziehungen im Vordergrund stehen, zeichnet sich dieses Modell durch eine explizite Trennung beider Komponenten aus. Wie bereits dargestellt, besitzt diese Trennung Vorteile bezüglich der Konzeption eines effizienten Constraint-Satisfaction-Verfahrens.

Das in Kapitel zwei vorgestellte Modell hat seine Vorteile in der effizienten Berechenbarkeit von Konstruktionsänderungen durch Änderung von Parameterwerten. Durch die direkte Modellierung der Konstruktionsform über die verschiedenen Klassen des geometrischen Modells kann eine Konstruktionsänderung ohne einen aufwendigen Constraint-Satisfaction-Mechanismus nachgeführt werden. Die Umsetzung von der interaktiven Spezifikation eines Entwurfs in das Modell der Objekte und Beziehungen erfolgt unmittelbar. Der Ansatz eignet sich daher im besonderen für parametrische Entwurfsaufgaben, in denen aus einer generischen Konstruktion verschiedene Varianten abzuleiten sind. Das verallgemeinerte Segmentkonzept ist ein Beispiel dafür.

Das in diesem Kapitel vorgestellte Modell hat seine Vorteile im Ausdruck geometrischer Beziehungen durch wenige Basis-Constraints, was eine Trennung von Objekt- und Constraint-Repräsentation erforderlich macht. Der Ansatz erlaubt ein effektiveres Constraint-Satisfaction-Verfahren, da im Gegensatz zum Modell aus Kapitel zwei die durch die Constraints gegebenen Abhängigkeiten zwischen den Parametern der geometrischen Elemente besser analysierbar sind. Die deklarative Modellierung erfordert nicht die Notwendigkeit der sequentiellen Konstruierbarkeit; die Menge der mit diesem Modell beschreibbaren Konstruktionen ist daher größer. Repräsentationen im Ansatz aus Kapitel zwei lassen sich direkt durch gerichtete, kreisfreie Modelle dieses Ansatzes ausdrücken. Die graph-basierte Repräsentation erfordert ein komplizierteres Constraint-Satisfaction-Verfahren, sie läßt aber den Einsatz aller im Bereich der Constraint-Modellierung entwickelten Verfahren zu. Der Ansatz eignet sich wegen des allgemeinen Constraint-Satisfaction-Algorithmus im besonderen für die Analyse von Entwürfen in frühen Phasen in denen es noch keine ausgezeichneten Parametermengen gibt über die eine Variation eines Entwurfs bestimmt wird.

4 Constraint-Verarbeitung bei endlichen Wertebereichen

4.1 Das “Constraint-Satisfaction-Problem”

Forschung auf dem Gebiet der Graphik beinhaltet immer stärker Elemente aus dem Bereich “Künstliche Intelligenz” (KI). In diesem Abschnitt wird eine ganz spezielle Technik zur Wissensrepräsentation aus der KI beleuchtet und ihre Einsatzfähigkeit für CAD aufgezeigt: *constraint processing*. Diese Herangehensweise kann man ebenso wie die im vorigen Abschnitt dargestellte Methode als deklarativ bezeichnen, sie hat aber nicht deren sequentiell-konstruktiven Charakter. Constraint-Verarbeitung im allgemeinen kann dazu dienen, alle möglichen Lösungen, welche gewissen Bedingungen (“Constraints”) innerhalb eines vorgegebenen Gestaltungsrahmens genügen, zu berechnen; hier wird beispielsweise versucht, alle möglichen Layout-Konfigurationen dem Benutzer anzubieten. Das “constraint satisfaction problem” (CSP) besteht nun darin, bei n vorgegebenen Variablen mit ihren endlichen Wertebereichen und vorliegenden Bedingungen an die Belegung gewisser Variablen mit Werten die Menge aller zugelassenen n-Tupel zu bestimmen. [HoRo93] gibt einen kleinen Einblick in die Komplexität der Berechnung eines CSP.

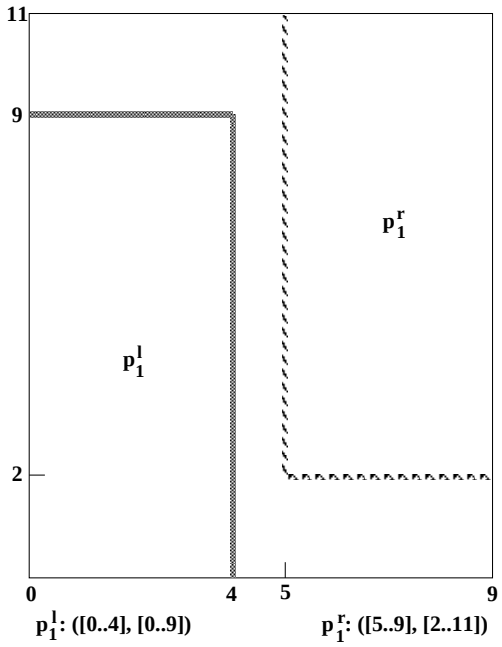
4.2 Anpassung an CAD-Erfordernisse

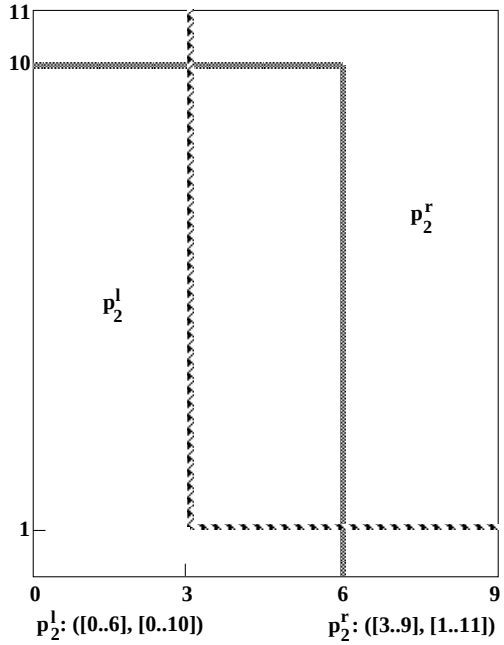
Der Ansatz, zunächst ein allgemeines Rahmenverfahren auszuprobieren, welches letztlich die global konsistente Lösung bereithält, wurde auch durch die Tatsache motiviert, daß bereits Untersuchungen hinsichtlich der Möglichkeit des intelligenten Aktualisierens eines Constraint-Netzwerkes gemacht wurden. Dabei geht es

darum, bei evtl. nur geringen Änderungen an die Spezifikation eines Problems nur solche Neu-Berechnungen durchzuführen, welche auch wirklich gebraucht werden, um die neue globale Lösung zu berechnen. Man betrachtet also nur diejenigen Teile nochmal, welche von den Modifikationen gegenüber des vorherigen CSP betroffen sind. Dieses "Merkmal der kleinsten Änderung" wird sehr oft als gewünschtes Element in vielen verwandten Artikeln genannt. Um nun das Kriterium der endlichen Beschreibbarkeit zu erfüllen, werden die Werte einer Variablen zu einem Intervall zusammengefaßt, um danach nur noch die Intervall-Grenzen zu betrachten. Dabei wird nicht mit herkömmlicher Intervall-Arithmetik hantiert, sondern lediglich die Grenzzahlen der Intervalle neu bestimmt. Sogesehen spielt es auch keine Rolle, ob man auf einem echt-kontinuierlichen Wertebereich arbeitet oder nur ganze Zahlen zuläßt; dies bleibt der jeweiligen Implementierung überlassen. Die Intervall-Grenzen stellen somit die Flächenbegrenzungen dar. Dieser Ansatz wird einem Diskretisieren der Wertebereiche vorgezogen, da sonst zuviele Einzelelemente abgeprüft werden müßten; ohnehin erlaubt das hier gewählte Vorgehen ein Abstrahieren von der Größe der Wertebereiche.

4.3 Derzeitiger Ansatz zur Constraint-Verarbeitung

Zur Illustrierung dieser Technik wird folgendes Beispiel-Problem beleuchtet: gegeben sei ein größeres Rechteck und mehrere kleinere; gesucht sind alle möglichen Positionen der kleineren Rechtecke, um im großen Rechteck (überschneidungsfrei) Platz zu finden. Hierzu wird je eine zwei-dimensionale Variable für jedes kleinere Rechteck eingeführt, um sowohl die Möglichkeiten für die Positionierung des linken unteren Punktes (P^l) als auch des rechten oberen Punktes (P^r) zu verwalten. Für beides werden Bereichs-Angaben entlang der X- und Y-Koordinaten bereitgestellt.





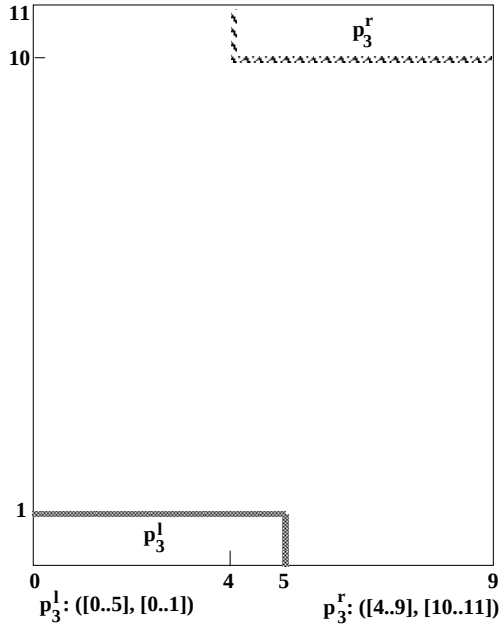


Abbildung 4.1. Initiale Intervalle

Sei zum Beispiel ein Zielrechteck der Größe 9×11 gegeben, welches in folgendem Koordinatenrahmen resultiert:

$P^l\colon (0,0),\ P^r\colon (9,11).$

Seien desweiteren drei kleinere Arbeitsrechtecke r_1, r_2, r_3 gegeben mit folgenden Angaben für Breite (b) u. Höhe (h):

$b_1 := 5,\ h_1 := 2;$
$b_2 := 3,\ h_2 := 1;$
$b_3 := 4,\ h_3 := 10.$

Zur Neu-Berechnung der ein-stelligen Constraints werden nun die jeweiligen Wertebereiche der Arbeitsrechtecke aufgrund der vorliegenden Angaben in Bezug auf die Größe des Zielrechtecks zusammen mit den eigenen Maßangaben bzgl. Breite und Höhe initialisiert. Den Eckpunkten der einzelnen Objekte bleibt somit zur Zeit folgender Spielraum, der in intervall-ähnlicher Notation festgehalten wird (Darstellung in Abbildung 4.1):[4]

[4] Natürlich braucht man die jeweilige obere Ecke nicht mit zu verwalten (und es wird eigentlich auch nicht gemacht); dies wird nur zur bequemeren Veranschaulichung hier mit aufgenommen.

P_1^l: $([0..4],[0..9])$, P_1^r: $([5..9],[2..11])$;

P_2^l: $([0..6],[0..10])$, P_2^r: $([3..9],[1..11])$;

P_3^l: $([0..5],[0..1])$, P_3^r: $([4..9],[10..11])$.

Das Verfahren fährt nun mit der Berechnung der zweistelligen Constraints fort, indem es die verschiedenen Möglichkeiten zur Plazierung von jeweils zwei Rechtecken abprüft: ob es beispielsweise möglich ist für ein bestimmtes Rechteck links (oder rechts) von einem anderen zu liegen oder über (oder unter) einem. Wir haben also in Bezug auf die beiden Rechtecke r_1 und r_2 folgende (mit "oder" verbundene) Möglichkeiten:

$x_{P_1^r} \leq x_{P_2^l}$ (r_1 ist links von r_2)
$x_{P_2^r} \leq x_{P_1^l}$ (r_2 ist links von r_1)
$y_{P_1^r} \leq y_{P_2^l}$ (r_1 ist unterhalb von r_2)
$y_{P_2^r} \leq y_{P_1^l}$ (r_2 ist unterhalb von r_1).

Zur Neu-Berechnung der Intervallgrenzen werden folgende Berechnungsvorschriften eingesetzt, wobei hier auch deutlich werden soll, daß in der Tat nur der linke untere Punkt eines Rechteckes verwaltet wird:

if $min_{x_1} + b_1 > min_{x_2}$ then
$min_{x_2} := min_{x_1} + b_1$;

if $max_{x_1} > max_{x_2} - b_1$ then
$max_{x_1} := max_{x_2} - b_1$,

welche bei den ersten beiden Fällen jeweils angewandt werden, während die folgenden beiden Vorschriften jeweils in den beiden letztgenannten Fällen Verwendung finden:

if $min_{y_1} + h_1 > min_{y_2}$ then
$min_{y_2} := min_{y_1} + h_1$;

if $max_{y_1} > max_{y_2} - h_1$ then
$max_{y_1} := max_{y_2} - h_1$.

Als Beispiel wird hier nun die Neu-Berechnung der Intervallgrenzen für die erst- und die letztgenannte Relation gezeigt. Zum Test von $x_{P_1^r} \leq x_{P_2^l}$ ergeben sich demnach folgende Intervall-Angaben:

P_1^l: $([0..1],[0..9])$, P_1^r: $([5..6],[2..11])$;

P_2^l: $([5..6],[0..10])$, P_2^r: $([8..9],[1..11])$.

Bei $y_{P_2^r} \leq y_{P_1^l}$ ergibt sich:

P_1^l: $([0..4],[1..9])$, P_1^r: $([5..9],[3..11])$;

P_2^l: $([0..6],[0..8])$, P_2^r: $([3..9],[1..9])$.

In Bezug auf die beiden Rechtecke r_1 und r_3 werden die Intervallgrenzen beim Test von $x_{P_1^r} \leq x_{P_3^l}$ wie folgt berechnet:

P_1^l: $([0..0],[0..9])$, P_1^r: $([5..5],[2..11])$;

P_3^l: $([5..5],[0..1])$, P_3^r: $([9..9],[10..11])$.

Bezogen auf r_2 und r_3 bringt der Test von $y_{P_2^r} \leq y_{P_3^l}$ folgende Werte:

P_2^l: $([0..6],[0..0])$, P_2^r: $([3..9],[1..1])$;

P_3^l: $([0..5],[1..1])$, P_3^r: $([4..9],[11..11])$.

Im letzten Schritt wird schließlich abgeprüft, ob (alle) drei Rechtecke (und wenn ja, in welchen Konstellationen) ins große Rechteck passen. Dabei werden dann die entsprechenden Intervalle "geschnitten" und nach Konsistenz überprüft. Insgesamt werden hier 44 "abstrakte" Lösungen berechnet, wobei jede abstrakte Lösung mehrere Einzellösungen darstellt und auch Vertauschungen von Breite und Höhe möglich sind. Zum Beispiel taucht folgende abstrakte Lösung auf, welche diesen Lösungsraum wie folgt beschreibt (Darstellung in Abbildung 4.2):

P_1^l: $([0..0],[1..9])$, P_1^r: $([5..5],[3..11])$;

P_2^l: $([0..6],[0..0])$, P_2^r: $([3..9],[1..1])$;

P_3^l: $([5..5],[1..1])$, P_3^r: $([9..9],[11..11])$

und $x_{P_1^r} \leq x_{P_3^l}$ (r_1 ist links von r_3)
und $y_{P_2^r} \leq y_{P_3^l}$ (r_2 ist unterhalb von r_3)
und $y_{P_2^r} \leq y_{P_1^l}$ (r_2 ist unterhalb von r_1).

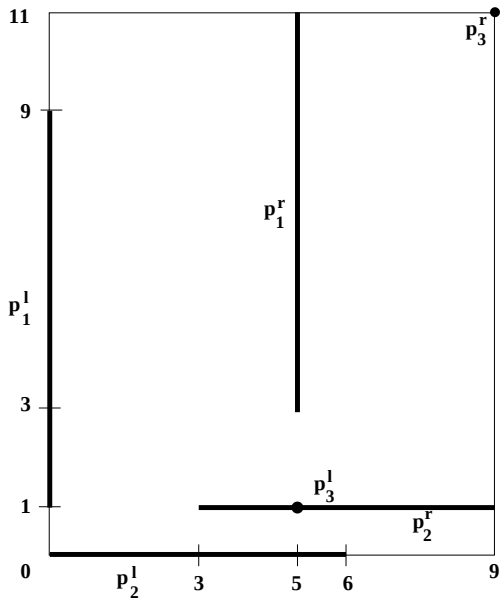


Abbildung 4.2. Eine abstrakte Lösung

Das Berechnungsverfahren arbeitet nur auf den Intervall-Grenzen, um ein akzeptables Verarbeiten zu ermöglichen. Der Algorithmus (siehe [Howe92]) kann beliebig-stellige Constraints verarbeiten; sind höher-stellige Constraints gegeben, so ist im allgemeinen noch eine Vorverarbeitung empfehlenswert, welche bereits zu Anfang die Grenzen der initialen Wertebereichs-Intervalle aktualisiert und somit frühzeitig eine Inkonsistenz erkannt werden kann. Wäre etwa im hier diskutierten Beispiel am Anfang gefordert worden, daß r_1 unterhalb von r_3 liegen soll, so wäre u.a. die Untergrenze des y-Intervalls des linken unteren Punktes von r_3 auf den Wert 2 hochgesetzt worden (P_3^l: $([0..5],[2..1])$), was schon in diesem Stadium zum Erkennen der globalen Inkonsistenz führt.

4.4 Abschließende Bemerkungen

Es gibt bestimmt bereits spezielle Routinen zur Lösung des obigen Beispiel-Problems. Die Motivation hinter der hier dargestellten Idee ist es zunächst einmal die Einsatzmöglichkeit eines globalen Constraint-Lösers im CAD-Bereich auszuloten. [HoRoBe93] stellte die Herangehensweise kurz vor; dort finden sich auch einige Literaturhinweise zu "verwandten" Arbeiten. Intelligente Heuristiken sollten üblicherweise in den Algorithmus eingebaut werden, um problem-abhängige Spezialitäten auszunutzen. Auch ist das genaue Betrachten der jeweiligen Constraint-Netzwerk-Strukur von Vorteil.

Der hier präsentierte Ansatz mag als ein weiterer explorativer Schritt in Richtung neuer Architekturkonzepte zur Gestaltung graphischer Systeme dienen.

Literatur

[AlMaRi91] Aldefeld B.; Malberg H.; Richter H.; Voss K. *Rule-Based Variational Geometry in Computer-Aided Design* In Pham D.T. (Hrsg.), *Artificial Intelligence in Design*; Springer-Verlag, 1991.

[ArWa91] Arbab F.; Wang B. *A Geometric Constraint Management System in Oar* In Hagen ten P.J.W.; Veerkamp P.J. (Hrsg.), *Intelligent CAD III, Practical Experience and Evaluation*, Seite 205–231; Springer-Verlag, 1991.

[BeDuRo91] Berling R.; Du C.; Rosendahl M. *A Relational CAD System with an Object Oriented Design* Forschungsbericht 10/91, Universität Koblenz, Institut für Informatik, 1991.

[BeRo92] Berling R.; Rosendahl M. *Zur Modellierung von Invarianten auf Geometriekonstruktionen* In Krause F.-L.; Ruland D.; Jansen H. (Hrsg.), *CAD'92 Neue Konzepte zur Realisierung anwendungsorientierter CAD-Systeme*, Seite 345–360; Springer-Verlag, 1992.

[BeRo93] Berling R.; Rosendahl M. *Geometry Modelling Using Dimensional Constraints* In CARs & FOF '93, 9th International Conference on CAD/CAM, Robotics & Factories of the Future, Newark, New Jersey, USA, August 18-20, 1993.

[ChSc89] Chung J.C.H.; Schussel M.D. *Comparison of Variational and Parametric Design* In Proceedings of AUTOFACT-89, Detroit, Michigan, USA, Seite 5–27 – 5–44, Oct. 30 – Nov. 2, 1989.

[DuRoBe93] Du C.; Rosendahl M.; Berling R. *Variation of Geometry and Parametric Design* In Tang Zesheng (Hrsg.), *New Advances in Computer Aided Design & Computer Graphics*, Seite 400–405, Beijing, China, August 23–26, 1993; International Academic Publishers.

[Freu78] Freuder Eugene C. *Synthesizing Constraint Expressions* Communications of the ACM, 21(11):958–966, November 1978.

[HoRo93] Hower Walter; Rosendahl Manfred *Notes on complexity issues within constraint satisfaction* In IJCAI-93 Workshop on Knowledge-based Production Planning, Scheduling and Control, Seite 179–186, Chambéry, Savoie, France, August 29, 1993.

[HoRoBe93] Hower Walter; Rosendahl Manfred; Berling Roland *Constraint Processing in Human-Computer Interaction with an Emphasis on Intelligent CAD* In Smith Michael J.; Salvendy Gavriel (Hrsg.), *Human-Computer Interaction: Applications and Case Studies; Proceedings of the Fifth International Conference on Human-Computer Interaction (HCI International '93, Orlando, Florida, USA)*, Seite 243–248, Amsterdam, The Netherlands, August 1993; Elsevier Science Publishers.

[Howe92] Hower Walter *Constraint satisfaction via partially parallel propagation steps* In Fronhöfer Bertram; Wrightson Graham (Hrsg.), *Parallelization in Inference Systems*, Seite 234–242; Lecture Notes in Artificial Intelligence, Subseries of Lecture Notes in Computer Science, Volume 590, Springer-Verlag, Berlin/Heidelberg, 1992.

[LiGo82] Light R. D.; Gossard D. *Modification of Geometric Models through Variational Geometry* CAD, 14(4), Juli 1982.

[Mack92] Mackworth Alan K. *Constraint Satisfaction* In Shapiro Stuart C. (Hrsg.), *Encyclopedia of Artificial Intelligence*, Seite 285–293; John Wiley & Sons, 1992.

[McHu90] McHugh J.A. *Algorithmic graph theory* Prentice-Hall, 1990.

[Owen91] Owen J.C. *Algebraic Solution for Geometry from Dimensional Constraints* In Proceedings of the Symp. on Solid Modeling Foundations and CAD/CAM Applications, Seite 397–407, Austin, Texas, USA, 1991; ACM.

[Roll91] Roller D. *Advanced Methods for Parametric Design* In Hagen ; Roller (Hrsg.), *Geometric Modeling: Methods and Applications*; Springer, 1991.

[SeGo87] Serrano D.; Gossard D. *Constraint Management in Conceptual Design* In Sriram D.; Adey R. A. (Hrsg.), *Knowledge Based Expert Systems in Engineering: Planning and Design*, Seite 211–224; Computational Mechanics Publications, 1987.

[SoBuEr86] Sowell E.F.; Buhl W.F.; Erdem A.E.; Winkelmann F.C. *A Prototype Object-based System for HVAC Simulation* In Proceedings of the 2nd Int. Conf. on System Simulation in Buildings, Liège, Belgium, Dezember 1986.

[VeScRo92] Verroust A.; Schonek F.; Roller D. *Rule-oriented Method for Parameterized Computer-Aided Design* CAD, 24(10), Oktober 1992.