Potenzkategorien
und ihre
Anwendung auf den Beweis von Programmeigenschaften

Herbert A. Klaeren
Universität Bonn

Zusammenfassung

Es bietet sich an, Programmeigenschaften kategorientheoretisch zu behandeln, da Programme in Flußdiagrammform als Basen von freien Kategorien betrachtet werden können. Die freie Kategorie über diesem Programm enthält als Morphismenmenge gerade alle möglichen Kontrollflüsse durch das Programm; diese repräsentieren alle möglichen Programmläufe. Aufgrund der speziellen Form der freien Kategorie (eindeutige Dekomponierbarkeit!) lassen sich Aussagen über Programmläufe allein durch Beweise über die Instruktionen erzielen, ohne daß man noch induktive Argumente hinzuziehen muß. Floyd's inductive assertion method läßt sich auf diese Weise kategorientheoretisch beweisen.

In dieser Arbeit werden die Potenzkategorien neu definiert; in ihnen sind Schnitte und Vereinigungen von Morphismen definiert. Mit ihrer Hilfe lassen sich Berechnungsverhalten bzw. Ein/Ausgabeverhalten von Programmen eleganter formulieren und es läßt sich die Klasse der Programme vergrößern, zwischen denen sich Programmhomomorphismen (das sind im wesentlichen Programmäquivalenzen im intuitiven Sinn) herstellen lassen.

Eine vollständige Version dieser Arbeit mit allen Beweisen liegt unter gleichem Titel als Institutsbericht vor (4). Dort wird auch vorgeführt, wie die Semantik von Flußdiagrammen durch Transport der Semantik der Programmiersprache in die freie Kategorie gewonnen werden kann; wir gehen hier nicht darauf ein.

Zur Erklärung der hier verwendeten Begriffe sei auf den Anhang, die Arbeiten von Burstall und Goguen und das Buch von Mac Lane verwiesen.

1. Potenzkategorien

1.1.Def.: Eine Potenzkategorie (P-Kategorie) ist eine Kategorie $\mathcal{P}$ mit den zusätzlichen Eigenschaften: (i) $(Ob_{\mathcal{P}}, \cup, \cap)$ und $(Hom_{\mathcal{P}}, \cup, \cap)$ sind vollständige distributive Verbände. (ii) Die Abbildungen $dom: Hom_{\mathcal{P}} \longrightarrow Ob_{\mathcal{P}}$, $cod: Hom_{\mathcal{P}} \longrightarrow Ob_{\mathcal{P}}$ und $id: Ob_{\mathcal{P}} \longrightarrow Hom_{\mathcal{P}}$ erhalten Suprema und Infima. (Im folgenden nennen wir dies Stetigkeit.) (iii) Die Komposition von Mor-

phismen ist distributiv über $\cup$ und $\cap$.

Es bezeichne $\top, \perp$ größtes und kleinstes Element von $\text{Ob}_{\mathcal{P}}$ und $\vee, \wedge$ größtes und kleinstes Element von $\text{Hom}_{\mathcal{P}}$.

1.2.Beispiel: Die Kategorie $\text{Rel}(M)$, die als einziges Objekt eine Menge M und als Morphismen alle Relationen $r \subseteq M \times M$ enthält, wird mit den mengentheoretischen Operationen $\cup, \cap, \circ$ und den Definitionen $\top := M$, $\perp := \emptyset$, $\vee := M \times M$, $\wedge := \emptyset$ zur Potenzkategorie. Unterkategorien von $\text{Rel}(M)$ werden später bei der Erklärung der Semantik eine entscheidende Rolle spielen.

1.3.Def.: Sei $B=(0,M,H)$ Basis einer Kategorie, $\hat{B}$ die davon erzeugte freie Kategorie. Definiere eine Kategorie $P(\hat{B})$ durch:

(i) $\text{Ob}_{P(\hat{B})} := \mathcal{P}(0)$ die Potenzmenge von 0 mit mengentheoretischer Vereinigung und Schnitt.

(ii) Es sei R die Menge aller Relationen (Aw, B) mit $A, B \in 0$ und $w \in M^*$.

Definiere $\text{dom}((Aw, B))=A$ und $\text{cod}((Aw, B))=B$. Auf R wird eine Komposition definiert durch $(Aw, B) \circ (Cv, A) := (Cv, B)$ für $A, B, C \in 0$ und $v, w \in M^*$.

R ist dieser Komposition gegenüber abgeschlossen. Nun seien $\cup$ und $\cap$ die mengentheoretische Vereinigung und Schnitt von Relationen und $\text{Hom}_{P(\hat{B})}$ der algebraische Abschluß von R unter $\cup$ und $\cap$. Weiter sei $\text{id} : \text{Ob}_{P(\hat{B})} \rightarrow \text{Hom}_{P(\hat{B})}$ die stetige Fortsetzung der Abbildung $A \mapsto (Ae, A)$ für $A \in 0$, wobei e das leere Wort in M^* sei. (Also $(Ae, A) = (A, A)$.) dom und cod werden nun ebenfalls durch ihre stetigen Fortsetzungen ersetzt.

1.4.Lemma: $P(\hat{B})$ ist Potenzkategorie und der Funktor $J : \hat{B} \rightarrow P(\hat{B})$, der durch die Basis $J(A) := \{A\}$ für $A \in 0$ und $J(A \xrightarrow{f} B) := (Af, B)$ für $f \in M$ definiert ist, ist eine Einbettung.

1.5.Def.: Ein Potenzfunktor (P-Funktor) ist ein Funktor zwischen Potenzkategorien, der Suprema und Infima respektiert.

1.6.Satz: Seien B, C Basen von Kategorien, $P(\hat{B}), P(\hat{C})$ die zugehörigen freien Kategorien. Dann gilt:

(i) Zu jedem Funktor $\hat{F} : \hat{B} \rightarrow P(\hat{C})$ gibt es einen eindeutig bestimmten Potenzfunktor $P(\hat{F}) : P(\hat{B}) \rightarrow P(\hat{C})$, der das folgende Diagramm kommutativ macht

$$\begin{array}{ccc} \hat{B} & \xrightarrow{\hat{F}} & P(\hat{C}) \\ J \searrow & & \nearrow P(\hat{F}) \\ & P(\hat{B}) & \end{array}$$

(J der Inklusions-funktor aus 1.4.)

(ii) Zu jedem Potenzfunktor $\mathfrak{F} : P(\hat{B}) \rightarrow P(\hat{C})$ gibt es einen eindeutig be-

stimmten Funktor $\xi: \hat{B} \longrightarrow P(\hat{C})$, so daß $\xi = P(\xi)$.

Dieser Satz gibt uns die Berechtigung, $P(\hat{B})$ die von B erzeugte freie Potenzkategorie zu nennen. Ebenso gelten die folgenden Sätze:

1.7.Korollar: Zu jedem Funktor $\hat{F}: \hat{B} \longrightarrow \hat{C}$ gibt es einen eindeutig bestimmten P-Funktor $P(\hat{F}): P(\hat{B}) \longrightarrow P(\hat{C})$, so daß

$$\begin{array}{ccc} \hat{B} & \xrightarrow{\hat{F}} & \hat{C} \\ J \downarrow & & \downarrow J \\ P(\hat{B}) & \xrightarrow{P(\hat{F})} & P(\hat{C}) \end{array}$$

kommutiert.

$P(\hat{F})$ heißt der von $\hat{F}$ erzeugte P-Funktor.

1.8.Satz: Seien $\hat{B}, \hat{C}$ freie Kategorien, $\hat{F}, \hat{G}: \hat{B} \longrightarrow \hat{C}$ Funktoren, $\gamma: \hat{F} \longrightarrow \hat{G}$ eine natürliche Transformation.

$P(\hat{B}), P(\hat{C})$ seien die zugehörigen P-Kategorien, $P(\hat{F}), P(\hat{G})$ die induzierten P-Funktoren.

Dann bestimmt γ eindeutig eine natürliche Transformation $P(\gamma): P(\hat{F}) \longrightarrow P(\hat{G})$.

2. Programme, Berechnungen, Ein/Ausgabeverhalten

2.1.Def.: Ein Programm ist ein Paar (B, S) , wobei B Basis einer Kategorie ist und S Basis eines Funktors $S: B \longrightarrow \text{Rel}$ in die Kategorie der Mengen und Relationen. Die Objekte von B heißen Marken, die Morphismen Anweisungen.

B beschreibt somit als Flußdiagramm die Kontrollstruktur des Programms, S die Semantik. (Zur Definition von S mit Hilfe der Semantik einer Programmiersprache siehe (4)). S assoziiert mit jeder Marke eine Menge, die man etwa als Menge der dort erlaubten Zustandsvektoren betrachten kann, und mit jeder Anweisung eine Relation zwischen Zustandsvektoren. (Bei deterministischen Programmen genügt es natürlich, sich auf partielle Funktionen zu beschränken.) Definiert man M als die Menge der überhaupt möglichen Zustandsvektoren, so kann man sich auf die Unterkategorie $\text{Rel}(M)$ von Rel beschränken; dies ist jedoch, wie gesagt, eine P-Kategorie.

S erzeugt daher einen P-Funktor $P(\hat{S})$, die die Semantik von Programmläufen beschreibt. $P(\hat{S})$ heißt die von dem Programm (B, S) induzierte Berechnung. Für einen Kontrollfluß $f \in \text{Hom}_B^A$ heißt $P(\hat{S})(f) = \hat{S}(f)$ die von (B, S) längs f berechnete Relation.

2.2.Def.: Sei (B, S) ein Programm, $A, A' \in \text{Ob}_{\hat{B}}$. I und O seien ausgezeichnete Anfangs- und Endmarke von B . Zeichne in $P(\hat{B})$ die folgenden Morphismen, aus:

$$C_B(A) := \bigcup \{ f : \exists_{M \in \text{Ob}_{\hat{B}}} A \xrightarrow{f} M \in \text{Hom}_{\hat{B}} \} \quad ; \quad C_B := C_B(I)$$

$$\mathcal{O}_B(A) := \bigcup \{ f : \exists_{M \in \text{Ob}_{\hat{B}}} M \xrightarrow{f} A \in \text{Hom}_{\hat{B}} \} \quad ; \quad \mathcal{O}_B := \mathcal{O}_B(O)$$

$$\text{Comp}_B(A, A') := C_B(A) \cap \mathcal{O}_B(A') \quad ; \quad \text{Comp}_B := C_B \cap \mathcal{O}_B$$

Definiere außerdem $C_{(B, S)}(A) := P(\hat{S})(C_B(A))$; $\mathcal{O}_{(B, S)}(A) := P(\hat{S})(\mathcal{O}_B(A))$ usf.

$\text{Comp}_{(B, S)}(A, A')$ heißt die von (B, S) zwischen A und A' berechnete Relation und $\text{Comp}_{(B, S)}$ die von (B, S) berechnete Relation.

$C_B(A)$ ist also die Summe über alle Kontrollflüsse des Programms, die in A beginnen, $\mathcal{O}_B(A)$ die Summe aller Berechnungsfolgen, die in A enden. Deshalb ist $\text{Comp}_B(A, A')$ die Summe aller Berechnungsfolgen, die von A nach A' führen. $\text{Comp}_{(B, S)}(A, A')$ ist die Semantik davon.

2.3.Lemma: $\text{Comp}_{(B, S)}(A, A') = \bigcup \{ \hat{S}(f) : A \xrightarrow{f} A' \in \text{Hom}_{\hat{B}} \}$

2.4.Lemma: Für ein Programm (B, S) mit $B = (O, M, H)$ und $A \in O$ gilt:

$$C_B(A) = \bigcup_i \{ C_B(A_i) \circ f_i : A \xrightarrow{f_i} A_i \in M \}$$

Dieses Lemma bietet eine Möglichkeit, C_B und auch Comp_B als Lösungen regulärer Gleichungssysteme darzustellen; näheres hierzu in (4). Der Semantik-P-Funktor ist mit der Lösung des Gleichungssystems verträglich; daher können auf diese Weise schon Korrektheitsbetrachtungen angestellt werden.

Auf die Behandlung von Korrektheitsproblemen mit Hilfe der kategorientheoretisch formulierten Methode von Floyd wird ebenfalls in (4) eingegangen.

2.5.Def.: Sei (B, S) Programm, I und O ausgezeichnete Anfangs- und Endmarke in B , V sei die Menge der Ein/Ausgabedaten und $e : V \rightarrow \hat{S}(I)$ und $a : \hat{S}(O) \rightarrow V$ seien Eingabe- und Ausgabeabbildung. Dann definiere das Ein/Ausgabeverhalten von (B, S) als $IO_{(B, S)} := a \circ \text{Comp}_{(B, S)} \circ e$.

Durch diese Definition wird es möglich, auf einfache Weise Programme miteinander zu vergleichen, die zwar auf gleichen Daten operieren, aber unterschiedlich lange oder unterschiedlich strukturierte Zustandsvek-

toren haben.

3. Homomorphismen und Äquivalenz von Programmen

Homomorphismen zwischen Programmen setzen die von diesen Programmen durchgeführten Berechnungen zueinander in Beziehung. Wir definieren hier zwei verschiedene Begriffe von Homomorphie: Den Goguenschen Homomorphismus (G-Homomorphismus) und den Potenzhomomorphismus (P-Homomorphismus). G-Homomorphismen sind leichter zu beweisen und reichen zur Behandlung von Korrektheits- und Terminationsproblemen nach Floyd aus; bei der Betrachtung von Äquivalenzproblemen sind jedoch die etwas komplizierteren P-Homomorphismen allgemeiner, da sie auch die Abbildung einzelner Instruktionen auf Schleifen zulassen. G-Homomorphismen implizieren P-Homomorphismen; die umgekehrte Relation gilt nicht. Beide Homomorphismen sind aufgrund der Freiheit der hier verwendeten Kategorien endlich beschreibbar.

3.1.Def.: Seien $(B, S), (B', S')$ Programme.

Ein G-Homomorphismus $h: (B, S) \longrightarrow (B', S')$ ist ein Paar $h = (\mathfrak{F}, \gamma)$ mit

(i) $\mathfrak{F}$ ist Funktor $\mathfrak{F}: \hat{B} \longrightarrow \hat{B}'$

(ii) γ ist natürliche Transformation $\gamma: \hat{S} \longrightarrow (\hat{S}' \circ \mathfrak{F})$

Ist für alle $v \in \text{Ob}_{\hat{B}} \gamma_v$ eine Inklusion, so heißt h eine G-Simulation.

Ein P-Homomorphismus $h: (B, S) \longrightarrow (B', S')$ ist ein Paar $h = (\mathfrak{F}, \gamma)$ mit

(i) $\mathfrak{F}$ ist P-Funktor $\mathfrak{F}: P(\hat{B}) \longrightarrow P(\hat{B}')$

(ii) γ ist natürliche Transformation $\gamma: P(\hat{S}) \longrightarrow (P(\hat{S}') \circ \mathfrak{F})$

Definiere P-Simulation analog zu G-Simulation.

3.2.Satz: Jeder G-Homomorphismus $h: (B, S) \longrightarrow (B', S')$ induziert eindeutig einen P-Homomorphismus $P(h): (B, S) \longrightarrow (B', S')$.

Beweis: 1.7. und 1.8.

3.3.Satz: $h = (\mathfrak{F}, \gamma): (B, S) \longrightarrow (B', S')$ ist P-Homomorphismus $\iff$

$\iff \left\{ \begin{array}{l} \text{(i) } \mathfrak{F} = P(\hat{F}), \text{ wobei } F: B \longrightarrow P(\hat{B}') \text{ Basis eines Funktors} \\ \text{(ii) } \gamma = P(\mathfrak{K}), \text{ wobei } \mathfrak{K} \text{ eine Abbildung } \mathfrak{K}: 0 \longrightarrow \text{Hom}_P(\hat{B}') \text{ ist, so} \\ \text{daß für alle } A, B \in 0 \text{ und } A \xrightarrow{f} B \in M \text{ das Diagramm} \end{array} \right.$

$$\begin{array}{ccc} SA & \xrightarrow{Sf} & SB \\ \mathfrak{K}(A) \downarrow & & \downarrow \mathfrak{K}(B) \\ P(\hat{S}')(FA) & \xrightarrow{P(\hat{S}')(Ff)} & P(\hat{S}')(FB) \end{array}$$

kommutiert.

Der Beweis findet sich in (4). Auf ähnliche - einfachere - Weise lassen sich G-Homomorphismen endlich beschreiben. Der folgende Satz stellt eine Beziehung zwischen den Homomorphismen und der intuitiven Äquivalenz von Programmen her. Dabei heißen zwei Programme äquivalent, wenn sie bei gleicher Eingabe die gleiche Ausgabe erzeugen. Um dies exakt zu formulieren, führen wir die Konsistenzbedingung ein.

3.4.Satz: Seien $(B,S), (B',S')$ Programme mit Anfangs- und Endmarken A,E bzw. A',E' . Für beide Programme sei V die Menge der Ein/Ausgabedaten. e,e' und a,a' seien die Ein/Ausgabeabbildungen. Es gelte $\text{def}(e)=\text{def}(e')$. $h:(B,S) \rightarrow (B',S')$ sei ein G-Homomorphismus $h=(\hat{F},\gamma)$ (bzw. ein P-Homomorphismus $h=(P(\hat{F}),P(\gamma))$), so daß $FA=A'$ und $FE=E'$.

Falls die Konsistenzbedingung

$$\gamma_A \circ e = e' \quad \text{und} \quad a' \circ \gamma_{E'} = a$$

gilt, so ist $IO_{(B,S)} \subset IO_{(B',S')}$. Ist außerdem $\hat{F}$ (bzw. $P(\hat{F})$) voll, so gilt sogar Gleichheit.

Zum Beweis genügt es, die Kommutativität des Diagramms

für alle $A \xrightarrow{f} E$ zu zeigen. Das mittlere Diagramm kommutiert, weil γ natürliche Transformation ist, die beiden äußeren Diagramme kommutieren aufgrund der Konsistenzbedingung; daher kommutiert das ganze Diagramm. Der Rest ist nun einfach zu zeigen.

Es ist im allgemeinen nicht möglich, $\text{Comp}_{(B,S)} \subset \text{Comp}_{(B',S')}$ zu beweisen, wenn nur ein Homomorphismus vorliegt. Es läßt sich aber zeigen, daß dies bei der G-Simulation (bzw. P-Simulation) der Fall ist. Hier kann man jedoch keine Gleichheit erzielen, indem man den Funktor als voll voraussetzt.

Mit dem Satz 3.4. ergibt es sich als eine weitere Methode, die Korrektheit von Programmen zu beweisen, daß man andere Programme sucht, deren Korrektheit bereits gezeigt oder offensichtlich ist und dann versucht, Homomorphismen dazwischen zu beweisen. Hierzu bieten sich besonders die allgemeineren P-Homomorphismen an.

In (4) ist außerdem eine spezielle Klasse von Homomorphismen definiert,

welche die Termination von Programmen erhalten; hiermit lassen sich auf ähnliche Weise Terminationsbeweise erreichen.

Anhang

Eine Kategorie $\mathcal{K}$ ist ein Tripel $(\text{Ob}_{\mathcal{K}}, \text{Hom}_{\mathcal{K}}, H_{\mathcal{K}})$, wobei $\text{Ob}_{\mathcal{K}}$, $\text{Hom}_{\mathcal{K}}$ Klassen von Objekten und Morphismen und $H_{\mathcal{K}}: \text{Hom}_{\mathcal{K}} \rightarrow \text{Ob}_{\mathcal{K}} \times \text{Ob}_{\mathcal{K}}$ eine Abbildung, so daß die üblichen Eigenschaften gelten. Wir benützen auch die Bezeichnungen $\text{dom}(f) = (\text{pr}^1 \circ H_{\mathcal{K}})(f)$ und $\text{cod}(f) = (\text{pr}^2 \circ H_{\mathcal{K}})(f)$ und $\mathcal{K}(A, B) = H_{\mathcal{K}}^{-1}(A, B)$. Die Begriffe des Funktors und der natürlichen Transformation sind wie üblich erklärt; eine Basis einer Kategorie ist ein Tripel $B = (O, M, H)$, wobei O und M Mengen und $H: M \rightarrow O \times O$ eine Abbildung. Die von B erzeugte freie Kategorie $\hat{B}$ stellt im wesentlichen den algebraischen Abschluß von B unter einer Komposition auf M dar. In dieser Arbeit sind Basen von Kategorien stets endlich. Eine Basis einer Kategorie kann man sich als gerichteten benannten Graphen vorstellen. Ist B Basis einer Kategorie und $\mathcal{K}$ eine weitere Kategorie, so ist die Basis eines Funktors $F: B \rightarrow \mathcal{K}$ eine Abbildung, die mit dom und cod verträglich ist. Der davon erzeugte freie Funktor ist die homomorphe Fortsetzung davon bezüglich der Komposition. Jeder Funktor von einer freien Kategorie ist selbst frei.

Literatur

1. Burstall, R.M., An Algebraic Description of Programs with Assertions, Verification and Simulation, Proc. on an ACM Confr. on Proving Assertions about Programs, Las Cruces 1972, pp. 7-14
2. Goguen, J.A., On Homomorphisms, Correctness, Termination, Unfoldments and Equivalence of Flow Diagram Programs, JCSS Vol.8 No.3, 1974
3. Goguen, J.A., Set-Theoretic Correctness Proofs, UCLA Comp.Sc.Dept. Reports on Semantics and Theory of Computation No.1, 1974
4. Klaeren, H.A., Potenzkategorien und ihre Anwendung auf den Beweis von Programmeigenschaften, Berichte d.Inst.f.Angew.Math. und Informatik Bonn, Band 1 (erscheint; dort weitere Literaturang.)
5. Mac Lane, S., Categories - for the working mathematician, Springer 1972