Andmetüübid, universaalalgebra ja koalgebrad.

Lauri Tart

16. ja 23. veebruar 2007.

Sissejuhatus

Eelmise loengu kohaselt teame me, et universaalkoalgebra on üks tore teooria, mille alla mahuvad mitmed AT jaoks huvitavad valdkonnad, aga ka topoloogilised teooriad (kas me nendeni ka kunagi jõuame?).

Teisalt, puhas duaalne universaalalgebra oli ka kunagi üks teooria, mis kedagi väga ei huvitanud; ja seda just näidete vähesuse tõttu.

Universaalalgebra ise on muidugi üks lähenemisviis andmetüüpidele, aga mitmed nähtused (eriti peaks seda olema mittedeterministlikkus) leiavad loomuliku seletuse ja raamistiku just universaalkoalgebras.

Sissejuhatus

Paljud koalgebra funktorid nõrku (st. mitteühese vahemorfismiga) konservatiivseid ruute säilitavad, ja see annab võimaluse ilusasti näidata mitmeid koalgebrate ehituse omadusi.

Kuid on olemas ka ilma sellise omaduseta olulisi funktoreid (jälle viide topoloogiale), lisaks kipub saadav teooria universaalalgebraga võrreldes väga spetsiifiline olema. Seega antud kursuses püütakse võimalikult palju teha ilma sellise eelduseta. Tuleb välja, et suur osa olulisest teooriast, sh. Birkhoffi teoreem (koteoreem?), on selliselt siiski saavutatav.

On muidugi mõned ehituslikud omadused, mille jaoks on vajalik mingisugune nõrkade piiride säilitamine. See peaks moodustama minu viimase peatüki ettekannete materjali. Ja näiteks koalgebrate kategooria täielikkus kasutab seda eeldust.

Sissejuhatus

Kuigi hulkade kategooria ei tundu olevat alusena just hädavajalik, on see siin intuitiivsuse suurendajana siiski baaskategooriana kasutusel. Gumm lubab, et ehk näeb ta kunagi valgust ja saab hulkade kategooriast lahti, aga mitte veel.

Andmetüübid ja universaalalgebra

Vaatamegi nüüd andmetüüpide universaalalgebralist käsitlust ja jõuame selle juurest universaalkoalgebralise lähenemiseni.

Näide: Programmeerija võib määrata andmetüübi (näiteks siin kahendpuud) funktsionaalses keeles umbes nii:

$$BT = \begin{array}{l} e : 1 \to BT \\ mkT : BT \times BT \to BT \end{array}$$

Universaalalgebraliselt tähendab see, et kahendpuud on mingi ühte fikseeritud elementi sisaldav rühmoid.

Andmetüübid ja universaalalgebra

Muidugi, selliseid rühmoide on paljuvõitu (nt. suvaline monoid). Seega oleks meil vaja kuidagi eraldada andmetüüpe sama signatuuriga universaalalgebrate hulgas. Selleks toome sisse kaks eeldust:

Aksioom 1: Andmetüüp sisaldab ainult oma definitsioonis sisalduvate tehete abil saadavaid elemente.

Tehted on siin siis andmetüübi konstruktorid ja andmetüüp sisaldab vaid konstruktorite poolt loodavaid elemente.

Andmetüübid ja universaalalgebra

Sellega me välistame teatud rühmoidid, nt. $(\mathbb{R}, 0, +)$, aga mitte kõik, nt. $(\mathbb{N} \setminus \{0\}, 1, +)$. Eriti paha on see, et üheelemendiline universaalalgebra (rühmoid elemendiga) rahuldab seda nõuet. Me tahaksime hoopis vastupidist sorti rühmoidi. Seega

Aksioom 2: Võrdsed on ainult sellised elemendid, mis on saadud ühe ja sama konstruktsiooni tulemusena.

Need aksioomid defineerivad andmetüübi kui antud signatuuriga absoluutselt vaba universaalalgebra tühja baasiga.

Andmetüübid ja universaalalgebra

Kahendpuude korral oleks siis

$$BT = \{e, mkT(e,e), mkT(e, mkT(e,e)), mkT(mkT(e,e), e), mkT(mkT(e,e), mkT(e,e)), \ldots\}$$

koos süntaktilise korrutamisega ja elemendi e fikseerimisega.

Üldiselt on eeltoodud sorti (arvatavasti tähendab see nullaarseid tehteid sisaldavat signatuuri) signatuuride korral vastava absoluutselt vaba tühja baasiga algebra olemas (ja ühene), ja me võime lihtsalt eeldada, et

Aksioom 3: Andmetüüp on oma spetsifikatsioonile vastava signatuuriga absoluutselt vaba tühja baasiga algebra.

Andmetüübid ja universaalalgebra

Seega saame me defineerida naturaalarvud spetsifikatsiooniga

$$\begin{aligned} Nat = & \\ & 0 : 1 \longrightarrow Nat \\ & succ : Nat \longrightarrow Nat \end{aligned}$$

Huvitavamate (*many-sorted*) tüüpide jaoks, nt. naturaalarvude pinu saamiseks

$$\begin{aligned} NatStack = & \\ & empty : 1 \longrightarrow NatStack \\ & push : Nat \times NatStack \longrightarrow NatStack \end{aligned}$$

Many-sorted arvatavasti tähendab siin parametriseeritud tehete pere olemasolu spetsifikatsioonis.

Andmetüübid ja universaalalgebra

Veel huvitavam on mitut sorti andmete hoidmine tüübi eri osades, nt puud, kus andmeid hoitakse nii lehtedes kui sisemistes tippudes:

$$
\begin{aligned}
\mathit{Tree} = \quad & \\
& \mathit{emptyTree} : 1 \to \mathit{Tree} \\
& \mathit{mkLeaf} : \mathit{Nat} \to \mathit{Tree} \\
& \mathit{mkNode} : \mathit{Tree} \times \mathit{Char} \times \mathit{Tree} \to \mathit{Tree}
\end{aligned}
$$

Predikaadid

Eelduse kohaselt on iga teatud tüüpi objekt üheselt konstrueeritav. Seetõttu saab sisse tuua predikaadid, mis määravad kindlaks, kas antud objekt on loodud mingi konkreetse konstruktori poolt. Näiteks eeltoodud puude korral

$$\mathit{isEmpty} : \mathit{Tree} \longrightarrow \mathit{Boolean}$$

$$\mathit{isLeaf} : \mathit{Tree} \longrightarrow \mathit{Boolean}$$

$$\mathit{isNode} : \mathit{Tree} \longrightarrow \mathit{Boolean}$$

Predikaadid

Kui meil on vaid kaks konstruktorit, siis on tegelikult vaja vaid ühte predikaati, mille eitamisel saame teise predikaadi. Seega $isPush = not\ isEmpty$ ja

$$isZero : Nat \rightarrow Boolean$$

on piisav, otsustamaks arvu pärinemise üle *succ*-konstruktorist.

Selektorid

Kui meil on mingit tüüpi andmed olemas, nt puud või pinud, siis me tahame neis olevaid andmeid näha ja neid komponentideks lahutada. Seda saame me teha siis, kui teame, mis sorti (üheselt määratud) konstruktoriga meie objekt saadud on. Seega on iga konstruktori iga argumendi jaoks seda argumenti tagastav (osaline) funktsioon (ehk *selektor*), nt.

$$mkNode : Tree \times Char \times Tree \longrightarrow Tree$$

korral

$$\begin{aligned} left &:: Tree \longrightarrow Tree \\ node &:: Tree \longrightarrow Char \\ right &:: Tree \longrightarrow Tree \end{aligned}$$

Selektorid

Need funktsioonid on defineeritud antud konstruktori poolt loodud objektidel, siin siis

$$\mathrm{dom}\,(\mathit{left}) = \mathrm{dom}\,(\mathit{right}) = \mathrm{dom}\,(\mathit{node}) = \{t \in \mathit{Tree} | \mathit{isNode}(t)\}$$

Analoogiliselt on konstruktori *mkLeaf* jaoks funktsioon $\mathit{Leaf} :: \mathit{Tree} \to \mathit{Nat}$, aga argumentideta konstruktori *emptyTree* jaoks on triviaalne selektor.

Naturaalarvude korral on meil $\mathit{pred} :: \mathit{Nat} \to \mathit{Nat}$, kus $\mathrm{dom}\,(\mathit{pred}) = \{n \in \mathit{Nat} | \mathit{not}\ \mathit{isZero}(n)\}$, ja pinude jaoks

$$\begin{aligned} \mathit{top} &:: \mathit{NatStack} \to \mathit{Nat} \\ \mathit{pop} &:: \mathit{NatStack} \to \mathit{NatStack}, \end{aligned}$$

kus $\mathrm{dom}\,(\mathit{top}) = \mathrm{dom}\,(\mathit{pop}) = \{s \in \mathit{NatStack} | \mathit{not}\ \mathit{isEmpty}(s)\}$.

Algebrast koalgebrasse

Süntaktiliselt on konstruktorid sellised morfismid, mille kodomeen on antud tüüpi, ja selektorid niisugused morfismid, mille domeen on antud tüüpi.

Viimase fakti abil saame me konstruktorid ja selektorid ühe morfismi abil kokku võtta.

Puude jaoks siis on konstruktor

$$c : 1 + \mathit{Nat} + (\mathit{Tree} \times \mathit{Char} \times \mathit{Tree}) \to \mathit{Tree}$$

ja selektor

$$s : \mathit{Tree} \to 1 + \mathit{Nat} + (\mathit{Tree} \times \mathit{Char} \times \mathit{Tree}).$$

Algebrast koalgebrasse

Konstruktorid on seega funktsioonid $f : F(X) \to X$, kus F on "hulgateoreetiline konstruktsioon" (ehk funktor), mille argumendiks on loodav tüüp. Antud näites

$$F(X) = \{0\} + \mathit{Nat} + X \times \mathit{Char} \times X.$$

Selektor seevastu on funktsioon $\alpha : X \to F(X)$. See fakt on andmetüüpide koalgebralise käsitluse aluseks.

Universaalalgebra: defineerimine

Kuna andmetüübid on üheselt konstrueeritud, siis saab neil defineerida rekursiivseid funktsioone argumendi konstruktsioonist lähtuvalt.

Näiteks pinu pikkuse leidmise funktsioon

$$\begin{aligned} \mathit{length}(\mathit{empty}) &= 0 \\ \mathit{length}(\mathit{push}(n, s)) &= 1 + \mathit{length}(s), \end{aligned}$$

või faktoriaal

$$\begin{aligned} \mathit{fact}(0) &= 1 \\ \mathit{fact}(\mathit{succ}(n)) &= \mathit{succ}(n) * \mathit{fact}(n). \end{aligned}$$

Universaalalgebra: defineerimine

Sellised definitsioonid on korrektsed, kui vasakul pool võetakse arvesse kõik konstruktorid, ja selline konstrueeritud andmete mustrite kasutamine on lihtsalt süntaktiline meetod selektorite vältimiseks. Eelnev on ekvivalentselt defineeritav kui

$$\begin{aligned} \text{if } \mathit{isEmpty}(u) \quad & \text{then } \mathit{length}(u) = 0 \\ & \text{else } \mathit{length}(u) = 1 + \mathit{length}(\mathit{pop}(u)). \end{aligned}$$

Mitme argumendi jaoks nt.

$$\begin{aligned} \mathit{append}(\mathit{empty}, s) &= s \\ \mathit{append}(\mathit{push}(n, s1), s2) &= \mathit{push}(n, \mathit{append}(s1, s2)). \end{aligned}$$

Universaalalgebra: defineerimine

Absoluutselt vaba algebra tühja baasiga (tähistame seda $\mathcal{D}$) on sama signatuuriga algebrate kategoorias algobjekt, st. mistahes sama signatuuriga algebra $\mathcal{A}$ korral leidub ühene homomorfism $\varphi : \mathcal{D} \to \mathcal{A}$. See annab meile võimaluse rekursiivsete funktsioonide defineerimiseks lihtsalt sama signatuuriga andmetüüpide abil. Näiteks funktsioon *length* on saadav homomorfismina algebrasse $(\mathbb{N}, \mathit{succ}, 0)$, kus

$$0 : 1 \to \mathbb{N},$$

$$\mathit{succ} : \{0\} \times \mathbb{N} \to \mathbb{N}.$$

Universaalalgebra: tõestamine

Absoluutselt vabal algebral tühja baasiga ei ole pärisalamalgebraid (see on järeldus esimesest aksioomist). Ekvivalentselt, kui $\mathcal{P} \subseteq \mathcal{D}$ on kinnine signatuuri kuuluvate tehete suhtes, siis $\mathcal{P} = \mathcal{D}$.

See annab meile nn. induktsioonipõhimõtte väidete tõestamiseks.

Kui $\mathcal{D} = Nat$, siis on induktsioon

$$\frac{P(0) \wedge \forall x \in Nat.[P(x) \Rightarrow P(succ(x))]}{\forall x \in Nat.P(x)}.$$

Kui $\mathcal{D} = NatStack$, siis

$$\frac{P(empty), \forall n \in Nat.\forall s \in NatStack.[P(s) \Rightarrow P(push(n, s))]}{\forall s \in NatStack.P(s)}.$$

Universaalalgebra: tõestamine

Üks andmetüüpide induktsiooni lai rakendusvaldkond on funktsioonide omaduste, eriti võrdsuse tõestamine. Näiteks

$$length(append(s_1, s_2)) = length(s_1) + length(s_2),$$

kus *append* defineeritakse võrdustega

$$\begin{aligned} append(empty, s) &= s \\ append(push(n, s1), s2) &= push(n, append(s1, s2)). \end{aligned}$$

Siis fikseerime pinu s ja vaatleme hulka

$$P = \{x \in NatStack | length(append(x, s)) = length(x) + length(s)\}.$$

Näeme, et $empty \in P$ ja $[x \in P \Rightarrow push(n, x) \in P]$, kuna

Universaalalgebra: tõestamine

$$\begin{aligned} length(append(push(n,x),s)) &= length(push(n,append(x,s))) \\ &= 1 + length(append(x,s)) \\ &= 1 + length(x) + length(s) \\ &= length(push(n,x)) + length(s). \end{aligned}$$

Seega $P = NatStack$ induktsiooni kohaselt, ehk antud funktsioonid on võrdsed.

Kuna kahe funktsiooni $\phi, \varphi : \mathcal{A} \to \mathcal{B}$ võrdsustaja

$$\{a \in \mathcal{A} | \phi(a) = \varphi(a)\}$$

on alati alamalgebra, siis ütleb induktsioonipõhimõte, et funktsioonid absoluutselt vabast algebrast tühja baasiga on üheselt määratud sihtalgebraga.

Koalgebralised tüübid

Kahjuks ei rahulda kõik praktiliselt kasutatavad andmestruktuurid esimest aksioomi. Näiteks lõpmatut sisendit, faile või jadasid modelleerivad vood.

Vaatleme järgnevaid definitsioone, kus $:=$ *cons* (vist).

$ones = 1 : ones$ defineerib jada $[1, 1, 1, \ldots]$, ja

$$from\, n = n : from(n + 1),$$
$$int = from(0)$$

defineerib *int* kui jada $[0, 1, 2, 3, \ldots]$.

Koalgebralised tüübid

Sellised objektid osutuvad kasulikuks, ja neil saab funktsioone defineerida samamoodi, nagu lõplikult konstrueeritud objektidel, nt.

$$twice\,(h : l) = (2 * h) : twice\,l$$

ja

$$add(h1 : l1)(h2 : l2) = (h1 + h2) : add(l1, l2).$$

Kahjuks ei saa me enam induktsiooni abil väiteid tõestada, kasvõi et

$$add(l, l) = twice(l).$$

Seda just esimese aksioomi rahuldamatuse tõttu, sest induktsioon (nt. üle :) ei jõua kunagi baasini.

Koalgebralised tüübid

Seega meie lõpmatud vood ei oma nullaarseid tehteid konstruktiivse definitsiooni alustamiseks. Samas saame me neid defineerida näiteks selektorite *head* ja *tail* abil:

$$\begin{aligned} S = & \\ head & : S \to Nat \\ tail & : S \to S, \end{aligned}$$

või kombineeritult $< s, t >: S \to Nat \times S$.

Koalgebralised tüübid: süsteem :)

Eelnev on näide selektorite või *vaatlejate* abil defineeritud *koandmetüübist* või *süsteemist*.

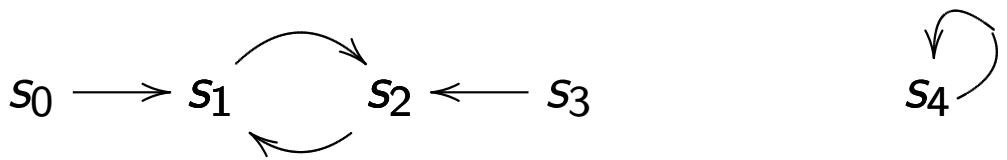
Intuitiivselt, selektorid tagastavad mingeid omadusi või komponente. Seega on kasulik selliseid tüüpe vaadelda seisundite hulkadena. Selektorid tagastavad siis seisundite omadusi või muudavad seisundeid.

Sellisena vooge modelleeritaksegi, teatud seisunditena. Funktsioon *head* tagastab voo esimese arvu, ja *tail* liigub voo ülejäänud osa modelleerivasse seisundisse.

Samas ei ole sugugi selge, et eelnev andmetüüp peaks tingimata olema $\mathbb{N}^\omega$.

Koalgebralised tüübid: näide

Näide: Olgu $S = \{s_0, s_1, s_2, s_3, s_4\}$, ja defineerime $head(s_i) = i \mod 2$, $tail(s_0) = tail(s_2) = s_1$, $tail(s_1) = tail(s_3) = s_2$, $tail(s_4) = s_4$.



Hulka S me ei suuda otse vaadelda (kuna põhimõtteliselt on aluseks eelnev selektorite abil antud spetsifikatsioon). Me saame funktsiooni *head* abil seda kaudselt vaadelda ja funktsiooni *tail* abil teisendada. Põhimõtteliselt võiks tegu olla sellise musta kastiga, millel on kaks nuppu.

Koalgebra: eraldamatus

Vaatluste abil saame me eraldada olekuid s_1, s_2 ja s_4, aga olekud s_0 ja s_3 on eraldamatud olekutest s_2 ja s_1. Kuna viimased paarid käituvad samamoodi ja ei ole otseselt vaadeldavad, ei ole ka põhjust neid eraldada.

Seega samastamisel saame üldistatud võrdsuse, mida väljendab

Aksioom 4: Kaks olekut loetakse võrdseteks, kui neid ei ole võimalik vaatluste (jada) abil eraldada.

Koalgebra: eraldamatus

Seda eraldamatuse seost tähistame sümboliga $\sim$.

See seos peaks rahuldama tingimust

$$\frac{u \sim v}{head(u) = head(v) \wedge tail(u) \sim tail(v)}.$$

Antud näites saame faktoriseerida ja tulemuseks on

ehk minimaalne süsteem, mis käitub vaatleja jaoks samamoodi.

Koalgebra: näide

Teiseks näiteks oleks lõplik automaat, mis otsustab, kas mingi sõne rahuldab antud regulaaravaldist. Tavaline spetsifikatsioon oleks:

$$\begin{aligned} \mathit{isTerminal} &: \mathit{State} \to \mathit{Boolean} \\ \mathit{delta} &: \mathit{State} \times \mathit{Char} \to \mathit{State}. \end{aligned}$$

See ei ole algebraline tüüp ega süsteem, aga muutub viimaseks, kui võtta

$$\mathit{delta} : \mathit{State} \to \mathit{State}^{\mathit{Char}}.$$

Koalgebra: eraldamatus

Kasutaja jaoks ei ole olekud olulised, talle on vaja, et automaat teeks, mida vaja. Jälle saame kaks olekut samastada, kui nende väline käitumine ei ole eraldatav. Vaatlused koosnevad sümbolite sisestamisest, millele järgneb *isTerminal* nupule vajutamine.

Eraldamatus selle andmetüübi jaoks on Nerode'i kongruentsus (mis see ka ei oleks). Selle järgi faktoriseerimine annab meile minimaalse automaadi.

Koalgebra: eraldamatus

Meil on Eraldamatuse reegliks

$$\frac{s \sim t}{\mathit{isTerminal}(s) = \mathit{isTerminal}(t) \wedge \forall c \in \mathit{Char}.\mathit{delta}(s,c) \sim \mathit{delta}(t,c)}.$$

Suvalist seda reeglit rahuldavat seos ρ, st seost, mille korral

$$\frac{s \,\rho\, t}{\mathit{isTerminal}(s) = \mathit{isTerminal}(t) \wedge \forall c \in \mathit{Char}.\mathit{delta}(s,c) \,\rho\, \mathit{delta}(t,c)},$$

nimetame *bisimulatsiooniks*. Vastavalt on juhul $u \,\rho\, v$ olekud u ja v bisimilaarsed. Bisimulatsioonide ühend on bisimulatsioon, seega on olemas suurim bisimulatsioon, milleks ongi eraldamatus $\sim$.

Koalgebra: koinduktsioon

Reeglid eraldamatuse jaoks on kasutatavad tõestamisel koinduktsiooniga. Põhimõtteliselt: kui kaks voogu on bisimilaarsed, siis on nad eraldamatud.

Näiteks saab tõestada, et $twice(l) = add(l, l)$. Selleks on lihtsalt vaja, et $\rho = \{(twice(l), add(l, l))|l \in Stream\}$ oleks bisimulatsioon.

Koalgebra: koinduktsioon

Viimases on lihtne veenduda:

$$head(twice(l)) = 2 * head(l) = head(l) + head(l) = head(add(l, l)),$$
$$tail(twice(l)) = twice(tail(l)) \, \rho \, add(tail(l), tail(l)) = tail(add(l, l)).$$

See muidugi tähendab, et $twice(l)$ ja $add(l, l)$ on eraldamatud, mitte tingimata võrdsed.

Koalgebra: lõppkoalgebra

Kui meil on mingi koandmetüüp, siis me tahaksime mingit universaalset koandmetüüpi, mis oleks sarnane universaalalgebra algalgebraga.

Paljudel juhtudel selline objekt õnneks leidub, ja kui ta leidub, siis peavad selle moodustama kõigi võimalike vaatluste bisimilaarsusklassid.

Koalgebra: lõppkoalgebra

Juhul

$$\begin{aligned} S = & \\ \mathit{head} & : S \to \mathit{Nat} \\ \mathit{tail} & : S \to S, \end{aligned}$$

on selleks kõikvõimalikud naturaalarvude lõpmatud vood, sest sedasorti süsteemi käitumised on vaadeldavad selliste voogudena, ja need vood on kõik ka eraldatavad.

Koalgebra: näide

Vastavas väiksemas näites on olekute s_0 ja s_2 käitumisteks vood $(01)^\omega$, olekute s_1 ja s_3 käitumisteks vood $(10)^\omega$, ja oleku s_4 käitumisteks vood $(0)^\omega$.

See käitumine annab meile võimaluse defineerida ühene homomorfism näites toodud süsteemist naturaalarvude voogude lõppsüsteemi.

Kokkuvõte

Kokkuvõttes, algebralise lähenemise korral eraldasime erinevalt konstrueeritud objekte, koalgebraliselt vaatleme peidetud olekutega süsteemide välist käitumist. Me samastame need olekud, mille väline käitumine on sama, st. nad on vaatleja jaoks eraldamatud.

Andmeobjektid	konstruktsioonid	vaatlused
Võrdsus	sama konstruktsioon	eraldamatus
Tõestus	induktsioon	koinduktsioon
Semantiline piirkond	algalgebra	lõppkoalgebra

Kokkuvõte: universaalalgebra

Universaalalgebra on traditsiooniliselt $\mathcal{A} = (A; (f_i)_{i \in I})$, kus A on (mittetühi) hulk, ja on antud tehted $f_i^A : A^{n_i} \to A$. Need tehted saab kombineerida kujutuseks $f^A : \sum_{i \in I} A^{n_i} \to A$, ehk siis üldine universaalalgebra on antud hulgaga A ja morfismiga

$$f^A : F(A) \to A,$$

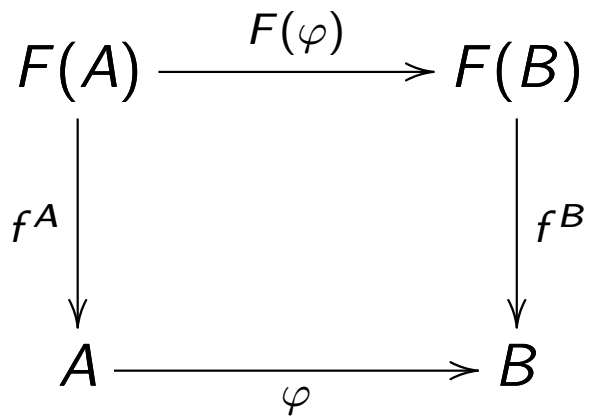
kus $F(X) = \sum_{i \in I} X^{n_i}$ annab algebra A sarnasustüübi (milleks iganes selline mõiste hea on).

Kokkuvõte: universaalalgebra

Iga kujutus $\varphi : \mathcal{A} \to \mathcal{B}$ indutseerib kanoonilisel viisil kujutuse

$$F(\varphi) : F(A) \to F(B).$$

Kujutuse φ homomorfsus on samaväärne ruudu



kommuteeruvusega.

Kokkuvõte: koalgebra

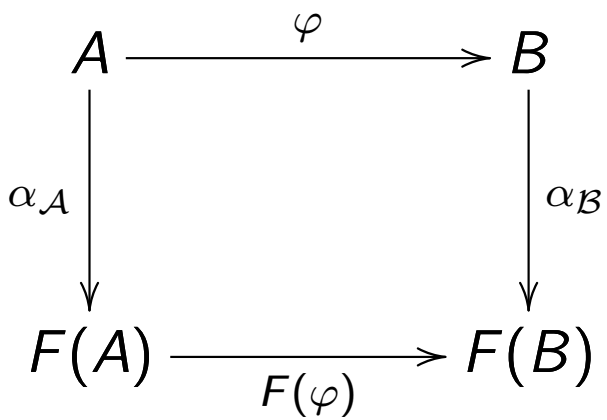
Koalgebraline tüüp on defineeritud mingi hulga selektoritega, mida saab samuti üheks morfismiks kombineerida:

$$\alpha_{\mathcal{A}} : A \to F(A).$$

Koalgebra on seega paar $\mathcal{A} = (A, \alpha_{\mathcal{A}})$, kus $\alpha_{\mathcal{A}} : A \to F(A)$.

Kokkuvõte: koalgebra

Homomorfism koalgebrate $\mathcal{A} = (A, \alpha_{\mathcal{A}})$ ja $\mathcal{B} = (B, \alpha_{\mathcal{B}})$ vahel on kujutus $\varphi : A \longrightarrow B$ (tegelikult on huvitav, miks just sedapidi), mille korral ruut

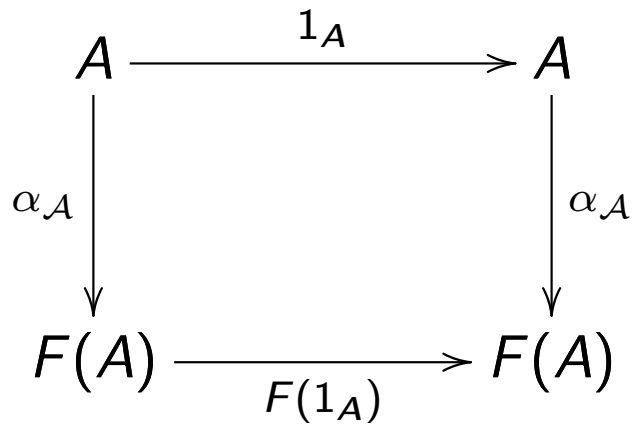


kommuteerub.

Kokkuvõte: funktorid

Eelnevalt on F olnud "mingi hulgateoreetiline konstruktsioon". Nagu näha, peab see toimima ka kujutustel, viies kujutuse $f : A \to B$ kujutuseks $F(f) : F(A) \to F(B)$.
Kui me tahame, et 1_A oleks homomorfism, peab

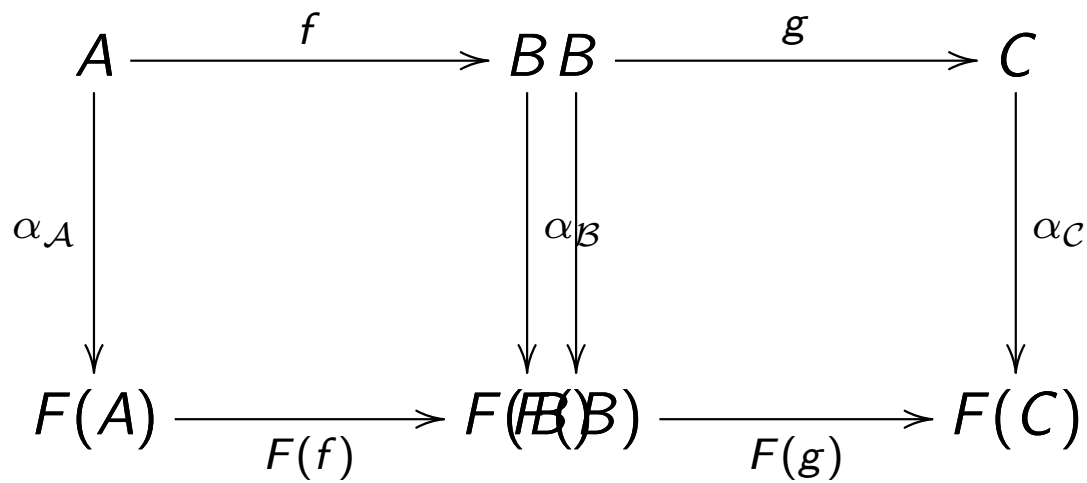
$$F(1_A) = 1_{F(A)}.$$



Kokkuvõte: funktorid

Analoogiliselt, kui homomorfismide kompositsioon peaks olema homomorfism, siis on vaja, et

$$F(f \circ g) = F(f) \circ F(g).$$



Kokkuvõte: funktorid

Selline asjandus kannab meil nime funktor, nagu loodetavasti järgmine kord selgub. Universaalalgebras on selline funktor kujul $F(X) = \sum_{i \in I} X^{n_i}$, aga meil läheb vaja ka üldisemaid funktoreid, nagu automaatide näitest näha oli. Seal tekkis

$$F(X) = Boolean \times X^{Char},$$

ja mittedeterministliku automaadi jaoks

$$F(X) = Boolean \times \mathcal{P}(X)^{Char}.$$

"Õige" on selliste asjade jaoks kasutada kategooriateooria keelt. Seega järgnevas seda tutvustataksegi. Nüüd tulevad

Exercises.

Exercises: 2.1

Exercise 2.1 Protofunktori F korral JVOS:
1) F on funktor;
2) 1_A on F-koalgebrate homomorfism ja F-koalgebrate homomorfismide kompositsioon on homomorfism.

Exercises: 2.1 lahendus

On kaunis selge, et 1) $\Rightarrow$ 2).
Teisalt järeldub tingimusest 2), et

$$F(1_A) \circ \alpha_{\mathcal{A}} = 1_{F_A} \circ \alpha_{\mathcal{A}}$$

ja

$$F(g \circ f) \circ \alpha_{\mathcal{A}} = F(g) \circ F(f) \circ \alpha_{\mathcal{A}}$$

suvalise koalgebra $\alpha_{\mathcal{A}} : A \to F(A)$ korral. Kuna see koalgebra on suvaline, siis saab temaga alati saavutada mistahes punktid, kus 1_{F_A} ja $F(1_A)$ või $F(g \circ f)$ ja $F(g) \circ F(f)$ erinevad, seega peavad viimased olema võrdsed.
Ma arvan, et see tähendab umbkaudu $F(A)$ generaatoriks olemist.

Exercises: 2.2

Exercise 2.2

1. Universaalalgebrad $A = (A, (\cdot, ^{-1}, 1))$ tüübiga $(2, 1, 0)$ vastavad üheselt kujutustele

$$f^A : (A \times A) + A + 1 \to A.$$

2. Kujutus F, mille korral $F(X) = (X \times X) + X + 1$ ja kui on antud $g : X \to Y$, siis

$$F(g)(w) = \begin{cases} (g(x_1), g(x_2)) & \text{kui } w = (x_1, x_2) \in X \times X; \\ g(x) & \text{kui } w = x \in X; \\ 0 & \text{kui } w = 0 \in 1, \end{cases}$$

on funktor.

Exercises: 2.2

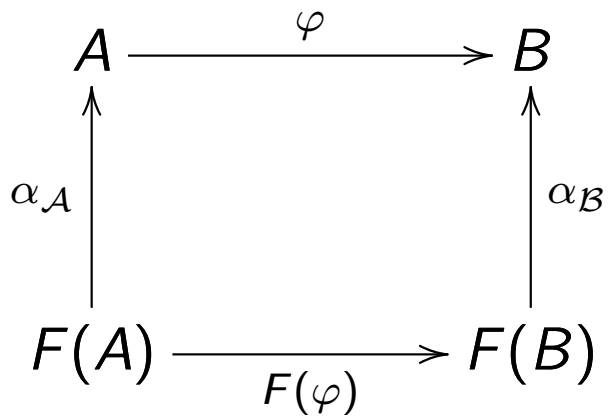
3. Kujutus $\varphi : \mathcal{A} \to \mathcal{B}$ algebrate

$$\mathcal{A} = (A, (\cdot, ^{-1}, 1))$$

ja

$$\mathcal{B} = (B, (\cdot, ^{-1}, 1))$$

vahel on algebrate homomorfism parajasti siis, kui ruut



kommuteerub.

Exercises: 2.2

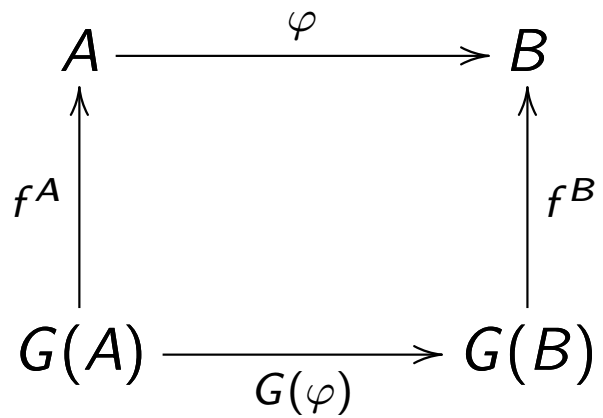
4. Olgu $G(X) = \sum_{i \in I} X^{n_i}$. Kujutus $\varphi : \mathcal{A} \to \mathcal{B}$ algebrate

$$\mathcal{A} = (A, (f_i : A^{n_i} \to A)_{i \in I})$$

ja

$$\mathcal{B} = (B, (f_i : B^{n_i} \to B)_{i \in I})$$

vahel on algebrate homomorfism parajasti siis, kui ruut



kommuteerub.

Exercises: 2.2 lahendus

1. Kokorrutise universaalomaduse kohaselt on morfismid kokorrutisest välja samastatavad morfismidega kokorrutatavatest samasse objekti.
2. Kaunis selge on, et

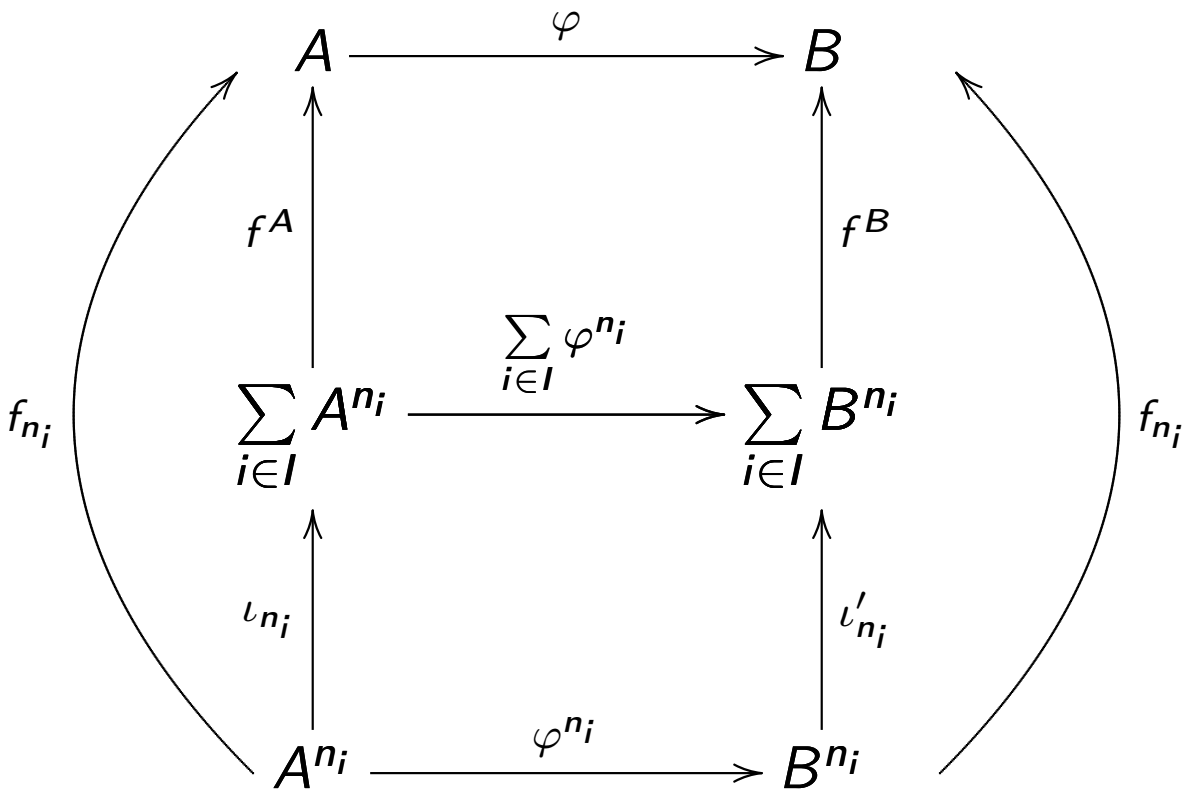
$$F = (- \times -) + - + 1 = + \circ <^2, + \circ < id, 1 >> .$$

3. Erijuht järgmisest.

Exercises: 2.2 lahendus

4. Järgmise diagrammide pere alumised veerandid kommuteeruvad morfismide summa definitsiooni kohaselt, vasakud ja paremad veerandid morfismide f^A ja f^B definitsioonide kohaselt.
Seega piisab välise ruudu kommuteeruvuseks ülemise poole kommuteeruvusest. Summa universaalomaduse kohaselt kehtib $\varphi \circ f^A = f^B \circ \sum_{i \in I} \varphi^{n_i}$ parajasti siis, kui
$\varphi \circ f^A \circ \iota_{n_i} = f^B \circ \sum_{i \in I} \varphi^{n_i} \circ \iota_{n_i}$ iga ι_{n_i} korral. See on aga samaväärne välise ruudu kommuteeruvusega. Kokkuvõttes on diagrammi ülemine veerand kommutatiivne parajasti siis, kui seda on kõik välised ruudud, ehk kui φ on homomorfism algebrate $\mathcal{A}$ ja $\mathcal{B}$ vahel.

Exercises: 2.2 illustratsioon



Exercises: 1.2.20

Exercise 1.2.20 (Lühidalt). Leida funktoritele $\mathbf{1}, A, X, A \times X$ ja $A \times X + 1$ vastavad morfismid, käitumisekvivalents ja lõppkoalgebra.

1) Juht $F = \mathbf{1}$. Morfism $f : (X, x) \to (Y, y)$ rahuldab tingimust $\mathbf{1}(x(z)) = y(f(z))$ iga $z \in X$ jaoks. Eraldatavaid käitumisi on üksainus. Lõppkoalgebra koosnebki sellest.
2) Juht $F = A$. Morfism $f : (X, x) \to (Y, y)$ rahuldab tingimust $x(z) = y(f(z)), z \in X$. Eraldatavad käitumised on eri väljundid hulgas A. Lõppkoalgebra on (A, id).
3) Juht $F = X$. Morfism $f : (X, x) \to (Y, y)$ rahuldab tingimust $f(x(z)) = y(f(z)), z \in X$. Eraldatavaid käitumisi on üksainus. Lõppkoalgebra koosneb sellest.

Exercises: 1.2.20

4) Juht $F = A \times X$. Morfism $f : (X, h, t) \to (Y, h, t)$ rahuldab tingimust $h(f(z)) = f(h(z))$ ja $t(f(z)) = f(t(z))$, $z \in X$. Eraldatavad käitumised on lõpmatud A-jadad. Lõppkoalgebraks on viimaste koalgebra koos oma h, t-funktsioonidega.

5) Juht $F = A \times X + 1$. Morfism $f : (X, h, t) \to (Y, h, t)$ rahuldab tingimust $x = 1 \Leftrightarrow f(x) = 1$ ja $h(f(z)) = f(h(z))$ ning $t(f(z)) = f(t(z))$, $z \in X$. Eraldatavad käitumised koosnevad lõplikest ja lõpmatuist A-jadadest, mis koos oma termineeruvuse ja osaliste h, t-funktsioonidega moodustavad lõppkoalgebra.

Exercises: 1.3.2

Näide 1.3.2 Moore'i automaat on koalgebra

$$\xi : X \to O \times X^I,$$

kus I ja O on sisend- ja väljundtähestikud. Potentsiaalselt iga protsess on antud oma algolekuga $x_o \in X$.

Moore'i automaatide morfism

$$f : (X, out, next) \to (Y, out, next)$$

rahuldab tingimusi $out(x) = out(f(x))$ ja
$f(next(x)(i)) = next(f(x), i)$, $x \in X, i \in I$.

Exercises: 1.3.2

Moore'i automaadi käitumine koosneb lõpmatust puust tüüpi $t : I^* \to O$. Formaalselt, käitumine on $(out(next^*(x_0, w)))_{w \in I^*}$, kus $next^*(x, w) = next(next^*(x, tail^{op}(w)), h^{op}(w))$.

Lõppkoalgebra Moore'i automaatide jaoks on $(O^{I^*}, out, next)$, kus $out(t) = t(<>)$ ja $next(t, i)(w) = t(i + w)$.

Moore'i automaatide bisimulatsioon $\rho \subseteq X \times Y$ on seos, mille korral

$$x \, \rho \, y \Rightarrow out(x) = out(y)$$

ja

$$x \rho y \Rightarrow next(x, i) \, \rho \, next(y, i),$$

$x \in X, y \in Y, i \in I$.

Exercises: 1.3.5

Näide 1.3.5 Mealy' automaat on koalgebra

$$\xi : X \to O^I \times X^I,$$

kus I ja O on sisend- ja väljundtähestikud. Potentsiaalselt on iga protsess jälle antud oma algolekuga $x_o \in X$.

Mealy' automaatide morfism

$$f : (X, out, next) \to (Y, out, next)$$

rahuldab tingimusi $out(x)(i) = out(f(x)(i))$ ja $f(next(x)(i)) = next(f(x), i)$, $x \in X, i \in I$.

Exercises: 1.3.5

Mealy' automaadi käitumine koosneb puust $I^+ \to O$, kus $I^+ = I^* \setminus \{<>\}$.

Lõppkoalgebra Mealy' automaatide jaoks on $(O^{I^+}, out, next)$, kus $out(t)(i) = t(i)$ ja $next(t, i)(w) = t(i + w)$.

The End

Tänan tähelapanu eest!