Staatiline analüüs. Abstraktne interpretatsioon.

klausr@ut.ee

Eesmärk: uurida programmi dünaamilist käitumist (ilma seda käivitamata).

Huvitavad küsimused: kas programm termineerub, kas muutujad on initsialiseeritud jne.

Rice'i teoreem: ei saa kirjutada arvutiprogrammi, mis kontrolliks mingit omadust suvalisel programmil.

Staatilise analüüs püüab lahendada praktilisi probleeme – teha programm kiiremaks, kokkuhoidlikumaks, leida tast vigu.

Otsime ligilähedasi vastuseid.

Vead kallutatame turvalises suunas.

Näide: milliseid väärtusi võib omandada *z*?

```
read(x);if (x>0) then y:=1 else (y:=2;S);z:=y
```

Rakendusi

- Kompileerimisfaasi tehnikad ennustamaks programmi käitumist. Võimaldab kompilaatoreil koodi genereerides liigseid arvutusi vältida.
  - olemasolevate tulemuste taaskasutus
  - tsükli invariantide viimine tsüklist välja
  - tulemusi, mida edasises ei kasutata, ei ole mõtet arvutada
  - tulemusi, mis juba kompileerimise ajal teada samuti mitte
- Valideerimistarkvara soovimatu käitumise piiramiseks.

Analüüsi käik

1. tõlgime programmi võrranditeks mingit omadust iseloomustaval osalisel järjestusel (võrel). Võrrandisüsteemi lahend annab analüüsi korrektse tulemuse. Osaline järjestus iseloomustab lahendite suhtelist täpsust.

2. Arvutame lahendi.

analüüsimeetodid: andmevooanalüüs, juhtvooanalüüs, *type and effect systems*. Abstraktse interpretatsiooni teooria analüüsi korrektsusest keele semantika suhtes.

Võreteooriat

täielik võre on osaliselt järjestatud hulk $L = (S, \sqsubseteq)$, mille igal alamhulgal leidub ülemine ja alumine raja. Edaspidi nimetame lihtsalt võreks.

võre kõrgus on pikim tee vähimast elemendist suurimasse

monotoonne funktsioon $f : L \to L$, mille puhul $\forall x, y \in L : x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$.

püsipunkt monotoonse funktsiooni $f : L \to L$ jaoks on selline $l \in L$, mille korral $f(l) = l$. Funktsiooni kõigi püsipunktide hulka tähistame $Fix(f)$.

Ahelat nimetatakse statsionaarseks, kui leidub selline k, et $\forall i \geq k : x_i = x_k$.

Püsipunkti teoreem. Olgu võre $(L, \subseteq)$, mille kõik kasvavad ahelad on statsionaarsed. Siis leidub iga monotoonsel funktsiooni $f : L \to L$ jaoks selline $k \in \mathbf{N}$, et $f^k(\bot) = f^{k+1}(\bot)$ on funktsiooni f vähim püsipunkt.

Naiivne algoritm püsipunkti leidmiseks: $f^k(\bot)$ otsene väljaarvutamine (ronime ahelat mööda vähima püsipunktini).

Püsipunkti arvutamise ajaline keerukus sõltub kolmest faktorist: võre kõrgus, f-i arvutamise hind, võrdsuse kontrollimise hind

Mõned olulised võred

$(\mathcal{P}(A), \subseteq)$, kus A on mingi lõplik hulk. Siin $\bot = \emptyset$, $\top = A$, $x \sqcup y = x \cup y$ ning $x \sqcap y = x \cap y$.

Lõpliku kõrgusega võrede *korrutis* on lõpliku kõrgusega võre:

$$L_1 \times L_2 \times \ldots \times L_n = \{(x_1, x_2, \ldots, x_n) | \, x_i \in L_i\},$$

kus järjestus, ülem- ning alamraja on defineeritud punktikaupa. Kõrgus on komponentvõrede kõrguste summa.

Summa:

$$L_1 + \ldots + L_n = \{(i, x_i) \mid x_i \in L_i \setminus \{\bot, \top\}\} \cup \{\bot, \top\}.$$

Siin $(i, x) \sqsubseteq (j, y) \Leftrightarrow i = j$ ja $x \sqsubseteq y$. Kõrguseks on maksimaalne komponentide kõrgustest.

Veel üks võre

Võre $A \mapsto L = \{[a_1 \mapsto x_1, \ldots, a_n \mapsto x_n] | x_i \in L\}$ on punktikaupa järjestatud võre, mille kõrgus on $|A| \cdot height(L)$.

Olgu L on lõpliku kõrgusega võre, vaatleme võrrandisüsteemi:

$$\begin{aligned} x_1 &= F_1(x_1, \ldots, x_n) \\ x_2 &= F_2(x_1, \ldots, x_n) \\ &\vdots \\ x_n &= F_n(x_1, \ldots, x_n) \end{aligned}$$

kus $x_i \in L$ on muutujad ja $F_i : L^n \to L$ on monotoonsed funktsioonid.

Süsteemi **ühene vähim lahend** on funktsiooni $F : L^n \to L^n$:

$$F(x_1, \ldots, x_n) = (F_1(x_1, \ldots, x_n), \ldots, F_n(x_1, \ldots, x_n))$$

vähim püsipunkt.

Sarnaselt saab lahendada ka võrratusi, kus võrdusmärki asendab sümbol $\sqsubseteq$, – relatsioon $x \sqsubseteq y$ on ekvivalentne võrdusega $x = x \sqcap y$.

Def. Juhtvoograaf on sidus suunatud graaf $G = (N, E, s, t)$, kus N on lõplik tippude hulk, $E = (N \times N)$ servade hulk, s on algtipp ja t on lõpptipp. Tipud vastavad programmipunktidele, kahe tipu vahel on serv parajasti siis, kui programm saab liikuda ühest avaldisest teise.

Andmevooanalüüs

- Programm kui juhtvoograaf (otseselt tõlgitav)
- Analüüsi *domeen* D – programmi kõikvõimalike seisundite hulk, seisundi mõiste sõltub analüüsist.
- Iga graafi tipuga $n \in N$ saame siduda seisundi $d \in D$.
- Tipu seisund – väide, mis tippu sisenemisel alati kehtib.
- Analüüs – funktsioon graafi tippude hulgast seisundite hulka.

Näide: Algväärtustamata muutujate analüüs

- $D = \mathcal{P}(\mathbf{Var}_*)$
- Otsime funktsiooni $AM = N \to \mathcal{P}(\mathbf{Var}_*)$.
- algseisund $\iota \in D$ – kõigi muutujate hulk
- iga servaga seome *üleminekufunktsiooni* $f_e : D \to D$
- graafi üleminekufunktsioonide komplekt – $tf : E \to (D \to D)$.
- igal sammul viskame väärtustatud muutujad minema

Def. *Teeks* tipust tippu nimetame paarikaupa ühiste otspunktidega servade jada tipust tippu. *Tee semantika* on vastavate üleminekufunktsioonide kompositsioon.

Erinevaid teid pidi laekuva informatsiooni ühendamine toob täpsuse kao.

AM-i otsese arvutamise asemel kasutame ligikaudset funktsiooni MOP (*Merge over all paths*), mis on AM-i suhtes korrektne (kallutame vigu).

Vigade kallutamise poliitika rakendamiseks vajame osalist järjestust. Väiksem domeeni element on informatsiooni mõttes täpsem. AM-i puhul – suurem element pakub rohkem initsialiseerimata muutujaid.

Erinevate teede informatsiooni ühendamiseks (*merge*) võtame ülemise raja. Meie näites $\{a, b\}$ ja $\{a, c\}$ ülemraja on hulkade ühend $\{a, b, c\}$.

Kuna nõuame, et domeen oleks järjestatud ja saaksime alati võtta ülemist raja, siis – domeen D olgu võre!

See, kuidas info ühendamine toimub, sõltub võrest. Nt. võre $\mathcal{P}(\mathcal{P}(\mathbf{Var}_*))$ puhul, olnuks ühendiks hoopis $\{\{a, b\}, \{a, c\}\}$. Meil oli aga tegemist nimetatud võre abstraktsema lähendiga*

*Sääraste lähendite pruukimisele annavad formaalse aluse Galois' vastavused.

$$MOP(n) = \sqcup\{\pi_{tf}(\iota) | \pi_{tf} \text{ on algtipust } s \text{ tipuni } n \text{ kulgeva tee semantika}\}$$

MOP pole otseselt arvutatav – tsükliga programmis on teid lõputult.

Ligikaudseks arvutamiseks koostame rekurrentsete võrrandite süsteemi, mis kirjeldaks seoseid graafi kaarte vahel

$$lfp(n) = \begin{cases} \iota & \text{kui } n = s \\ \sqcup\{tf(e)(lfp(n')) | e = (n', n) \in E\} & \text{vastasel korral} \end{cases}$$

Sellise süsteemi vähim lahend – **vähim püsipunkt** (lfp), on otsitavaks turvaliseks lähendiks MOP-ile.

Teoreem (*Kam ja Ullmann*): Kui üleminekufunktsioon $tf(e)$ on monotoonne iga $e \in E$ korral, siis on iga n korral $lfp(n) \sqsupseteq MOP(n)$. Kui üleminekufunktsioon $tf(e)$ on distributiivne ülemraja võtmise suhtes iga $e \in E$ korral ja domeeni kõik kasvavad ahelad on statsionaarsed, siis iga n korral $MOP(n) = lfp(n)$.

Monotoonsus on loomulik tingimus – ebatäpsemast seisundist ei tohi olla tuletatav täpsem seisund kui on tuletatav täpsemast.

Näide. Siin $f_{i,j}(X) = X \setminus \{v_i\}$, kui programmipunktis i toimub omistamine muutujasse v_i ning $f_{i,j}(X) = X$ muudel juhtudel.

```
x:=0;[1]  while (x<3)[2]  x:=x+1;[3]  (skip)[4]
```

$$
\begin{aligned}
\mathrm{MOP}(1) &= \iota \\
\mathrm{MOP}(2) &= f_{1,2}(\mathrm{MOP}(1)) \sqcup f_{3,2}(\mathrm{MOP}(3)) \\
\mathrm{MOP}(3) &= f_{2,3}(\mathrm{MOP}(2)) \\
\mathrm{MOP}(4) &= f_{2,4}(\mathrm{MOP}(2))
\end{aligned}
$$

Näiteid: bitivektoranalüüsid

$D = \mathcal{P}(A)$ – mugav realiseerida bitivektorina. D on lõpliku kõrgusega, ülem-inekud on distributiivsed.

- Arvutatud avaldiste analüüs (*Available Expressions*)
- Omistusanalüüs (*Reaching Definitions*)
- Vajalike avaldiste analüüs (*Very Busy Expressions*)
- Elusate muutujate analüüs (*Live Variables*)

Omistusanalüüs (reaching definitions)

Kust on nähtavad muutujad oma väärtused saanud?

Domeeniks on kõigi antud programmi omistuslausete hulga kõigi alamhulkade hulk: $D = \mathcal{P}(n \in N | n \text{ vastab omistuslausele})$.

```
1. x:=input          4.   y:=x*y;
2. y:=1;             5.   x:=x-1;
3. while x>1 do (    6. );
```

Olgu $d_n \in D$ juhtvoograafi tipule n vastav seisund ja $pred(n)$ tipu n otseste eellaste hulk. Info ühendamiseks tipus n kasutame järgmist abifunktsiooni:

$$JOIN(n) = \bigcup_{m \in pred(n)} d_m.$$

Omistuslausete puhul rakendame üleminekufunktsiooni

$assignment(n) = JOIN(n) \downarrow id \cup \{n\}$, kus $\downarrow id$ eemaldab kõik omistamised n-i vasakuks pooleks olevasse muutujasse id.

Ülejäänud juhtudel rakendub üleminekufunktsioonina

$default(n) = JOIN(n)$.

Võrrandisüsteem:

$$A(0) = [entry] = \{\}$$
$$A(1) = [x:=input] = A(0) \cup \{x:=input\}$$
$$A(2) = [y:=1] = A(1) \cup \{y:=1\}$$
$$A(3) = [x>1] = A(2) \cup A(5)$$
$$A(4) = [y:=x*y] = A(3) \downarrow y \cup \{y:=x*y\}$$
$$A(5) = [x:=x-1] = A(4) \downarrow x \cup \{x:=x-1\}$$
$$A(6) = [exit] = A(3)$$

Vähim lahend:

$[entry] = \{\}$
$[x:=input] = \{x:=input\}$
$[y:=1] = \{x:=input, y:=1\}$
$[x>1] = \{x:=input, x:=x-1, y:=1, y:=x*y\}$
$[y:=x*y] = \{x:=input, x:=x-1, y:=x*y\}$
$[x:=x-1] = \{x:=x-1, y:=x*y\}$
$[exit] = \{x:=input, x:=x-1, y:=1, y:=x*y\}$

Omistusanalüüsi tulemust võib vaadelda graafina, mis seob punktid, milles muutuja saab väärtuse, nende punktidega, kus seda väärtust kasutada võidakse.

Rakendus: surnud koodi elimineerimine, *code motion*.

Elusate muutujate analüüs (liveness)

```
0.  x:=input;          3.  if (y>0)
1.  y:=input;          4.    then z:=y;
2.  x:=1;              5.    else z:=x*x;
                       6.  x:=z;
```

Milliste muutujate väärtuseid vaadeldavas programmipunktis kasutatakse programmi edasises töös? Domeeniks on võre $D = (2^{\{x,y,z\}}, \subseteq)$.

Üleminekufunktsioonid

Olgu $d_n \in D$ juhtvoograafi tipule n vastav seisund ja $succ(n)$ tipu n otseste järglaste hulk. Info ühendamiseks tipus n kasutame järgmist abifunktsiooni:

$$JOIN(n) = \bigcup_{m \in succ(n)} d_m.$$

Tähistagu $vars(E)$ vaadaldavale tipule n vastavas avaldises E esinevate muutujate hulka;

tähistagu identifikaator id muutujat tipule n vastava omistuslause vasakus pooles;

tähistagu $id_1, \ldots, id_k$ – tipule n vastavas muutujadeklaratsioonis esinevaid muutujaid.

Üleminekud:

$$exit = \{\}$$

$$condition(n) = output(n) = JOIN(n) \cup vars(E)$$

$$assignment(n) = JOIN(n) \setminus \{id\} \cup vars(E)$$

$$declaration(n) = JOIN(n) \setminus \{id_1, \ldots, id_k\}$$

$$default(n) = JOIN(n)$$

Võrrandisüsteem

$$A(0) = [\texttt{x:=input}] = A(1) \setminus \{x\}$$
$$A(1) = [\texttt{y:=input}] = A(2) \setminus \{y\}$$
$$A(2) = [\texttt{x:=1}] = A(3) \setminus \{x\}$$
$$A(3) = [\texttt{y>0}] = A(4) \cup A(5)$$
$$A(4) = [\texttt{z:=y}] = A(6) \setminus \{z\} \cup \{y\}$$
$$A(5) = [\texttt{z:=x*x}] = A(6) \setminus \{z\} \cup \{x\}$$
$$A(6) = [\texttt{x:=z}] = A(7) \setminus \{x\} \cup \{z\}$$
$$A(7) = [\textit{exit}] = \{\}$$

Vähim lahend:

$[\texttt{x:=input}] = \{\}$
$[\texttt{y:=input}] = \{\}$
$[\texttt{x:=1}] = \{y\}$
$[\texttt{y>0}] = \{x, y\}$
$[\texttt{z:=y}] = \{y\}$
$[\texttt{z:=x*x}] = \{x\}$
$[\texttt{x:=z}] = \{z\}$
$[\textit{exit}] = \{\}$

Pisut efektiivsem programm oleks selle analüüsi põhjal selline:

```
/* x:=input ? */      3.  if (y>0)
1.  y:=input;         4.    then x:=y;
2.  x:=1;             5.    else x:=x*x;
```

Arvutatud avaldiste analüüs (*Available expressions*)

```
1.  var x,y,z,a,b;
2.  z:=a+b;
3.  y:=a*b;
4.  while (y>a+b) {
5.    a:=a+1;
6.    x:=a+b;
    }
```

Millised avaldised on juba välja arvutatud?

Domeeniks on avaldiste hulga kõigi alamhulkade hulk, antud juhul

$$D = (2^{\{a+b, a*b, y>a+b, a+1\}}, \supseteq)$$

Üleminekud.

Abifunktsioon $JOIN$:

$$JOIN(n) = \bigcap_{m \in pred(n)} d_m$$

Abifunktsioon $exps$:

$$exps(intconst) = exps(id) = exps(input) = \emptyset$$

$$exps(E_1 \text{op} E_2) = \{E_1 \text{op } E_2\} \cup exps(E_1) \cup exps(E_2)$$

$$entry = \{\}$$

$$condition(n) = output(n) = JOIN(n) \cup exps(E)$$

$$assignment(n) = (JOIN(n)) \cup exps(E)) \downarrow id$$

– siin $\downarrow id$ eemaldab kõik avaldised, mis viitavad muutujale id.

$$default(n) = JOIN(n)$$

Kitsendused:

$$A(0) = [\textit{entry}] = \{\}$$
$$A(1) = [\texttt{var x,y,z,a,b;}] = A(0)$$
$$A(2) = [\texttt{z:=a+b}] = exps(a + b)$$
$$A(3) = [\texttt{y:=a*b}] = (A(6) \cup exps(a * b)) \downarrow y$$
$$A(4) = [\texttt{y>a+b}] = (A(3) \cap A(6)) \cup exps(y > a + b)$$
$$A(5) = [\texttt{a:=a+1}] = (A(4) \cup exps(a + 1)) \downarrow a$$
$$A(6) = [\texttt{x:=a+b}] = (A(5) \cup exps(a + b)) \downarrow x$$
$$A(7) = [\textit{exit}] = A(4)$$

Vähim lahend

[*entry*]$= \{\}$
[var x,y,z,a,b;]$= \{\}$
[z:=a+b]$= \{a + b\}$
[y:=a*b]$= \{a + b, a * b\}$
[y>a+b]$= \{a + b, y > a + b\}$
[a:=a+1]$= \{\}$
[x:=a+b]$= \{a + b\}$
[*exit*]$= \{a + b, y > a + b\}$

Optimeeritud programm:

```
var x,y,z,a,b,aplusb;
aplusb=a+b;
z=aplusb;
y=a*b;
while (y>aplusb) {
  a=a+1;
  aplusb=a+b;
  x=aplusb;
}
```

Vajalike avaldiste analüüs (*Very Busy Expressions*)

Otsime avaldisi, mille väljaarvutatud väärtust programmi edasise töö käigus tingimata vajatakse (arvutatakse korduvalt).

Domeeniks on jällegi $D = (2^{Avaldised}, \supseteq)$.

```
1. var x,a,b;        5. while(x>0) {
2. x:=input;         6.   output a*b-x;
3. a:=x-1;           7.   x:=x-1;   }
4. b:=x-2;           8. output(a*b);
```

Defineerime üleminekufunktsioonid:

$$JOIN(n) = \bigcap_{m \in succ(n)} d_n$$

$$exit = \{\}$$

$$condition(n) = output(n) = JOIN(n) \cup exps(E)$$

$$assignment(n) = JOIN(n) \downarrow id \cup exps(E)$$

$$default(n) = JOIN(n)$$

Analüüs selgitab, et tsüklikehas arvutatavat avaldist $a * b$ läheb edaspidi alati tarvis, seega on parem programm selline:

```
1. var x,a,b,atimesb;
2. x:=input;
3. a:=x-1;
4. b:=x-2;
5. atimesb:=a*b;
6. while(x>0) {
7.   output atimesb-x;
8.   x:=x-1;   }
9. output(atimesb);
```

Bitivektoranalüüside klassifikatsioon

- edasisuunalised – arvutatakse juhtvoograafi tipu esivanemate järgi
- tagasisuunalised – arvutatakse juhtvoograafi tipu järeltulijate järgi
- ülemised lähendid (*may*) – kirjeldavad asju, mis võivad olla tõesed
- alumised lähendid (*must*) – kirjeldavad asju, mis on kindlasti tõesed

Protseduuridevaheline analüüs

Seni tegelesime protseduurisisese analüüsiga. Kuidas viia läbi protseduuride-vahelist analüüsi ehk kuidas toimida, kui laiendame keelt funktsioonidega?

- Juhtvoograaf moodustatakse funktsioonide juhtvoograafidest.
- Kasutame varimuutujaid `ret-f`, `call-i`, kus `i` on unikaalne indeks
- Kohalikud muutujad salvestame funktsiooniväljakutse ajaks – `save-i-x:=x`, et väärtused naasmisel taastada.

- funktsiooni argumentide tarvis toome sisse ajutised muutujad `temp-i-x`

- kasu võib olla sellest, kui tekitada iga väljakutse tarvis eraldi alamgraaf

Kontrollvooanalüüs

Kui toome keelde sisse kõrgemat järku funktsioonid, objektid või funktsioonividad, siis senine olukord, kus info programmiplokkide vaheliste seoste kohta on vahetult olemas, kaob.

Näide: järgnevas programmis pole väga ilmne, millise programmilõigu käivitab `x 1` rakendamine.

```
let  f = fn x => x 1;
     g = fn y => y+2;
     h = fn z => z+3;
in   (f g) + (f h)
```

Kontrollvooanalüüsi eesmärgiks on selgitada, millised elementaarblokid millistele viivad.

Antud juhul – tuvastada iga alamavaldise puhul see funktsioonide hulk, millega avaldis võib väärtustuda.

Rohkem me siinkohal juhtvooanalüüsi ei puuduta.

Püsipunktide lähendused

Intervallvõre. Olgu

$$\mathbf{Interval} = \{\bot\} \cup \{[z_1, z_2] | z_1 \leq z_2, z_1 \in \mathbb{Z} \cup \{-\infty\}, z_2 \in \mathbb{Z} \cup \{\infty\}\},$$

kus $\leq$ on defineeritud loomulikul viisil ning $\bot$ on tühi vahemik. Võre ($\mathbf{Interval}, \sqsubseteq$) nimetame **intervallvõreks**, edaspidi rakendame seda mitmes näites.

$$\inf(int) = \begin{cases} \infty & \text{kui } int = \bot \\ z_1 & \text{kui } int = [z_1, z_2] \end{cases}$$

$$\sup(int) = \begin{cases} -\infty & \text{kui } int = \bot \\ z_2 & \text{kui } int = [z_1, z_2] \end{cases}$$

$int_1 \sqsubseteq int_2$ parajasti siis, kui $\inf(int_2) \leq \inf(int_1)$ ja $\sup(int_1) \leq \sup(int_2)$

Programmi p toime omaduse l_1 muutmisel omaduseks l_2 ($p \vdash l_1 \triangleright l_2$), antakse tavaliselt monotoonse funktsiooni $f(l_1) = l_2$ abil.

Otsime vähimat püsipunkti $lfp(f)$, kuid – $(f^n(\bot))_n$ ei pruugi stabiliseeruda ja tema ülemraja ei pruugi olla tingimata $lfp(f)$. Näiteks äsjavaadeldud intervallvõres ei õnnestu püsipunktiteoreemi rakendada – algoritm ei pruugi tööd lõpetada.

Vajame lähendit püsipunktile.

Turvalise lähendi annaks muuhulgas iteratiivse jada $(f^n(\top))_n$ suvaline liige.

Funktsioon $f : L \to L$ võrel L on *kahanev* elemendil $l \in L$ parajasti siis kui $f(l) \sqsubseteq l$ (*kasvav* kui $f(l) \sqsupseteq l$).

$Red(f)$ – kahanemispiirkond

$Ext(f)$ – kasvamispiirkond.

Tarski teoreem: Kui $f : L \to L$ on monotoonne, siis

$$lfp(f) = \sqcap Red(f) \in Fix(f),$$

$$gfp(f) = \sqcup Ext(f) \in Fix(f).$$

Laiendusoperaatorid

Üritame asendada jada $(f^n(\bot))_n$ jadaga $(f^n_\nabla)_n$, mis stabiliseeruks ja annaks turvalise lähendi vähimale püsipunktile.

Operaatorit ∇ nimetame *laiendusoperaatoriks.*

Defineerime esiteks *ülemtõkkeoperaatori* – nõnda nimetatakse operaatorit $\breve{\sqcup} : L \times L \to L$, mille puhul

$$\forall l_1, l_2 \in L : \; l_1 \sqsubseteq (l_1 \breve{\sqcup} l_2) \sqsupseteq l_2.$$

Olgu $(l_n)_n$ jada L-i elementidest ning $\phi : L \times L \to L$ kõikjal defineeritud funktsioon. Konstrueerime uue jada $(l_n^\phi)_n$ järgnevalt:

$$l_n^\phi = \begin{cases} l_n & \text{kui } n = 0 \\ l_{n-1}^\phi \phi l_n & \text{kui } n > 0 \end{cases} .$$

Väide. Kui $(l_n)_n$ on jada ja $\check{\sqcup}$ on ülemtõkkeoperaator, siis $(l_n^{\check{\sqcup}})_n$ on kasvav ahel ja $l_n^{\check{\sqcup}} \sqsupseteq \sqcup\{l_0, l_1, \ldots, l_n\}$, mistahes n korral.

Näide. Muudame intervallvõre kasvavaks ahelaks. Olgu int suvaline fikseeritud element hulgast $\mathbf{Interval}$. Defineerime:

$$int_1 \breve{\sqcup}^{int} int_2 = \begin{cases} int_1 \sqcup int_2 & \text{kui } int_1 \sqsubseteq int \vee int_2 \sqsubseteq int_1 \\ [-\infty, \infty] & \text{muudel juhtudel} \end{cases}$$

Ilmselt on tegemist ülemtõkkeoperaatoriga. Rakendame jadale

$$[0,0], [1,1], [2,2], \ldots$$

seda operaatorit, valides $int = [0, \infty]$. Saame kasvava ahela, mis ei stabiliseeru.

Samas, valides $int = [0, 2]$, saame kasvava ahela, mis stabiliseerub.

Operaator $\nabla : L \times L \to L$ on laiendamisoperaator parajasti siis kui ta on ülemtõkkeoperaator ning mistahes kasvava jada $(l_n)_n$ puhul on $(l_n^{\nabla})_n$ kasvav ja stabiliseerub.

Olgu $f : L \to L$ monotoonne ja olgu ∇ laiendamisoperaator. Siis saame arvutada jada $(f_{\nabla}^n)_n$, mis on defineeritud võrdusega

$$f_{\nabla}^n = \begin{cases} \bot & \text{kui } n = 0 \\ f_{\nabla}^{n-1} & \text{kui } n > 0 \wedge f(f_{\nabla}^{n-1}) \sqsubseteq f_{\nabla}^{n-1} \\ f_{\nabla}^{n-1} \nabla f(f_{\nabla}^{n-1}) & \text{muudel juhtudel} \end{cases} .$$

See on kasvava ahel, saab tõestada, et stabiliseerub.

Selgub, et mingi m-i korral

$$f(f_{\nabla}^{m}) \sqsubseteq f_{\nabla}^{m}.$$

Seega on f kahanev punktis f_{∇}^{m} ja Tarski teoreemi põhjal $f_{\nabla}^{m} \sqsupseteq lfp(f)$.

Võtame otsitud turvaliseks püsipunkti lähendiks

$$lfp_{\nabla}(f) = f_{\nabla}^{m}.$$

Kitsendusoperaatorid

Teades, et $f(f^m_\nabla) \sqsubseteq f^m_\nabla \in Red(f)$, otsime vahendit laiendamisel saadud tulemuse parandamiseks.

Pole põhjust arvata, et see jada stabiliseeruks.

Def. Nimetame operaatorit $\triangle : L \to L$ kitsendusoperaatoriks parajasti siis kui

- $\forall l_1, l_2 \in L : \; l_2 \sqsubseteq l_1 \Rightarrow l_2 \sqsubseteq (l_1 \triangle l_2) \sqsubseteq l_1$

- mistahes kahaneva ahela $(l_n)_n$ puhul jada $(l_n^\triangle)_n$ stabiliseerub.

Eeldades, et $f^m_\triangle$ rahuldab tingimust $f(f^m_\triangle) \sqsubseteq f^m_\triangle$, konstrueerime jada

$$[f]^n_\triangle = \begin{cases} f^m_\triangle & \text{kui } n = 0 \\ [f]^{n-1}_\triangle \triangle f([f]^{n-1}_\triangle) & \text{kui } n > 0 \end{cases}$$

See on kahanev ahel, mille kõik elemendid rahuldavad tingimust $lfp(f) \sqsubseteq [f]^n_\triangle$. Saab tõestada, et ta stabiliseerub – mingi väärtuse m' korral $[f]^{m'}_\triangle = [f]^{m'+1}_\triangle$.

Võtame otsitud lähendiks

$$lfp^\triangle_\nabla(f) = [f]^{m'}_\triangle.$$

Galois’ vastavused

Arvutused võrel L võivad olla liialt kulukad või teostamatud – sel puhul on põhjust kolida lihtsamale võrele M.

Näide: $\mathcal{P}(\mathbb{Z})$ asendamine intervallvõrega.

L-i elementide esitamiseks M-is rakendame *abstraktsioonifunktsiooni*

$$\alpha : L \to M$$

M-i elementide esitamiseks L-is rakendame *konkretiseerimisfunktsiooni*

$$\gamma : M \to L.$$

Def. (L, α, γ, M) on *Galois' vastavus* võrede L ja M vahel parajasti siis kui α ja γ on monotoonsed funktsioonid, mis rahuldavad tingimusi

$$\gamma \circ \alpha \sqsupseteq \lambda l.l$$

$$\alpha \circ \gamma \sqsubseteq \lambda m.m$$

Näide. Galois' vastavus $(\mathcal{P}(\mathbb{Z}), \alpha_{ZI}, \gamma_{ZI}, \mathbf{Interval})$.

$$\gamma_{ZI}(int) = \{z \in \mathbb{Z} \mid \inf(int) \leq z \leq \sup(int)\}$$

$$\alpha_{ZI}(Z) = \begin{cases} \bot & \text{kui } Z = \emptyset \\ [\inf'(Z),\ \sup'(Z)] & \text{muudel juhtudel} \end{cases}$$

Def. Ütleme, et (L, α, γ, M) on *adjunktsioon* parajasti siis kui α ja γ on kõikjal määratud rahuldades mistahes $l \in L$ ja $m \in M$ korral tingimust

$$\alpha(l) \sqsubseteq m \Leftrightarrow l \sqsubseteq (m).$$

Lause. (L, α, γ, M) on adjunktsioon parajasti siis kui ta on Galois' vastavus.

Galois' vastavused tuletusfunktsioonide kaudu

Olgu meil *esitusfunktsioon* $\beta : V \to L$, kus V on mingi semantiline hulk ja L selle elemente kirjeldav võre. Kui defineerime $V' \subseteq V$ ja $l \in L$ jaoks

$$\alpha(V') = \bigsqcup\{\beta(v) | v \in V'\}$$

$$\gamma(l) = \{v \in V | \beta(v) \sqsubseteq l\}$$

saame Galois' vastavuse $(\mathcal{P}(V), \alpha, \gamma, L)$.

Seejuures $\alpha(\{v\}) = \beta(v)$.

Juhul kui $L = (\mathcal{P}(D), \subseteq)$ mingi hulga D korral, nimetame funktsiooni $\eta : V \to D$ *tuletusfunktsiooniks (extraction function).*

Defineerime $\beta_\eta : V \to \mathcal{P}(D)$ nõnda: $\beta_\eta(v) = \{\eta(v)\}$ ja saame Galois' vastavuse

$$(\mathcal{P}(V), \alpha_\eta, \gamma_\eta, \mathcal{P}(D)),$$

defineerides

$$\alpha_\eta(V') = \bigcup\{\beta_\eta(v) | v \in V'\} = \{\eta(v) | v \in V'\}$$

$$\gamma_\eta(D') = \{v \in V | \beta_\eta(v) \subseteq D'\} = \{v | \eta(v) \in D'\}$$

kus $V' \in V$ ja $D' \in D$.

Näide. Vaatleme võresid $(\mathcal{P}(\mathbb{Z}), \subseteq)$ ja $(\mathcal{P}(\mathbf{Sign}), \subseteq)$, kus $\mathbf{Sign} = \{-, 0, +\}$. Tuletusfunktsioon $sign = \mathbb{Z} \to \mathbf{Sign}$ olgu defineeritud nõnda:

$$sign(z) = \begin{cases} - & \text{kui } z < 0 \\ 0 & \text{kui } z = 0 \\ + & \text{kui } z > 0 \end{cases}$$

Võttes

$$\alpha_{sign}(Z) = \{sign(z) | z \in Z\}$$

$$\gamma_{sign}(S) = \{z \in \mathbb{Z} | sign(z) \in S\}$$

kus $Z \subseteq \mathbb{Z}$ ja $S \subseteq \mathbf{Sign}$, saame Galois' vastavuse

$$(\mathcal{P}(\mathbb{Z}), \alpha_{sign}, \gamma_{sign}, \mathcal{P}(\mathbf{Sign})).$$

Lemma: kui (L, α, γ, M) on Galois' vastavus, siis

- α määrab üheselt γ: $\gamma(m) = \sqcup\{l|\alpha(l) \sqsubseteq m\}$ ja γ määrab üheselt α: $\alpha(l) = \sqcap\{m|l \sqsubseteq \gamma(m)\}$.

- α on täielikult aditiivne ja γ on täielikult multiplikatiivne.

- $\alpha(\bot) = \bot$ ja $\gamma(\top) = \top$.

Lemma: Kui α on täielikult aditiivne, siis leidub γ nii et (L, α, γ, M) on Galois' vastavus. Analoogselt – kui γ on täielikult multiplikatiivne, siis leidub α nii et (L, α, γ, M) on Galois' vastavus.

Väide: Kui (L, α, γ, M) on Galois' vastavus, siis $\alpha \circ \gamma \circ \alpha = \alpha$ ja $\gamma \circ \alpha \circ \gamma = \gamma$.

Näide: Vaatleme võresid $\mathbf{Interval}$ ja $\mathcal{P}(\mathbf{Sign})$. Defineerime

$$\begin{array}{cc} \gamma_{\mathrm{IS}}(\{-,0,+\}) = [-\infty,\infty] & \gamma_{\mathrm{IS}}(\{-,0\}) = [-\infty,0] \\ \gamma_{\mathrm{IS}}(\{-,+\}) = [-\infty,\infty] & \gamma_{\mathrm{IS}}(\{0,+\}) = [0,\infty] \\ \gamma_{\mathrm{IS}}(\{-\}) = [-\infty,-1] & \gamma_{\mathrm{IS}}(\{0\}) = [0,0] \\ \gamma_{\mathrm{IS}}(\{+\}) = [1,\infty] & \gamma_{\mathrm{IS}}(\emptyset) = \bot \end{array}$$

Otsime vastupidist funktsiooni α_{IS}, nii et tekiks Galois' vastavus.

Kontrollime γ_{IS} multiplikatiivsust.

Lemma: Kui L ja M on võred, ja M on lõplik, siis väide, et

1. $\gamma : M \to L$ on monotoonne

2. $\gamma(\top) = \top$

3. $\gamma(m_1 \sqcap m_2) = \gamma(m_1) \sqcap \gamma(m_2)$ parajasti siis kui $m_1 \not\sqsubseteq m_2$ ja $m_2 \not\sqsubseteq m_1$

on ekvivalentne väitega: $\gamma : M \to L$ on täielikult multiplikatiivne.

Esimesed kaks tingimust on γ_{IS} puhul ilmselt täidetud, aga kolmanda kontroll paaril $(\{-, 0\}, \{-, +\})$ annab negatiivse tulemuse – järelikult ei leidu sellist funktsiooni α_{IS}, et $(\mathbf{Interval}, \alpha_{\mathrm{IS}}, \gamma_{\mathrm{IS}}, \mathcal{P}(\mathbf{Sign}))$ oleks Galois' vastavus.

Korrektsus ja Galois' vastavused

Olgu $R : V \times L \to \{t, v\}$ korrektsusrelatsioon V ja L-i vahel. Asendades V suhtes korrektse võre L võrega M, soovime, et ka M oleks V suhtes korrektne.

Kui meil on Galois' vastavus (L, α, γ, M), saame konstrueerida korrektsusrelatsiooni V ja M vahele: $S : V \times M \to \{t, v\}$ defineerime

$$v \, S \, m \text{ parajasti siis kui } v \, R \, (\gamma(m)).$$

Lähtudes sellest, et $(v\,R\,l_1) \wedge l_1 \sqsubseteq l_2 \Rightarrow v\,R\,l_2$, saame näidata, et

$$(v\,S\,m_1) \wedge m_1 \sqsubseteq m_2 \Rightarrow v\,S\,m_2.$$

Samamoodi, lähtudes sellest, et $(\forall l \in L' \subseteq L : v\,R\,l) \Rightarrow v\,R\,(\sqcap L')$, saame, et

$$(\forall m \in M' \subseteq M : v\,S\,m) \Rightarrow v\,S\,(\sqcap M').$$

Galois' sisestused

Nimetame (L, α, γ, M) *Galois' sisestuseks* parajasti siis, kui α ja γ on monotoonsed funktsioonid, mis rahuldavad tingimusi

$$\gamma \circ \alpha \sqsupseteq \lambda l.l$$

$$\alpha \circ \gamma = \lambda m.m$$

Näide.

$$(\mathcal{P}(\mathbb{Z}), \alpha_{ZI}, \gamma_{ZI}, \mathbf{Interval})$$

on Galois' sisestus.

Lemma. Galois' vastavuse (L, α, γ, M) jaoks on ekvivalentsed väited:

- (L, α, γ, M) on Galois' sisestus.
- α on sürjektiivne – $\forall m \in M : \exists l \in L : \alpha(l) = m$
- γ on injektiivne – $\forall m_1, m_2 \in M : \gamma(m_1) = \gamma(m_2) \Rightarrow m_1 = m_2$
- γ on järjestust säilitav – $\forall m_1, m_2 \in M : \gamma(m_1) \sqsubseteq \gamma(m_2) \Leftrightarrow m_1 \sqsubseteq m_2$

Näide. Vaatleme võresid $(\mathcal{P}(\mathbb{Z}), \subseteq)$ ja $(\mathcal{P}(\mathbf{Sign} \times \mathbf{Parity}), \subseteq)$, kus $\mathbf{Sign} = \{-, 0, +\}$ ja $\mathbf{Parity} = \{\text{odd}, \text{even}\}$. Defineerime tuletusfunktsiooni $signparity$: $\mathbb{Z} \to \mathbf{Sign} \times \mathbf{Parity}$:

$$signparity(z) = \begin{cases} (sign(z), odd) & \text{kui } z \text{ on paaritu arv} \\ (sign(z), even) & \text{kui } z \text{ on paarisarv} \end{cases} .$$

Paar $(0, odd)$ ei kirjelda ainsatki täisarvude hulga elementi, seega ei ole $signparity$ sürjektiivne ja $(\mathcal{P}(\mathbb{Z}), \alpha_{signparity}, \gamma_{signparity}, \mathcal{P}(\mathbf{Sign} \times \mathbf{Parity}))$ pole Galois' sisestus.

Sisestuse konstrueerimine.

Lause. Olgu (L, α, γ, M) Galois' vastavus ja olgu $\varsigma : M \to M$ reduktsioonioperaator:

$$\varsigma(m) = \sqcap\{m' \mid \gamma(m) = \gamma(m')\}$$

Siis $\varsigma[M] = (\{\varsigma(m) \mid m \in M\}, \sqsubseteq_M)$ on võre ja $(L, \alpha, \gamma, \varsigma[M])$ on Galois' sisestus.

Olgu Galois' vastavus $(\mathcal{P}(V), \alpha_\eta, \gamma_\eta, \mathcal{P}(D))$ antud mingi tuletusfunktsiooni $\eta : V \to D$ abil ja olgu reduktsioonioperaator

$$\varsigma_\eta(D') = D' \cap \eta[V],$$

kus $\eta[V] = \{\eta(v) | v \in V\}$ on η kujutis ja $\varsigma_\eta[\mathcal{P}(D)]$ on isomorfne hulgaga $\mathcal{P}(\eta(D))$. Tulemuseks saadav Galois' sisestus on isomorfne nelikuga $(\mathcal{P}(V), \alpha_\eta, \gamma_\eta, \mathcal{P}(\eta[V]))$.

Näide. Rakendades nüüd seda tehnikat viimases näites, saame

$$signparity[\mathbb{Z}] = \{(-, \text{odd}), (-, \text{even}), (0, \text{even}), (+, \text{even}), (+, \text{odd})\},$$

ja seega on $(\mathcal{P}(\mathbb{Z}), \alpha_{signparity}, \gamma_{signparity}, \mathcal{P}(\mathbb{Z}))$ Galois' sisestus.

Galois' vastavuste süstemaatiline konstrueerimine.

Järjestikkompositsioon.

Liigume lähtevõrelt L_0 aina uutele lähendvõredele $L_1, L_2, \ldots$, kuni jõuame rahuldavate arvutuslike omadusteni.

Kui $(L_{i-1}, \alpha_i, \gamma_i, L_i)$ ja $(L_i, \alpha_{i+1}, \alpha_{i+1}, L_{i+1})$ on Galois' vastavused, siis on ka

$$(L_{i-1}, \alpha_{i+1} \circ \alpha_i, \gamma_i \circ \gamma_{i+1}, L_{i+1})$$

Galois' vastavus.

Näide. Massiivi piiride analüüsi komponent, mis arvutab kahe arvu suurusjärkude erinevust. Esimene samm:

$$(\mathcal{P}(\mathbb{Z}\times\mathbb{Z}), \alpha_{\text{diff}}, \gamma_{\text{diff}}, \mathcal{P}(\mathbb{Z})),$$

$$\text{diff}(z_1, z_2) = |z_1| - |z_2|,$$

$$\alpha_{\text{diff}}(ZZ) = \{|z_1| - |z_2| \,|\, (z_1, z_2) \in ZZ\},$$

$$\gamma_{\text{diff}}(Z) = \{(z_1, z_2) |\, |z_1| - |z_2| \in Z\}.$$

Teine samm, siin $\mathbf{Range} = \{$<-1, -1, 0, +1, >+1$\}$:

$$(\mathcal{P}(\mathbb{Z}), \alpha_{\text{range}}, \gamma_{\text{range}}, \mathcal{P}(\mathbf{Range}),$$

$$\text{range}(z) = \begin{cases} \text{<-1} & \text{kui } z < -1 \\ \text{-1} & \text{kui } z = -1 \\ \text{0} & \text{kui } z = 0 \\ \text{+1} & \text{kui } z = 1 \\ \text{>+1} & \text{kui } z > 1 \end{cases},$$

$$\alpha_{\text{range}}(Z) = \{\text{range}(z) \mid z \in Z\},$$

$$\gamma_{\text{range}}(R) = \{z \mid \text{range}(z) \in R\}.$$

Kompositsioon:

$$(\mathcal{P}(\mathbb{Z}\times\mathbb{Z}), \alpha_R, \gamma_R, \mathcal{P}(\mathbf{Range})),$$

$$\text{range}(z_1, z_2) = |z_1| - |z_2|,$$

$$\alpha_R(ZZ) = \alpha_{\text{range}} \circ \alpha_{\text{diff}} = \{\text{range}(|z_1| - |z_2|) \mid (z_1, z_2) \in ZZ\},$$

$$\gamma_R(R) = \gamma_{\text{diff}} \circ \gamma_{\text{range}} = \{(z_1, z_2) \mid \text{range}(|z_1| - |z_2|) \in R\}.$$

Kombineerimistehnikate kataloog

Struktuuri komponentide analüüsid liidetakse struktuuri analüüsiks (sõltumatute atribuutide meetod, relatsiooniline meetod).

Täielike funktsioonide ruumidele ja monotoonsete funktsioonide ruumidele saab leida lähendeid Galois' vastavuste abil.

Sama struktuuri analüüside kombineerimine üheks analüüsiks (otsekorrutis, tensorkorrutis, redutseeritud korrutised).

Kui meil on olemas mingi hulk baasanalüüse, mille korrektsus on tuvastatud, saame nendest konstrueerida keerulisemaid, kasutades nimetatud tehnikaid.

Kombineerimine komponenthaaval

Sõltumatute atribuutide meetod.

Olgu $(L_1, \alpha_1, \gamma_1, M_1)$ ja $(L_2, \alpha_2, \gamma_2, M_2)$ Galois' vastavused. Siis on Galois' vastavus ka $(L_1 \times L_2, \alpha, \gamma, M_1 \times M_2)$, kus

$$\alpha(l_1, l_2) = (\alpha_1(l_1), \alpha_2(l_2)),$$

$$\gamma(m_1, m_2) = (\gamma_1(m_1), \gamma_2(m_2)).$$

Näide. Massiivi tõkete analüüsi komponent, mis otsib täisarvupaari võimalikke märke. Lähtume Galois' vastavusest

$$(\mathcal{P}(\mathbb{Z}), \alpha_{sign}, \gamma_{sign}, \mathcal{P}(\mathbf{Sign})).$$

Rakendame seda kummalegi arvupaari komponendile, selleks kombineerime vastavust iseendaga ja saame

$$(\mathcal{P}(\mathbb{Z}) \times \mathcal{P}(\mathbb{Z}), \alpha_{SS}, \gamma_{SS}, \mathcal{P}(\mathbf{Sign}) \times \mathcal{P}(\mathbf{Sign})),$$

$$\alpha_{SS}(Z_1, Z_2) = (\{\text{sign}(z) \mid z \in Z_1\}, \{\text{sign}(z) \mid z \in Z_2\}),$$

$$\gamma_{SS}(S_1, S_2) = (\{z \mid \text{sign}(z) \in S_1\}, \{z \mid \text{sign}(z) \in S_2\}).$$

Relatsiooniline meetod. Piirdume juhuga, kus võredeks on potentshulgad.

Olgu $(\mathcal{P}(V_1), \alpha_1, \gamma_1, \mathcal{P}(D_1))$ ja $(\mathcal{P}(V_2), \alpha_2, \gamma_2, \mathcal{P}(D_2))$ Galois' vastavused. Siis on Galois' vastavus ka $(\mathcal{P}(V_1 \times V_2), \alpha, \gamma, \mathcal{P}(D_1 \times D_2))$, kus

$$\alpha(VV) = \bigcup\{\alpha_1(\{v_1\}) \times \alpha_2(\{v_2\}) \mid (v_1, v_2) \in VV\},$$

$$\gamma(DD) = \{(v_1, v_2) \mid \alpha_1(\{v_1\}) \times \alpha_2(\{v_2\}) \subseteq DD\},$$

$VV \subseteq V_1 \times V_2$ ja $DD \subseteq D_1 \times D_2$.

Näide. Täpsem analüüs eelmise näite probleemi tarvis:

$$(\mathcal{P}(\mathbb{Z}\times\mathbb{Z}), \alpha_{SS'}, \gamma_{SS'}, \mathcal{P}(\mathbf{Sign}\times\mathbf{Sign})),$$

$$\alpha_{SS'}(ZZ) = \{(\text{sign}(z_1),\ \text{sign}(z_2) \mid (z_1, z_2) \in ZZ\},$$

$$\gamma_{SS'}(SS) = \{(z_1, z_2) \mid \text{sign}(z_1), \text{sign}(z_2)) \in SS\}.$$

Täielike funktsioonide ruum

Kui L on võre ja S mingi hulk, siis on võre ka $S \to L$.

Kui (L, α, γ, M) on Galois' vastavus, siis on Galois' vastavus ka

$$(S \to L, \alpha', \gamma', S \to M),$$

kus

$$\alpha'(f) = \alpha \circ f,$$

$$\gamma'(g) = \gamma \circ g$$

Monotoonsete funktsioonide ruum

Olgu $(L_1, \alpha_1, \gamma_1, M_1)$ ja $(L_2, \alpha_2, \gamma_2, M_2)$ Galois' vastavused. Võttes

$$\alpha(f) = \alpha_2 \circ f \circ \gamma_1,$$

$$\gamma(g) = \gamma_2 \circ g \circ \alpha_1$$

saame Galois' vastavuse

$$(L_1 \to L_2, \alpha, \gamma, M_1 \to M_2).$$

Kombineerime analüüse, mis tegelevad samade andmetega.

Def. Olgu $(L, \alpha_1, \gamma_1, M_1)$ ja $(L, \alpha_2, \gamma_2, M_2)$ Galois' vastavused. Nende *otsekorrutiseks* nimetame Galois' vastavust

$$(L, \alpha, \gamma, M_1 \times M_2),$$

kus

$$\alpha(l) = (\alpha_1(l), \alpha_2(l)),$$

$$\gamma(m_1, m_2) = \gamma_1(m_1) \sqcap \gamma_2(m_2).$$

Näide. Kombineerime märgianalüüsi suurusjärgu analüüsiga.

$$(\mathcal{P}(\mathbb{Z}\times\mathbb{Z}, \alpha_{SSR}, \gamma_{SSR}, \mathcal{P}(\mathbf{Sign}\times\mathbf{Sign})\times\mathcal{P}(\mathbf{Range})),$$

$$\begin{aligned}\alpha_{\text{SSR}}(ZZ) = \; & (\{(\text{sign}(z_1), \text{sign}(z_2)) \mid (z_1, z_2) \in ZZ\},\\ & \{\text{range}(|z_1| - |z_2|) \mid (z_1, z_2) \in ZZ\}),\end{aligned}$$

$$\begin{aligned}\gamma_{\text{SSR}}(SS, R) = \; & \{(z_1, z_2) \mid (\text{sign}(z_1), \text{sign}(z_2)) \in SS\}\\ & \cap\{(z_1, z_2) \mid \text{range}(|z_1| - |z_2|) \in R\}.\end{aligned}$$

Olgu $(\mathcal{P}(V), \alpha_i, \gamma_i, \mathcal{P}(D_i))$, $i = 1, 2$ Galois' vastavused. Nende tensorkorrutiseks (*direct tensor product*) nimetame Galois' vastavust

$$(\mathcal{P}(V), \alpha, \gamma, \mathcal{P}(D_1 \times D_2)),$$

kus

$$\alpha(V') = \bigcup\{\alpha_1(\{v\}) \times \alpha_2(\{v\}) \mid v \in V'\},$$

$$\gamma(DD) = \{v \mid \alpha_1(\{v\}) \times \alpha_2(\{v\}) \subseteq DD\}.$$

Sedagi tulemust saab üldistada juhule, kus tegemist pole tingimata potentshulkadega.

Näide. Rakendades viimases näites tensorkorrutist, saame vastavuse

$$(\mathcal{P}(\mathbb{Z} \times \mathbb{Z}, \alpha_{SSR}, \gamma_{SSR}, \mathcal{P}(\mathbf{Sign} \times \mathbf{Sign} \times \mathbf{Range})),$$

$$\alpha_{\mathrm{SSR}'}(ZZ) = \{(\mathrm{sign}(z_1), \mathrm{sign}(z_2), \mathrm{range}(|z_1| - |z_2|) \mid (z_1, z_2) \in ZZ\},$$

$$\gamma_{\mathrm{SSR}'}(SSR) = \{(z_1, z_2) \mid (\mathrm{sign}(z_1), \mathrm{sign}(z_2), \mathrm{range}(|z_1| - |z_2|)) \in SSR\}.$$

Olgu $(L, \alpha_1, \gamma_1, M_1)$ ja $(L, \alpha_2, \gamma_2, M_2)$ Galois' vastavused. Nende *redutseeritud korrutiseks* nimetame Galois' sisestust

$$(L, \alpha, \gamma, \varsigma[M_1 \times M_2]),$$

kus

$$\alpha(l) = (\alpha_1(l), \alpha_2(l)),$$

$$\gamma(m_1, m_2) = \gamma_1(m_1) \sqcap \gamma_2(m_2),$$

$$\varsigma(m_1, m_2) = \sqcap\{(m_1', m_2') \mid \gamma_1(m_1) \sqcap \gamma_2(m_2) = \gamma_1(m_1' \sqcap \gamma_2(m_2')\}.$$

Olgu $(\mathcal{P}(V), \alpha_i, \gamma_i, \mathcal{P}(D_i))$ Galois' vastavused. Nende *redutseeritud tensorkorrutiseks* nimetame Galois' sisestust

$$(\mathcal{P}(V), \alpha, \gamma, \varsigma[\mathcal{P}(D_1 \times D_2)]),$$

kus

$$\alpha(V') = \bigcup\{\alpha_1(\{v\}) \times \alpha_2(\{v\}) \mid v \in V'\},$$

$$\gamma(DD) = \{v \mid \alpha_1(\{v\}) \times \alpha_2(\{v\}) \subseteq DD\},$$

$$\varsigma(DD) = \bigcap\{DD' \mid \gamma(DD) = \gamma(DD')\}.$$

Näide. Vaatame veelkord võret $\mathcal{P}(\mathbf{Sign} \times \mathbf{Sign} \times \mathbf{Range})$, kus $\mathcal{P}(\mathbb{Z} \times \mathbb{Z})$ kirjeldamiseks on liiga palju elemente ($\emptyset$, $\{(0, 0, \text{<-1})\}$, jne.).

Defineerime

$$\varsigma_{\text{SSR}'}(SSR) = \bigcap\{SSR' \mid \gamma_{\text{SSR}'}(SSR) = \gamma_{\text{SSR}'}(SSR')\}.$$

Kui esialgses võres on 45 elementi, siis redutseeritud võresse $\mathcal{P}(\mathbf{AB})$ jääb neid üksnes 29.