TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Simmo Saan

Abstraktsete domeenide omaduspõhine testimine

Bakalaureusetöö (9 EAP)

Juhendaja: Vesal Vojdani, PhD
Juhendaja: Kalmer Apinis, PhD

Tartu 2018

Abstraktsete domeenide omaduspõhine testimine

Lühikokkuvõte:

One or two sentences providing a basic introduction to the field, comprehensible to a scientist in any discipline.

Two to three sentences of more detailed background, comprehensible to scientists in related disciplines.

One sentence clearly stating the general problem being addressed by this particular study.

One sentence summarising the main result (with the words “here we show´´ or their equivalent).

Two or three sentences explaining what the main result reveals in direct comparison to what was thought to be the case previously, or how the main result adds to previous knowledge.

One or two sentences to put the results into a more general context.

Two or three sentences to provide a broader perspective, readily comprehensible to a scientist in any discipline, may be included in the first paragraph if the editor considers that the accessibility of the paper is significantly enhanced by their inclusion.

Võtmesõnad:

List of keywords

CERCS:

CERCS kood ja nimetus: `https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e`

Property-based Testing of Abstract Domains

Abstract:

Inglisekeelne lühikokkuvõte

Keywords:

List of keywords

CERCS:

CERCS code and name: https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e

Sisukord

Unsolved issues

Sissejuhatus

What is it in simple terms (title)?

Why should anyone care?

What was my contribution?

What you are doing in each section (a sentence or two per section)

Tip: if it's hard for you to start writing, then try to split it to smaller parts, e.g. if the title is "Type Inference for a Cryptographic Protocol Prover Tool" then the "What is it" can be divided into "what is type inference", "what is cryptographic protocol" and "what is the prover tool". These three can also be split to smaller parts etc.

Valdkonna kirjeldus

Alamsektsioon sissejuhatusega kokku

Veel rohkem taustast, alustada kaugemalt: *static vs dynamic analysis*

Staatiline programmianalüüs on võimalikult automaatne protsess, mis programmi lähtekoodi põhjal järeldab midagi selle programmi käitumise kohta. Staatilist analüüsi teostatakse mitmel põhjusel. Esiteks, programmi optimeerimise eesmärgil teostatakse analüüsi kompilaatorites, leidmaks kohti programmikoodis, mida on automaatse muudatusega võimalik optimeerida, ilma et sellest muutuks programmi käitumine. Teiseks, programmist vigade leidmise eesmärgil, leidmaks vigu, ilma et oleks tarvis programm käivitada ja vigane olukord esile kutsuda. Kolmandaks, programmi korrektsuse näitamiseks, veendumaks, et programm kindlasti käitub oodatud veatul moel.

Mainida alternatiivseid staatilise analüüsi meetodeid

Nimetada ja viidata olemasolevatele analüsaatoritele

Andmevooanalüüs (*data-flow analysis*) on üks intuitiivne meetod staatilise programmianalüüsi teostamiseks. Selle keskseks ideeks on võimalike programmi seisundite, sh sageli muutujate võimalike seisundite, määramine selle programmi punktides. Andmevooanalüüs kasutab programmi juhtimisvoograafi (*control-flow graph*), et järgida programmi seisundi muutumist selle võimalike töövoogude jooksul.

Üldiselt pole võimalik staatilise analüüsiga alati täpselt määrata programmi seisundit, sest see oleks samaväärne programmi käivitamisega. Seetõttu vaadeldakse ligikaudseid seisundeid, mis vastavad konkreetsetele seisunditele. Sellist ligikaudsete seisundite uurimist nimetatakse abstraktseks interpretatsiooniks ja see põhineb rangel teoreetilisel alusel, millel on ligikaudsusele vaatamata head omadused. Nimelt, abstraktse interpretatsiooni

teooria lubab, et analüüs on korrektne, st kui uuritavast programmist otsitavat tüüpi viga ei leita, siis võib olla kindel, et seda seal päriselt ka ei ole.

Abstraktse interpretatsiooni korrektsuseks on vajalik, et vaadeldavad ligikaudsed programmi seisundid, mis moodustavadki abstraktse domeeni, rahuldaks teatud algebralisi omadusi, mis võimaldavad andmevooanalüüsi teostada sobiva võrrandisüsteemi lahendamise teel. Seetõttu on hädavajalik, et staatilist analüüsi teostav programm, analüsaator, ise oleks implementeeritud korrektselt, sest vastasel juhul pole teostatavate analüüside tulemused usaldusväärsed ja korrektsed.

Omaduspõhine testimine on testimismeetod, mis on sobib hästi programmi loogika matemaatiliste omaduse kontrollimiseks. Selleks kirjeldatakse kontrollitavad omadused predikaatidena ja neile juhuslike argumentide genereerimise metoodika. Nende kombineerimisel genereeritakse soovitud kogus juhuslike argumentide komplekte, millel leitakse predikaatide väärtused, kinnitades omaduse kehtimist või kummutades selle. Lisaks toetab omaduspõhise testimise raamistik leitud vääravate testjuhtude lihtsustamist.

Siduda analüsaatori korrektsus omaduspõhise testimise kasutamisega

Konkreetselt Goblint-ist

1 Abstraktsed domeenid

Andmevooanalüüsiga püütakse võimalikult täpselt määrata programmi seisundit igas programmi punktis.

Definitsioon 1.1. Domeeniks nimetatakse programmi kõikvõimalike seisundite hulka [1].

Selline informaalne definitsioon on ebapiisav mingisuguse teooria arendamiseks, mistõttu tegelikult vaadetakse domeene, mis moodustavad täieliku võre.

1.1 Võred

Kuigi võreteooria on arvestatav matemaatika haru, siis sügavamale laskumata on siin toodud põhilised mõisted võrede mõistmiseks. Järgnevad eestikeelsed definitsioonid on refereeritud V. Laane loengukonspektist [2], kuid tähistused on kohandatud programmianalüüsi kirjandusele omaseks [3:17]:

Definitsioon 1.2. Osaliselt järjestatud hulk on paar $(A, \sqsubseteq)$, kus A on hulk, millel on defineeritud binaarne seos $\sqsubseteq$, mis iga $a, b, c \in A$ korral rahuldab järgnevaid tingimusi:

- $a \sqsubseteq a$ (refleksiivsus),
- kui $a \sqsubseteq b$ ja $b \sqsubseteq c$, siis $a \sqsubseteq c$ (transitiivsus),
- kui $a \sqsubseteq b$ ja $b \sqsubseteq a$, siis $a = b$ (antisümmeetrilisus).

Olgu $(A, \sqsubseteq)$ osaliselt järjestatud hulk ja $X \subseteq A$.

Definitsioon 1.3. Elementi c nimetatakse hulga X **ülemiseks tõkkeks**, kui iga $x \in X$ korral $x \sqsubseteq c$. Vähimat ülemist tõket nimetatakse **ülemiseks rajaks**, st X-i iga ülemise tõkke d korral $c \sqsubseteq d$.

Definitsioon 1.4. Elementi c nimetatakse hulga X **alumiseks tõkkeks**, kui iga $x \in X$ korral $c \sqsubseteq x$. Suurimat alumist tõket nimetatakse **alumiseks rajaks**, st X-i iga alumise tõkke d korral $d \sqsubseteq c$.

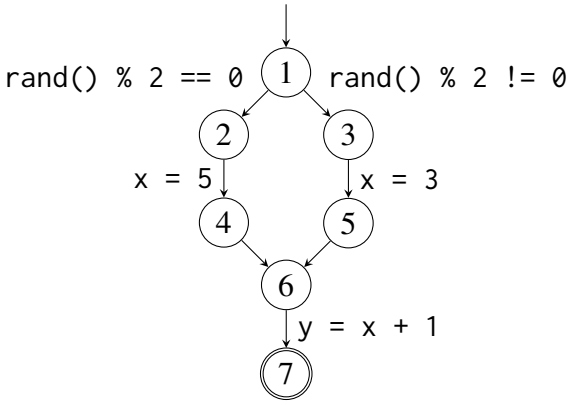
Hulga X ülemist ja alumist raja tähistame vastavalt $\bigsqcup X$ ja $\bigsqcap X$. Kui $X = \{a, b\}$, siis tähistame ülemise ja alumise raja vastavalt $a \sqcup b$ ja $a \sqcap b$.

Definitsioon 1.5. Täielik võre on osaliselt järjestatud hulk, mille igal alamhulgal leidub ülemine ja alumine raja.

Täielikus võres $(A, \sqsubseteq)$ leidub vähim element $\bot = \bigsqcap A$ ja suurim element $\top = \bigsqcup A$, mida nimetatakse vastavalt *bottom*iks ja *top*iks.

```c
int x;
if (rand() % 2 == 0)
    x = 5;
else
    x = 3;
int y = x + 1;
```

(a) C-keelne lähtekood



programmi laused tippudes või servades?

(b) Juhtimisvoograaf

Joonis 1. Tsüklita programmi näidis.

1.1.1 Motivatsioon

Kui nõuda, et domeen oleks täielik võre, siis pole ilmselge, mis rolli täidab sellel vaadeldav järjestus ja rajad.

Seisundite hulga vaatlemiseks võrena on vaja defineerida seisundite osaline järjestus. Kokkuleppeliselt järjestatakse seisundid täpsemast ebatäpsema suunas — kirjutis $a \sqsubseteq b$ tähendab, et seisund a on vähemalt sama täpne kui seisund b ning b on vähemalt sama üldine kui a. Seega $\top$ tähistab kõige ebatäpsemat seisundit.

Sellise järjestuse korral seisundite ülemine raja kujutab endast nende seisundite ühendamist.

*join*i intuitsioon

1.2 Potentshulga domeen

powerset = potentshulk?

Iga hulga S korral saab vaadelda selle kõigi alamhulkade hulka $\mathcal{P}(S)$, millel on loomulik osaline järjestus $\subseteq$. Osutub, et see on ka täielik võre, kus

- $a \sqsubseteq b \Leftrightarrow a \subseteq b$,
- $a \sqcup b = a \cup b$ ja $a \sqcap b = a \cap b$,
- $\bot = \varnothing$ ja $\top = S$.

Hulk X kirjeldab täisarvu z parajasti siis, kui $z \in X$. Sellega on defineeritud, kuidas peaks seostama domeeni elemente konkreetsete võimalike väärtustega.

Olgu vaatluse all programm jooniselt 1, kus on kõrvuti C-keelse lähtekoodi jupp ja selle juhtimisvoograaf, mille tippudes on programmi punktid, kus seisundeid vaadeldakse, ja servadel vastavad laused, mida täidetakse. Muutuja x võimalikke väärtusi ehk selle seisundeid saab analüüsida domeenis $(\mathcal{P}(\mathbb{Z}), \subseteq)$:

- Punktides 1, 2 ja 3 on muutuja veel väärtustamata. Kuna C keele semantika sellisel juhul mingit vaikeväärtust ei anna, siis võimalikke väärtusi kirjeldab kõige ebatäpsem domeeni element $\top = \mathbb{Z}$.
- Punktis 4 on muutujale just antud konstantne väärtus, mistõttu seda kirjeldab kõige paremini element $\{5\}$.

  Iseenesest poleks vale seostada selle programmi punktiga mõnda (osalise järjestuse järgi) üldisemat seisundit, nt $\{5, 6, 7\}$ või lausa $\top$, kuid see poleks nii kasulik, sest analüüsi mõte on siiski leida võimalikult täpne kirjeldus. Just selle täpsuse matemaatiliseks kirjeldamiseks nõutaksegi osalist järjestust.
- Punktiga 5 sobib samal põhjusel seostada element $\{3\}$.
- Punktis 6 on olukord huvitavam, sest seda seisundit pole programmis oleva hargnemise (täpsemalt selle ühendumise) tõttu kirjeldada ühe täisarvuga, vaid elemendiga $\{3, 5\}$. Selle tulemuseni jõudmiseks peab intuitiivselt ühendama eelneva kahe punkti seisundid — leidma seisundi, mis hõlmaks eelnevaid, olles seejuures võimalikult täpne. Just selleks nõutaksegi ülemise raja leidmise tehet, millega seda teha. Antud juhul $\{3\} \sqcup \{5\} = \{3, 5\}$.

Näitest peaks olema selge, miks üldse nõuda, et domeen oleks võre, ja millised võimalused see annab domeeni kasutamiseks programmide analüüsimisel. Kirjeldatud näide analüüsis ainult muutujat x ja jättis täielikult kõrvale muutuja y. Loomulikult võiks analüüs midagi öelda ka selle kohta.

Näide tervest analüüsist: +üleminekufunktsioonid +võrratuste süsteem +süsteemi lahendamine. Sama näite peal?

1.3 Kujutusdomeen

mapdomain — funktsioondomeen? kujutusdomeen?

Olgu Var programmi muutujate hulk ja Val domeen, milles vaadeldakse ühe muutuja seisundit. Sel juhul saab vaadelda abstraktsete muutujate väärtustuste domeeni [3:45]

$$\mathbb{D} = (\mathsf{Var} \to \mathsf{Val})_\perp = (\mathsf{Var} \to \mathsf{Val}) \cup \{\perp\}.$$

Domeeni elementideks on kujutused muutujate hulgast abstraktsete väärtuste hulka, mis on väga analoogilised konkreetsete muutujate väärtustustega funktsionaalselt kirjeldatuna. Element $\bot$ on tehislikult lisatud, et domeen moodustaks täieliku võre, ja tähendab hetkel teadaolevalt saavutamatut programmi punkti. Osaline järjestus selles domeenis on defineeritud järgnevaga:

$$D_1 \sqsubseteq D_2 \quad \iff \quad D_1 = \bot \quad \vee \quad \forall x \in \mathsf{Var}\ D_1(x) \sqsubseteq D_2(x).$$

Tehted kujutustel, $\top$?

Eelnevas näites $\mathsf{Var} = \{\mathsf{x}, \mathsf{y}\}$ ja jätkamiseks sobib juba nähtud $\mathsf{Val} = \mathcal{P}(\mathbb{Z})$. Analüüsides programmi jooniselt 1 nüüd selles domeenis, saame seisundid:

- Punktides 1, 2 ja 3: $\{\mathsf{x} \mapsto \top, \mathsf{y} \mapsto \top\}$.
- Punktis 4: $\{\mathsf{x} \mapsto \{5\}, \mathsf{y} \mapsto \top\}$.
- Punktis 5: $\{\mathsf{x} \mapsto \{3\}, \mathsf{y} \mapsto \top\}$.
- Punktis 6: $\{\mathsf{x} \mapsto \{3, 5\}, \mathsf{y} \mapsto \top\}$.
- Punktis 7: $\{\mathsf{x} \mapsto \{3, 5\}, \mathsf{y} \mapsto \{4, 6\}\}$, sest muutuja y on avaldatud x-i kaudu, mis on juba analüüsitud.

Abstraktne liitmistehe

Siit selgubki, et väikese vaevaga on võimalik lihtsam ühte muutujat korraga kirjeldav domeen laiendada kõigile muutujatele korraga. Seejuures muutub võimalikuks järgida andmete liikumist muutujate vahel, mis vastab juba paremini andmevooanalüüsi nimele.

Sellegipoolest pole vaadeldu praktiliselt teostatav, sest võimalike väärtuste hulk võib olla lõpmatu ja seejuures ka ebatriviaalne. Üldiselt pole võimalik selliseid hulki programmis kirjeldada ja nendega opereerida. Seetõttu loobutakse ülimast täpsusest suvaliste hulkade kujul ja kasutatakse ebatäpsemaid domeene väärtuste kirjeldamiseks, mida on võimalik analüsaatorisse implementeerida. Üheks variandiks on vaadelda väärtuste lõike.

1.4 Intervalldomeen

Intervalldomeen on domeen, milles täisarvude väärtuste abstraheerimiseks kasutatakse arvtelje lõike.

Definitsioon 1.6. Intervalldomeeniks [3:55] nimetatakse hulka

$$\mathbb{I} = \{[l, u] \mid l \in \mathbb{Z} \cup \{-\infty\}, u \in \mathbb{Z} \cup \{+\infty\}, l \leq u\},$$

millel on osaline järjestus

$$[l_1, u_1] \sqsubseteq [l_2, u_2] \quad \iff \quad l_2 \leq l_1 \wedge u_1 \leq u_2.$$

Sellises domeenis

$$[l_1, u_1] \sqcup [l_2, u_2] = [\min\{l_1, l_2\}, \max\{u_1, u_2\}].$$

$\mathbb{I}$ pole täielik võre! vaja eraldi võre definitsiooni?

Lõigu otspunktides on lubatud vastavad lõpmatust kirjeldavad väärtused, mis võimaldavad rääkida selles domeenis suurimast elemendist $\top = [-\infty, +\infty]$.

Lõik $[l, u]$ kirjeldab täisarvu z parajasti siis, kui $l \leq z \leq u$.

Valides eelnevas joonisel 1 tehtud näites $\mathsf{Var} = \mathbb{I}$, saame seisundid analoogiliselt:

- Punktides 1, 2 ja 3: $\{\mathsf{x} \mapsto \top, \mathsf{y} \mapsto \top\}$.
- Punktis 4: $\{\mathsf{x} \mapsto [5, 5], \mathsf{y} \mapsto \top\}$.
- Punktis 5: $\{\mathsf{x} \mapsto [3, 3], \mathsf{y} \mapsto \top\}$.
- Punktis 6: $\{\mathsf{x} \mapsto [3, 5], \mathsf{y} \mapsto \top\}$, kus x-i seisundi saab avaldada harudes olevatest: $[3, 3] \sqcup [5, 5] = [3, 5]$.
- Punktis 7: $\{\mathsf{x} \mapsto [3, 5], \mathsf{y} \mapsto [4, 6]\}$.

Kokkuvõte

what did you do?

What are the results?

future work?

Viidatud kirjandus

[1] Vojdani V. Mitmelõimeliste C-programmide kraasimine analüsaatoriga Goblin. Magistritöö. TÜ arvutiteaduse instituut, 2006. http://dspace.ut.ee/handle/10062/1253 (27. 02. 2018).

[2] Laan V. Võreteooria. Loengukonspekt. TÜ. Sügis 2017. https://courses.ms.ut.ee/MTMM.00.039/2017_fall/uploads/Main/kon.pdf (27. 02. 2018).

[3] Seidl H., Wilhelm R., and Hack S. Foundations and Intraprocedural Optimization. *Compiler Design: Analysis and Transformation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–114. https://doi.org/10.1007/978-3-642-17548-0_1 (02/27/2018).

[4] Vojdani V. et al. Static race detection for device drivers: the Goblint approach. *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016*. ACM, 2016, pp. 391–402.

[5] Might M. Order theory for computer scientists. http://matt.might.net/articles/partial-orders/ (03/03/2018).

Eemalda nocite

Lisad

I Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Simmo Saan**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

   Abstraktsete domeenide omaduspõhine testimine

   mille juhendajad on Vesal Vojdani ja Kalmer Apinis

   1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, pp.kk.aaaa

Kuupäev