Boole'i kehtestatavusel põhinev programmianalüüs

Kalmer Apinis
kalmer.apinis@ut.ee

Programmeerimiskeelte semantika uurimisseminar
Arvutiteaduse Instituut, Tartu Ülikool
detsember 2008.

Kokkuvõte

Näitame Dillig, Dillig ja Aiken'i uut meetodi programmide staatilise analüüsi jaoks, mis on täielikult konteksti- ja rajatundlik. Kirjeldatud tehnika kasutab rekursiivseid, kvantifitseeritud valemeid, mille abil saame paljusid programmi omadusi täpselt modelleerida. Valemites eristatakse vaadeldavas skoobis jälgitavaid ja mittejälgitavaid muutujaid, milledest mittejälgitavad seotakse eksistentsikvantoriga. Saadud valemitest arvutatakse suletud vormid, kasutades omadust, et mittejälgitavad muutujad saab teatavatel tingimustel elimineerida.

Näitame, et antud lahendus on täpne vastamaks *võib* ja *peab* küsimustele ning suur osa selliste küsimuste lahendusi on praktikas väikesed. Lisaks näitame, et antud meetodi kasutav analüsaator skaleerub kogu Linuxi kernelile, kus on rohkem kui 6 miljonit rida koodi.

1 Sissejuhatus

Staatilise analüüsi juures soovitakse tõestada programmi kohta mingeid omadusi. Sellise omaduse näiteks on *null*-viidast lugemine. Niisuguseid omaduste analüüsijaid võime jaotada kahte rühma – korrektsuse tõestajad ja vea leidmise tööriistad.

Võime tahta tõestada, et mingi programmi käivitamisel ei jõuta kunagi olukorrani, kus üritatakse andmeid lugeda *null*-viidast. Selliste soovide puhul on vaja olla korrektne st. kui sisendprogrammis ühtegi viga ei leitud siis ka reaalselt viga ei teki.

Samas võib motivatsioon olla hoopis erinev, näiteks võidakse tahta üles leida võimalikult suur hulk vigu. Ehk siis on vaja leida üles programmiread, kus mingitel juhtudel üritatakse *null*-viidast andmeid lugeda. Sellisel juhul pole tähtis, et tegemist oleks tegemist korrektse analüüsiga, oluline on hoopis see, et üles leida võimalikult suur hulk vigu, millest valede veateadete arv oleks minimaalne.

Rajatundlik analüüs omab infot, millist rada mööda on programmi mingi seisundini jõutud, ja see on paljude programmianalüsaatorite tähtis osa. Eelnevalt ei ole leidunud skaleeruvat,

korrektset ja täielikku algoritmi rekursiooniga keelte rajatundliku analüüsi jaoks. Isegi mittetäielikud implementatsioonid on olnud liiga aeglased protseduuridevaheliste rajatundlike olukordade arvutamiseks. Olemasolevad lähenemisviisid kasutavad heuristikaid, aktsepteerivad osalist skaleeruvust või võimalikku mittedeterminismi.

Isil Dillig, Thomas Dillig ja Alex Aiken pakuvad välja täielikult raja- ja kontekstitundliku programmianalüsaatori[4]. Nende algoritm on korrektne ja täielik suletud programmis (ehk kogu programmi analüüsides), lõplike domeenide korral. Tehnika põhiidee seisneb selles, et muutujate võrdlust literaalidega modelleeritakse analüüsis ülitäpselt ning Boole'i kujule viies tehakse lihtsustusi. Nullviitade analüüsi lahendamiseks skaleerub meetod piisavalt hästi, et analüüsida Linuxi kernelit, kus on üle 6 miljoni rea koodi.

Käesolevas töös tuuakse esmalt sisse lihtne funktsionaalne keel, mida analüüsida ja kirjeldatakse, kuidas eelnevalt antud funktsionaalsest keele suvalist programmi teisendada võrrandite süsteemiks. Edasi näidatakse meetodit, millega teisendada programmi võrrandite süsteem Boole'i võrrandite süsteemiks, kirjeldatakse vajalikke tingimusi Boole'i võrrandisüsteemile ja antakse juhend süsteemi tõlkimiseks tingimusi rahuldavale kujule. Järgnevalt antakse algoritm saadud rekursiivsete Boole'i võrrandite lahendamiseks. Seejärel vaadatakse Dillig et al. tehnikale sarnaseid algoritme, ning erinevusi nende vahel. Kahes viimases peatükis räägitakse algoritmi implementeerimisest analüsaatorisse Saturn ja eksperimentaalseid tulemusi Saturni rakendamisel suurtele vabavaralistele C programmidele.

2 Kitsendused

Tehnika näitamiseks valime sobiva funktsionaalse keele:

$$
\begin{aligned}
\text{programm } P &::= F^+ \\
\text{funktsioon } F &::= \text{define } f(x) = e \\
\text{väljend } E &::= \mathsf{true} \mid \mathsf{false} \mid c_i \mid x \mid f(e) \\
&\quad \mid \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \\
&\quad \mid \text{let } x = e_1 \text{ in } e_2 \\
&\quad \mid e_1 = e_2 \mid e_1 \wedge e_2 \mid e_1 \vee e_2 \mid \neg e
\end{aligned}
$$

Väljendid (ik *expressions*) võivad olla tõesed (true), väärad (false), abstraktsed väärtused (c_i), funktsiooni argumendid, funktsiooni väljakutsed, tingmusavaldised, *let*-avaldised või võrdlusavaldised. Tõeväärtuste koostamisel kasutame standardseid operaatoreid ($\wedge, \vee, \neg$). Mittejälgitavate omadusi modelleeritakse vabade muujatega. Vabu muutujaid vaatamegi, kui mittedeterministliku väärtust, mis on valitud funktsiooni väljakutsel.

Selline keel väljendab praktikas uuritava keele (näiteks C) alamosa, mida antud tehnika suudab väga täpselt uurida. Tähtis on, et väärtuste hulk oleks lõplik. Samuti eeldame, et funktsioonide tagastusväärtus ja võrdluse avaldise $e_1 = e_2$ juures mõlemad alamavaldised väärtustuvad abstraktseteks väärtusteks. Lisaks omab selline keel kaht kasutatud tehnika näitamiseks vajalikku omadust. Esiteks, saame kasutada predikaate tingimusavaldistes, mis teeb rajakindluse

mittetriviaalseks. Teiseks, funktsioonid võivad tagastada suvalise väärtuse c_i.

Abstraktsed väärtused c_i tähistavad analüüsi tegijaid huvitavaid muutujale omitatavaid väärtusi. Näiteks, kui tõlgime C keelset programmi, võime valida abstraktseteks väärtusteks konkteetsete väärtuste hulgad: $c_1 = \{42\}$ ja $c_2 = \{\dots, 40, 41, 43, 44, \dots\}$. Selliselt saame lõpliku domeeni ning konkreetsetel konstantidel täpse analüüsi.

Igale funktsioonile omistatakse sellele vastavad kitsendused kujul:

$$\begin{aligned} \text{võrrand } \mathcal{E} &::= [\overrightarrow{\Pi}_i] = \exists \beta_1, \cdots, \beta_m.[\overrightarrow{\mathcal{F}_i}] \\ \text{kitsendus } \mathcal{F} &::= (\tau_1 = \tau_2) \mid \Pi[C_i/\alpha] \\ &\quad \mid \mathcal{F}_1 \wedge \mathcal{F}_2 \mid \mathcal{F}_1 \vee \mathcal{F}_2 \mid \neg\mathcal{F} \\ \text{tüüp } \tau &::= \alpha \mid C_i \end{aligned}$$

Kitsendused (ik *constraints*) on tüüpide (e. tüübimuutujate ja abstraktsete väärtuste) võrrandid, asendusega kitsenduste muutujad või lausearvutuse tehted kitsenduste vahel. Kitsendus $\Pi_{f,\alpha,c}$ väljendab tingimust, millal funktsioon f sisendil α tagastab mingi konkreetse väärtuse c. Indeks jäetakse tihti kirjutamata, kui see on kontekstist arusaadav.

Näide 1. Kui meil on kaks abstraktset väärtust c_1 ja c_2 siis võrrand

$$[\Pi_{f,\alpha,C_1}, \Pi_{f,\alpha,C_2}] = [\mathsf{true}, \mathsf{false}]$$

näitab, et funktsioon f tagastab alati väärtuse c_1. Samuti näide

$$[\Pi_{f,\alpha,C_1}, \Pi_{f,\alpha,C_2}] = [\alpha = C_2, \alpha = C_1]$$

kirjeldab seda, et funktsioon f tagastab c_1, kui sisendiks on c_2, ja c_2, kui sisendiks on c_1.

Näide 2. Funktsioon

$$\text{define } f(x) = \text{if}(y = c_2) \text{ then } c_1 \text{ else } c_2,$$

kus y on vaba muutuja, modelleeritakse võrrandiga

$$[\Pi_{f,\alpha,C_1}, \Pi_{f,\alpha,C_2}] = \exists \beta.[\beta = C_2, \beta = C_1].$$

Näeme, et β modelleerib tundmatut muutujat y ning β on jagatud mõlema kitsenduse vahel. Igal lahendusel peab β olema kas C_1 või C_2 seega on peab funktsioon tagastama ühe väärtuse.

Joonisel 1 on antud keele kitsenduste tuletusreeglid. Reeglid 1-5 kirjeldavad otsuseid kujul $A \vdash_{\mathsf{true,false}} e : \mathcal{F}$ st. kitsendusi $\mathcal{F}$, kus avaldis e väärtustub mingiks tõeväärtuseks. Reeglid 6-11 kirjeldavad otsuseid $A \vdash_{c_i} e : \mathcal{F}$, kus avaldis e väärtustub abstraktseks väärtuseks c_i. Reegel 12 konstrueerib võrrandite süsteemi, andes tingimused, kuna funktsioon tagastab vastava abstraktse väärtuse.

Reegel 1 ütleb, et avaldis true väärtustub alati (tingimusel true) väärtuseks true ja reegel 2, et avaldis false ei väärtustu kunagi (tingimusel false) väärtuseks true. Reegel 3 ütleb, et avaldiste e_1 ja e_2 võrdsuskontroll tagastab väärtuse true kõikidel juhtudel, kus mõlemad avaldised väärtustuvad võrdseteks abstraktseteks väärtusteks. Reeglid 4 ja 5 katavad ära juhtumid, kus avaliseks on tõeväärtuse operatsioonid.

Joonis 1: Kitsenduste tuletusreeglid

$$\frac{}{A \vdash_{true} true : true} \tag{1}$$

$$\frac{}{A \vdash_{true} false : false} \tag{2}$$

$$\frac{A \vdash_{c_i} e_1 : \mathcal{F}_{1,i} \qquad A \vdash_{c_i} e_2 : \mathcal{F}_{2,i}}{A \vdash_{true} e_1 = e_2 : \bigvee_i (\mathcal{F}_{1,i} \wedge \mathcal{F}_{2,i})} \tag{3}$$

$$\frac{A \vdash_{true} e : \mathcal{F}}{A \vdash_{true} e : \neg\mathcal{F}} \tag{4}$$

$$\frac{A \vdash_{c_i} e_1 : \mathcal{F}_{1,i} \qquad A \vdash_{c_i} e_2 : \mathcal{F}_{2,i}}{A \vdash_{true} e_1 \otimes e_2 : \mathcal{F}_{1,i} \otimes \mathcal{F}_{2,i}} \qquad (\otimes \in \{\wedge, \vee\}) \tag{5}$$

$$\frac{}{A \vdash_{c_i} c_i : true} \tag{6}$$

$$\frac{i \neq j}{A \vdash_{c_i} c_j : false} \tag{7}$$

$$\frac{A(x) = \alpha}{A \vdash_{c_i} x : (\alpha = C_i)} \tag{8}$$

$$\frac{A \vdash_{true} e_1 : \mathcal{F}_1 \qquad A \vdash_{c_i} e_2 : \mathcal{F}_2 \qquad A \vdash_{c_i} e_3 : \mathcal{F}_3}{A \vdash_{c_i} \text{ if } e_1 \text{ then } e_2 \text{ else } e_3 : (\mathcal{F}_1 \wedge \mathcal{F}_2) \vee (\neg\mathcal{F}_1 \wedge \mathcal{F}_3)} \tag{9}$$

$$\frac{A \vdash_{c_j} e_1 : \mathcal{F}_{1j} \qquad A, x : \alpha \vdash_{c_i} e_2 : \mathcal{F}_{2i} \quad (\alpha \text{ fresh})}{A \vdash_{c_i} \text{ let } x = e_1 \text{ in } e_2 : \bigvee_j (\mathcal{F}_{1j} \wedge \mathcal{F}_{2i} \wedge (\alpha = C_j))} \tag{10}$$

$$\frac{A \vdash_{c_k} e : \mathcal{F}_k}{A \vdash_{c_i} f(e) : \bigvee_k (\mathcal{F}_k \wedge \Pi_{f,\alpha,c_i}[C_k/\alpha])} \tag{11}$$

$$\frac{x : \alpha \notin \{\beta_1, \cdots, \beta_j\}, y_1 : \beta_1, \cdots, y_m : \beta_m \vdash_{c_i} e : \mathcal{F}_i \quad 1 \leq i \leq n}{\vdash \text{define} f(x) = e : [\overrightarrow{\Pi}_{f,\alpha,c_i}] = \exists \beta_1, \cdots, \beta_m . [\overrightarrow{\mathcal{F}}_i]} \tag{12}$$

Näide 3. Analüüsime funktsiooni

$$\text{define } f(x) = \text{if } ((x = c_1) \vee (y = c_2)) \text{ then } c_1 \text{ else } f(c_1)$$

kus y on defineerimata ja ainukesed abstraktsed väärtused on c_1 ja c_2. Vaatame juhtu, kui f väljastab väärtuse c_1.

$$\dfrac{\dfrac{\dfrac{A(x) = \alpha}{A \vdash_{\mathsf{true}} x = c_1 : \alpha = C_1} \quad \dfrac{A(y) = \beta}{A \vdash_{\mathsf{true}} y = c_2 : \beta = C_2}}{A \vdash_{\mathsf{true}} x = c_1 \vee y = c_2 : \alpha = C_1 \vee \beta = C_2} \quad \dfrac{}{A \vdash_{c_1} c_1 : \mathsf{true}} \quad \dfrac{A \vdash_{c_1} c_1 : \mathsf{true} \qquad A \vdash_{c_2} c_1 : \mathsf{false}}{A \vdash_{c_1} f(c_1) : \Pi_{f,\alpha,C_1}[c_1\alpha]}}{x : \alpha, y : \beta = A \vdash_{c_1} \text{if } (x = c_1 \vee y = c_2) \text{ then } c_1 \text{ else } f(c_1) : (\alpha = C_1 \vee \beta = C_2) \wedge \neg(\alpha = C_1 \vee \beta = C_2) \wedge \Pi_{f,\alpha,C_1}[C_1\alpha]}$$

Siis saame kitsenduseks

$$\begin{bmatrix} \Pi_{f,\alpha,C_1} \\ \dots \end{bmatrix} = \exists\beta \begin{bmatrix} (\alpha = C_1 \vee \beta = C_2) \vee \neg(\alpha = C_1 \vee C_2) \wedge \Pi_{f,\alpha,C_1}[C_1/\alpha] \\ \dots \end{bmatrix}$$

kusjuures $[C_1/\alpha]$ näitab, et rekursiivse väljakutse argumendiks on c_1.

Kitsendusi interpreteerime nelja väärtusega võres, kus $\bot \leq \mathsf{true}, \mathsf{false}, \top$ ja $\bot, \mathsf{true}, \mathsf{false} \leq \top$ ning $\wedge$ on *meet* ja $\vee$ in on *join*. Samuti $\neg\bot = \bot$, $\neg\top = \top$, $\neg\mathsf{true} = \mathsf{false}$ ja $\neg\mathsf{false} = \mathsf{true}$. Olgu meil eksistentsiaalsete muutujate väärtustuse θ kitsenduste süsteemile E. Sellisel juhul on E semantikaks E võrrandite lahtirullimise piirväärtus. Meid huvitab nii vähim püsipunkt (kus algseisus on Π muutujad $\bot$) ja suurim püsipunkt (kus alustatakse muutujatega $\top$).

Väärtus $\bot$ vähima püsipunkti (ja $\top$ suurima püsipunkti) semantikas on mitte-termineeruv programm. Kuna eesmärgiks pole termineerumise üle arutlemine, eeldame et programmid termineeruvad. Antud tuletusreeglid garanteerivad, et $\Pi_{f,\alpha,C_1} \wedge \Pi_{f,\alpha,C_j} \leq \mathsf{false}$ vähima püsipunkti semantikas kui $i \neq j$. See tähendab, et funktsiooni erinevate väärtuste tagastamise tingimused on üksteist välistavad. Sarnaselt kehtib $\bigvee_i \Pi_{f,\alpha,C_i} \geq \mathsf{true}$ suurima püsipunkti semantikas.

Näide 4. Analüüsides C funktsiooni:

```
int f(int x){
  if ((x == 1) || (y == 2))
    return 1
  else
    return f(rnd());
}
```

saame funktsiooni f väärtuse 1 tagastamise tingimuseks

$$\begin{aligned} \Pi_{f,x,1} = &((x = 1 \vee y = 2) \wedge \mathsf{true}) \vee \\ &(\neg(x = 1 \vee y = 2) \wedge ((\Pi_{rnd,,1} \wedge \Pi_{f,x,1}[1/x]) \vee (\Pi_{rnd,,2} \wedge \Pi_{f,x,1}[2/x])) \vee \dots) \end{aligned}$$

3 Boole'i kitsendused

Iga muutuja σ_i kohta, tuuakse sisse Boole'i muutujad $\sigma_{i1}, \sigma_{i2}, \cdots, \sigma_{in}$ nii, et σ_{ij} on tõene parajasti siis, kui $\sigma_i = C_j$. Jälgitavale muutujatele viitame kui α_{ij} ja mittejälgitavaid muutujaid kui β_{ij}. Lisaks tõlgendatakse Π_{f,α,C_i}, kui Boole'i muutujat, mis tähistab tingimust, et funktsioon f tagastab C_i sisendil α.

Samuti tõlgime kõik $\tau_1 = \tau_2$ esinemised tüübikitsendustes:

$$C_i = C_i \Leftrightarrow \mathsf{true} \tag{13}$$

$$C_i = C_j \Leftrightarrow \mathsf{false} \qquad (i \neq j) \tag{14}$$

$$v_i = C_j \Leftrightarrow v_{ij} \tag{15}$$

Paneme tähele, et võrdlust kujul $v_i = v_j$ ei teki joonisel 1 antud reeglite järgi. Tõlgime kõik asendused $[C_j/\alpha_{ik}]$ Boole'i asendustega $[\mathsf{true}/\alpha_{ij}]$ ja $[\mathsf{false}/\alpha_{ik}]$ kus $k \neq k$. Nüüd moodustab algne tüübikitsendus rekursiivse süsteemi Boole'i kitsendustest:

$$E = \begin{bmatrix} [\overrightarrow{\Pi}_{f_1,\alpha,C_i}] = \exists\overrightarrow{\beta_1}.[\overrightarrow{\psi}_{1i}(\overrightarrow{\alpha}_1, \overrightarrow{\beta}_1, \overrightarrow{\Pi}[\overrightarrow{b}_1\overrightarrow{\alpha}])] \\ \cdots \\ [\overrightarrow{\Pi}_{f_k,\alpha,C_i}] = \exists\overrightarrow{\beta_k}.[\overrightarrow{\psi}_{ki}(\overrightarrow{\alpha}_k, \overrightarrow{\beta}_k, \overrightarrow{\Pi}[\overrightarrow{b}_k\overrightarrow{\alpha}])] \end{bmatrix} \tag{16}$$

kus $\overrightarrow{\Pi} = \langle \Pi_{f_1,\alpha,C_1}, \ldots, \Pi_{f_k,\alpha,C_n} \rangle$ ja $b_i \in \{\mathsf{true}, \mathsf{false}\}$ ning ψ on kvantifitseerimata valem üle $\overrightarrow{\beta}, \overrightarrow{\alpha}$ ja $\overrightarrow{\Pi}$. Asendused kujul $\overrightarrow{\Pi}[\overrightarrow{b}/\overrightarrow{\alpha}]$ tulenevad funktsiooniväljakutse tuletusreeglist (reegel 11).

Hetkel defineeritud Boole'i kitsendused ei säilita tüübikitsenduste lahendusi. Lisame reeglid, mis garanteerivad, et lahendatud kitsenduste kõikidele muutujatele omistatakse mingi abstraktne väärtus ja ühelegi muutujale ei omistata väärtus mitu korda. Iga Boole'i vektori v_i korral peavad kehtima järgmised tingimused:

1. Unikaalsus $\quad \psi_{\text{unik}} = \bigwedge_{j \neq k} \neg(v_{ij} \wedge v_{ik})$
2. Eksistents $\quad \psi_{\text{eks}} = \bigvee_j v_{ij}$

Definitsioon 3.1. $SAT^*(\phi) \equiv SAT(\phi \wedge \psi_{unik} \wedge \psi_{eks})$

Definitsioon 3.2. $VALID^*(\phi) \equiv (\{\psi_{unik}\} \cup \{\psi_{eks}\} \models \phi)$

Kehtestatavus peab jälgima unikaalsuse ja eksistentsi tingimusi igal temas esinevas Boole'i vektoril v_i, lisaks valemi enda kehtestatavusele. Samaselt tõesus võib aga kasutada unikaalsuse ja eksistentsi eeldust st. $\alpha_{11} \vee \alpha_{12}$ on samaselt tõene, kui keeles on ainult kaks väärtust.

Definitsioon 3.3. *Olgu ϕ kvantoriteta valem üle α_{ij}, β_{ij} ja Π_{ij}. Olgu $\mathcal{M}(\phi)$ valemi ϕ eituse normaalkuju st. eitused võimalikult sügaval, kahekordsete eitusteta ja $\neg v_{ij}$ asendatud $\bigvee_{k \neq j} v_{ik}$. Sellisel juhul ütleme, et $\mathcal{M}(\phi)$ on monotoone v_{ij}-s.*

Lemma 1. $SAT^*(\phi) \Leftrightarrow SAT(\mathcal{M}(\phi) \wedge \psi_{unik})$

Tõestus. Olgu $\overline{v}$ valemi $\mathcal{M}(\phi) \wedge \psi_{\text{unik}}$ väärtustus. Juhul, kui $\overline{v}$ väärtustab v_{ij} väärtuseks false nii, et ψ_{eks} oleks rikutud. Kuna $\mathcal{M}(\phi)$ ei sisalda eitust, saame ϕ-s väärtustada v_{ij} väärtuseks true, siis $\mathcal{M}(\phi), \psi_{\text{eks}}$ ja ψ_{unik} on rahuldatud. Järelikult ka $\text{SAT}^*(\phi)$ Teist pidi tõestus on samuti lihtne, ja seda pole välja toodud. □

Lemma 2. *VALID*$^*(\phi) \Leftrightarrow (\{\psi_{eks}\} \models \mathcal{M}(\phi))$

Tõestus. Duaalne eelmise lemma tõestusega. □

Näide 5. Kasutades näite 4 funktsiooni f tagastusväärtuste kitsenduseks

$$\exists v_y \exists \Pi_{rnd,,1} \exists \Pi_{rnd,,2} \dots \exists \Pi_{rnd,,max\_int}[\Pi_{f,x,1}, \dots, \Pi_{f,x,3}, \dots], \tag{17}$$

kus

$$\begin{aligned}\Pi_{f,x,1} = (v_{x1} \vee v_{y2}) \vee (\neg(v_{x1} \vee v_{y2}) \wedge ((&\Pi_{rnd,,1} \wedge \Pi_{f,x,1}[\mathsf{true}/v_{x1}]) \vee \\ &(\Pi_{rnd,,2} \wedge \Pi_{f,x,1}[\mathsf{false}/v_{x1}]) \vee \\ &(\Pi_{rnd,,3} \wedge \Pi_{f,x,1}[\mathsf{false}/v_{x1}]) \vee \dots)\end{aligned}$$

ja

$$\begin{aligned}\Pi_{f,x,3} = \neg(v_{x1} \vee v_{y2}) \wedge ((&\Pi_{rnd,,1} \wedge \Pi_{f,x,3}[\mathsf{true}/v_{x1}]) \vee \\ &(\Pi_{rnd,,2} \wedge \Pi_{f,x,3}[\mathsf{false}/v_{x1}]) \vee \\ &(\Pi_{rnd,,3} \wedge \Pi_{f,x,3}[\mathsf{false}/v_{x1}]) \vee \dots)\end{aligned}$$

4 Vajalike ja piisavate tingimuste arvutamine

Algsetel kitsendustel oli neli väärtustust, kuid Boole'i kitsendustel ainult kaks. Järgnevalt püüame tõestada, et kui algne kitsendus on väärtusega true või false, siis ka peale tõlkimist on väärtus sama. Lahenduse võtmeks on tuletada vajalikud/piisavad tingimused kitsenduste süsteemist C. See tähendab vajalikud (/piisavad) tingimused peavad olema kehtestatavad (/samaselt tõesed) samal ajal kui C on kehtestatav (/samaselt tõene).

Nõuetele on kaks tehnilist vajadust:

- Vajalikud (/piisavad) tingimused peavad olema nii tugevad (/nõrgad) kui võimalik.
- Vajalikud ja piisavad tingimused peaksid kasutama ainult jälgitavaid muutujaid.

Järgnevalt kasutame $\mathcal{V}^-(\phi)$ ja $\mathcal{V}^+(\phi)$, et tähistada vastavalt mittejälgitavaid ja jälgitavaid muutujaid β_{ij} ja α_{ij} valemis ϕ

Definitsioon 4.1. *Olgu ϕ kvantoriteta valem. Ütleme, et $\lceil\phi\rceil$ on tugevaim jälgitav vajalik tingimus ϕ jaoks, kui*

1. $\phi \Rightarrow \lceil\phi\rceil$
2. $\forall\phi'.((\phi \Rightarrow \phi') \Rightarrow (\lceil\phi\rceil \Rightarrow \phi))$, *kui* $\mathcal{V}^-(\phi') = \emptyset$ *ja* $\mathcal{V}^+(\phi') = \mathcal{V}^+(\phi)$

Esimene definitsiooni tingimus tähendab, et $\lceil\phi\rceil$ on vajalik ϕ jaoks. Samas on $\lceil\phi\rceil$ tugevam, kui kõik teised vajalikud tingimused, milles ainukesed muutujad on ϕ jälgitavad muutujad.

Definitsioon 4.2. *Olgu ϕ kvantoriteta valem. Ütleme, et $\lfloor\phi\rfloor$ on nõrgim jälgitav piisav tingimus ϕ jaoks, kui*

1. $\lfloor\phi\rfloor \Rightarrow \phi$

2. $\forall\phi'.((\phi' \Rightarrow \phi) \Rightarrow (\phi' \Rightarrow \lfloor\phi\rfloor))$, *kui* $\mathcal{V}^-(\phi') = \emptyset$ *ja* $\mathcal{V}^+(\phi') = \mathcal{V}^+(\phi)$

Esimene definitsiooni tingimus tähendab, et $\lfloor\phi\rfloor$ on piisav ϕ jaoks. Samas on $\lfloor\phi\rfloor$ nõrgim, kui kõik teised piisavad tingimused, milles ainukesed muutujad on ϕ jälgitavad muutujad.

Ilma tõestuseta võtame teadmiseks järgmise üldteatava lemma:

Lemma 3. *Iga valemi ψ jaoks, jälgitavad tugevaimad vajalikud ja nõrgimad piisavad tingimused leiduvad ja on unikaalsed loogilise võrduse täpsuseni.*

Tugevaim vajalik ja nõrgim piisav tingimus on programmianalüüsile omane, kuna see vastab päringutele, mis käivad programmi omaduste kohta. Olgu ϕ tingimus, millal programmi mingi konkreetne omadus P kehtib. Siis piisava ja vajaliku tingimuse definitsioonist tuleneb:

- kui SAT($\lceil\phi\rceil$) siis P võib kehtida,
- kui VALID($\lfloor\phi\rfloor$) siis P peab kehtima.

Ja lisaks vastava tugevaima ja nõrgima tingimuse kohta kehtivad tingimused

- kui UNSAT($\lceil\phi\rceil$), siis P ei saa kehtida ja
- kui INVALID($\lfloor\phi\rfloor$), siis P ei pruugi kehtida.

Kui ϕ-s on ainult jälgitavad muutujad, siis $\lceil\phi\rceil = \phi = \lfloor\phi\rfloor$, samas kui ϕ-s esinevad vaid mittejälgitavad muutujad, siis $\lceil\phi\rceil = \mathsf{true}$ ja $\lfloor\phi\rfloor = \mathsf{false}$. Niisiis on väärt tähele panna, et tugevaima vajaliku ja nõrgima piisava tingimuse vahe on sama mis ebakindlus algse valemi suhtes.

5 Kitsenduste arvutamine

Järgnevalt anname samm-sammult algoritmi, mis arvutab jälgitava tugevaima vajaliku ja nõrgima piisava tingimuse eelnevalt defineeritud valemite jaoks. Esmalt peame elimineerima eksistentsiaalselt kvantifitseeritud mittejälgitavad muutujad β_{ij}. Seda teeme, kasutades üldteadaolevat lemmat:

Lemma 4. *1. Tugevaim vajalik tingimus ϕ jaoks, kus ei esine v:*

$$SNC(\phi, v) \equiv \phi[\mathsf{true}/v] \vee \phi[\mathsf{false}/v] \tag{18}$$

2. Nõrgim piisav tingimus ϕ jaoks, kus ei esine v:

$$WSC(\phi, v) \equiv \phi[\mathsf{true}/v] \wedge \phi[\mathsf{false}/v] \tag{19}$$

Selle lemma tõestas esimesena Boole[2] ja selliselt tugevaima vajaliku ja nõrgima piisava tingimuse arvutamine on tuntud, kui vahepealse termi elimineerimine ning muutuja unustamine.

Tuletame meelde, et peame rahuldama ka unikaalsuse ja eksistentsi eeldust. Kui arvutame lemma 4 järgi tugevaimat vajalikku tingimust $\beta_{11} \wedge \beta_{12}$ jaoks saame vastuseks true ehk tõene, kuid unikaalsust arvestades on õigeks vastuseks väär. Sarnased probleemid tekivad ka nõrgima piisava tingimusega seega peame andma definitsioonid, mis meie eeldustega sobiks.

Definitsioon 5.1. *1. Tugevaim vajalik tingimus, mis rahuldab unikaalsus ja eksistentsi tingimusi:*

$$SNC^*(\phi, v) \equiv (\phi \wedge \psi_{eks} \wedge \psi_{unik})[\mathsf{true}/v] \vee (\phi \vee \psi_{eks} \vee \psi_{unik})[\mathsf{false}/v] \tag{20}$$

2. Nõrgim piisav tingimus, mis rahuldab unikaalsuste ja eksistentsi tingimusi:

$$WSC^*(\phi, v) \equiv (\phi \wedge \psi_{eks} \wedge \psi_{unik})[\mathsf{true}/v] \wedge (\phi \vee \psi_{eks} \vee \psi_{unik})[\mathsf{false}/v] \tag{21}$$

Nüüd asendame kõik avaldised $\exists v.\phi$ avaldistega $\mathrm{SNC}^*(\phi, v)$ tugevaima vajaliku valemi saamiseks ja $\mathrm{WSC}^*(\phi, v)$ nõrgima piisava valemi saamiseks. Saame kaks süsteemi E_{NC} ja E_{SC}.

$$E_{\mathrm{NC}} = \begin{bmatrix} \lceil \Pi_{f_1,\alpha,C_1} \rceil & = & \phi'_{11}(\overrightarrow{\alpha}_1, \lceil \overrightarrow{\Pi} \rceil [\overrightarrow{b}_1/\overrightarrow{\alpha}]) \\ & \cdots & \\ \lceil \Pi_{f_k,\alpha,C_n} \rceil & = & \phi'_{kn}(\overrightarrow{\alpha}_k, \lceil \overrightarrow{\Pi} \rceil [\overrightarrow{b}_k/\overrightarrow{\alpha}]) \end{bmatrix} \tag{22}$$

$$E_{\mathrm{SC}} = \begin{bmatrix} \lfloor \Pi_{f_1,\alpha,C_1} \rfloor & = & \phi'_{11}(\overrightarrow{\alpha}_1, \lfloor \overrightarrow{\Pi} \rfloor [\overrightarrow{b}_1/\overrightarrow{\alpha}]) \\ & \cdots & \\ \lfloor \Pi_{f_k,\alpha,C_n} \rfloor & = & \phi'_{kn}(\overrightarrow{\alpha}_k, \lfloor \overrightarrow{\Pi} \rfloor [\overrightarrow{b}_k/\overrightarrow{\alpha}]) \end{bmatrix} \tag{23}$$

Näide 6. Kasutades näite 4 funktsiooni f saame arvutada tugevaimat vajalikku tingimust, et väljastatakse 1.

$$\begin{aligned} \mathrm{SNC}(\Pi_{f,x,1}) = {} & (v_{x1} \vee \mathsf{true}) \vee (\neg(v_{x1} \vee \mathsf{true}) \wedge \dots) \vee \\ & (v_{x1} \vee \mathsf{false}) \vee (\neg(v_{x1} \vee \mathsf{false}) \wedge (\Pi_{f,x,1}[\mathsf{true}/v_{x1}] \vee \Pi_{f,x,1}[\mathsf{false}/v_{x1}])) \\ = {} & \mathsf{true} \end{aligned}$$

Näeme, et valemist lihtsustub välja *rnd()* erinevad tagastusväärtused ja jääb sisse ainult tegelikult kontrollitud tingimus muutujas v_{x1}. Kuna eksistentsi ega unikaalsuse vastu ei eksita $(\mathrm{SNC}(\Pi_{f,x,1}) = \mathrm{SNC} * (\Pi_{f,x,1}))$ seega järeldub, et f võib tagastada väärtuse 1.

6 Omaduste säilitamine asendustel

Näide 7. Olgu defineeritud järgmised funktsioonid

$$\begin{aligned} &\text{define } g(x) = \text{if } (x = c_1 \wedge z = c_2) \text{ then } c_1 \text{ else } c_2 \\ &\text{define } f(x) = \text{let } y = \ g(c_1) \text{ in} \\ &\qquad\qquad\qquad \text{if } (\neg(y = c_1)) \text{ then } c_1 \text{ else } c_2 \end{aligned}$$

Tahame arvutada tugevaimat piisavat tingimust, et f väljastaks c_1. Saame süsteemi

$$\lceil \Pi_{f,\alpha,C_1} \rceil = \neg(\lceil \Pi_{g,\alpha,C_1} \rceil [\mathsf{true}/\alpha_1][\mathsf{false}/\alpha_2])$$
$$\lceil \Pi_{g,\alpha,C_1} \rceil = \alpha_1$$

kus $\alpha_1 = \mathsf{true}$ parajsti siis, kui $\alpha = c_1$ ja $\alpha_2 = \mathsf{true}$ parajasti siis, kui $\alpha = c_2$. Tehes asendused esimeses võrrandis saame $\lceil \Pi_{f,\alpha,C_1} \rceil = \neg\mathsf{true} = \mathsf{false}$. See tulemus on aga ilmselgelt vale, kuna f tagastab c_1, kui $z = c_1$. Õige tulemus oleks $\lceil \Pi_{f,\alpha,C_1} \rceil = \mathsf{true}$.

Soovime lahendada süsteeme E_{NC} ja E_{SC} iteratiivselt püsipunkti arvutades, asendades funktsioonide tagastusväärtusi vastavate substitutsioonidega. Nagu eelnevas näites, ei pruugi asendusi tehes säilida tugevaim vajalik ja nõrgim piisav tingimus, kui süsteemid ei ole monotoonsed.

Teisendame E_{NC} ja E_{SC} monotoonseteks süsteemideks $\mathcal{T}_{\mathrm{NC}}$ ja $\mathcal{T}_{\mathrm{NC}}$ nii, et

1. võrrandid ei sisalda tagastusmuutujate eitust
2. $\mathrm{SAT}^*(E_{\mathrm{NC}}) \Leftrightarrow \mathrm{SAT}(\mathcal{T}(E_{\mathrm{NC}}))$
3. $\mathrm{VALID}^*(E_{\mathrm{SC}}) \Leftrightarrow \mathrm{VALID}(\mathcal{T}(E_{\mathrm{SC}}))$

Esimene tingimus definitsioonis tagab, et tugevaim vajalik ja nõrgim piisav tingimus kehtib vastavas süsteemis ka peale substitutsiooni. Teine ja kolmas tingimus garanteerivad, et antud valemid säilitavad kehtestatavust ja samaselt tõesust. Lemmad 5-8 väidavad, et iga E_{NC} ja E_{SC} jaoks leiduvad vajalikud monotoonsed süsteemid mis rahuldavad antud tingimusi. Lemmade 5 ja 7 tõestused annavad juhendi, kuidas vastavaid süsteeme konstrueerida.

Lemma 5. *Süsteemi E_{NC} jaoks leidub süsteem $\mathcal{T}(E_{NC})$, kus kõik tagastusmuutujad esinevad monotoonselt ja iga $\phi \in E_{NC}$ ja $\phi' \in \mathcal{T}(E_{NC})$ korral $SAT^*(\phi) \Leftrightarrow SAT(\phi')$.*

Tõestus. Lemma 1 järgi $\mathrm{SAT}^*(\phi) \Leftrightarrow \mathrm{SAT}(\mathcal{M}(\phi)) \wedge \psi_{\mathrm{unik}}$ st. piisab kui näidata

$$\mathrm{SAT}(\phi') \Leftrightarrow \mathrm{SAT}(\mathcal{M}(\phi) \wedge \psi_{\mathrm{unik}}) \tag{24}$$

Leidmaks ϕ', konverteerime $\mathcal{M}(\phi)$ disjunktiivsele normaalkujule:

$$\phi' = \mathrm{DNF}(\mathcal{M}(\phi)) = (p_{11} \wedge \cdots \wedge p_{1n}) \wedge (p_{m1} \wedge \cdots \wedge p_{mn}) \tag{25}$$

Saame rahuldama unikaalsust, kui asendame vastuolud disjunktides st. kui valemis $(p_{i1} \wedge \cdots \wedge p_{in})$ esineb nii Π_{f,α,C_i} ja Π_{f,α,C_j} kus $i \neq j$ siis asendame selle klausli false-ga. Saadud valem ϕ' on samaaegselt kehtestatav kui $\mathcal{M}(\phi) \wedge \psi_{\mathrm{unik}}$ □

Lemma 6. *Olgu phi $\in \mathcal{T}(E_{NC})$ valem, mis sisaldab literaale $\overrightarrow{\alpha}$ ja $\overrightarrow{\Pi}$. Olgu $\mathcal{F}$ tugevaim vajalik tingimus mis tagastusmuutujale Π_{α,C_i}, mis sisaldab ainult jälgitavaid muutujaid. Siis tugevaim vajalik tingimus ϕ jaoks mis ei sisalda Π_{α,C_i} on*

$$\lceil \phi \rceil = \phi[\mathcal{F}/\Pi_{\alpha,C_i}] \tag{26}$$

Tõestus. Esmalt näitame, et $\phi[\mathcal{F}/\Pi_{\alpha,C_i}]$ on vajalik tingimus ϕ jaoks.

1. Kui $\phi = \Pi_{\alpha,C_i}$, siis $\phi[\mathcal{F}/\Pi_{\alpha,C_i}] = \mathcal{F}$. Siis $\phi \Rightarrow \phi[\mathcal{F}/\Pi_{\alpha,C_i}]$ kuna $\Pi_{\alpha,C_i} \Rightarrow \mathcal{F}$. Kui $\phi = v$ kus $v \neq \Pi_{\alpha,C_i}$, siis $\phi \Rightarrow \phi[\mathcal{F}/\Pi_{\alpha,C_i}]$ kuna $\phi[\mathcal{F}/\Pi_{\alpha,C_i}] = \phi$.

2. Me ei pea vaatama juhtu $\neg\Pi_{\alpha,C_i}$ kuna Π_{α,C_i} võis esineda ainult monotoonselt.

3. $\phi = \phi_1 \wedge \phi_2$: Oletame, et $\phi_1 \wedge \phi_2 \not\Rightarrow (\phi_1 \wedge \phi_2)[\mathcal{F}/\Pi_{\alpha,C_i}]$ nii, et

$$\phi_1 \wedge \phi_2 \not\Rightarrow (\phi_1[\mathcal{F}/\Pi_{\alpha,C_i}]) \wedge (\phi_2[\mathcal{F}/\Pi_{\alpha,C_i}]) \tag{27}$$

   Sellisel juhul leidub tõeväärtuste väärtustus $\bar{v}$ mis rahuldab $\phi_1 \wedge \phi_2$ kuid ei rahulda $(\phi_1[\mathcal{F}/\Pi_{\alpha,C_i}]) \wedge (\phi_2[\mathcal{F}/\Pi_{\alpha,C_i}])$. Induktsiooni kohaselt

$$\phi_1 \Rightarrow \phi_2[\mathcal{F}/\Pi_{\alpha,C_i}] \wedge \phi_2[\mathcal{F}/\Pi_{\alpha,C_i}] \tag{28}$$

   Seega, kui ϕ_1 ja ϕ_2 on tõesed siis ka $(\phi_1[\mathcal{F}/\Pi_{\alpha,C_i}]) \wedge (\phi_2[\mathcal{F}/\Pi_{\alpha,C_i}])$ on tõene, mis tekitab vastuolu võrrandiga 27.

4. $\phi = \phi_1 \wedge \phi_2$: Sarnaselt eelmise punktiga

Nüüd näitame, et $\phi[\mathcal{F}/\Pi_{\alpha,C_i}]$ on tugevaim piisav tingimus ϕ jaoks $\phi[\mathcal{F}/\Pi_{\alpha,C_i}] \Rightarrow \phi[\mathcal{F}''/\Pi_{\alpha,C_i}]$ iga piisava tingimuse $\mathcal{F}''$ jaoks.

1. $\phi = \Pi_{\alpha,C_i}$: Olgu $\phi[\mathcal{F}''/\Pi_{\alpha,C_i}]$ mingi vajalik tingimus ϕ jaoks. Kuna $\mathcal{F}$ on tugevaim vajalik tingimus Π_{α,C_i} jaoks, saame $\mathcal{F} \Rightarrow \mathcal{F}''$ ning $\phi[\mathcal{F}/\Pi_{\alpha,C_i}] \Rightarrow \phi[\mathcal{F}''/\Pi_{\alpha,C_i}]$.

2. $\phi = v$, $v \neq \Pi_{\alpha,C_i}$ on triviaalne juhtum

Induktsiooni samm:

1. Olgu $\phi = \phi_1 \wedge \phi_2$ ja $\phi[\mathcal{F}''/\Pi_{\alpha,C_i}]$ mingi vajalik tingimus nii, et

$$\phi[\mathcal{F}/\Pi_{\alpha,C_i}] \not\Leftarrow \phi[\mathcal{F}''/\Pi_{\alpha,C_i}] \tag{29}$$

   see tähendab, et eksisteerib tõeväärtuste väärtustus $\bar{v}$, mis rahuldab tingimust $\phi[\mathcal{F}/\Pi_{\alpha,C_i}]$ aga mitte $\phi[\mathcal{F}''/\Pi_{\alpha,C_i}]$. Induktsiooni järgi

$$\phi_1[\mathcal{F}/\Pi_{\alpha,C_i}] \Leftarrow \phi_1[\mathcal{F}''/\Pi_{\alpha,C_i}] \wedge \tag{30}$$
$$\phi_2[\mathcal{F}/\Pi_{\alpha,C_i}] \Leftarrow \phi_2[\mathcal{F}''/\Pi_{\alpha,C_i}] \tag{31}$$

   tähendab

$$(\phi_1[\mathcal{F}/\Pi_{\alpha,C_i}] \wedge \phi_2[\mathcal{F}/\Pi_{\alpha,C_i}]) \wedge (\phi_1[\mathcal{F}''/\Pi_{\alpha,C_i}] \wedge \phi_2[\mathcal{F}''/\Pi_{\alpha,C_i}]) \tag{32}$$

   seega $\bar{v}$ peab rahuldama $\phi[\mathcal{F}/\Pi_{\alpha,C_i}]$ mis on vastuolu.

2. $\phi = \phi_1 \vee \phi_2$ juhtum on sümmeetriline

□

Lemma 7. *Iga süsteemi E_{SC} jaoks leidub süsteem $\mathcal{T}(E_{SC})$, kus kõik tagastuspredikaadid esinevad monotoonselt ja iga $\phi \in E_{SC}$ ja $\phi' \in \mathcal{T}(E_{SC})$ nii, et VALID$^*(\phi) \Leftrightarrow$ VALID(ϕ').*

Tõestus. Lemma 2 järgi VALID$^*(\phi) \Leftrightarrow \{\psi_{\text{eks}} \models \mathcal{M}(\phi)\}$. Seega piisab näidata, et leidub ϕ' nii, et $(\{\psi_{\text{eks}} \models \phi\}) \Leftrightarrow (\{\} \models \phi')$. Konstrueerime ϕ', kui teisendame $\mathcal{M}(\phi)$ konjunktiivsele normaalkujule:

$$\phi' = \text{CNF}(\phi) = (p_{11} \vee \cdots \vee p_{1n}) \wedge \cdots \wedge (p_{m1} \vee \cdots \vee p_{mn}) \tag{33}$$

Seejärel eemaldame tautoloogiad, asendades kõik klauslid, mis sisaldavad Π_{α,C_i} iga $1 \leq i \leq n$, väärtusega true. On lihtne näha, et $(\{\psi_{\text{eks}} \models \phi\}) \Leftrightarrow (\{\} \models \phi')$. □

Lemma 8. *Olgu $\phi \in \mathcal{T}(E_{SC})$ valem, mis sisaldab literaale $\overrightarrow{\alpha}$ ja $\overrightarrow{\Pi}$. Olgu $\mathcal{F}$ nõrgim piisav tingimus Π_{α,C_i}. Siis nõrgim piisav tingimus ϕ jaoks, mis ei sisalda Π_{α,C_i} on*

$$\lfloor \phi \rfloor = \phi[\mathcal{F}/\Pi_{\alpha,C_i}] \tag{34}$$

Tõestus. Tõestus on sarnane lemma 6 tõestusega. □

7 Rekursiooni elimineerimine

Peale kõigi mittejälgitavate muutujate elimineerimist ja vastavate monotoonsete omaduste rahuldamiseks tehtud transformatsioone saame järgmisel kujul valemid:

$$\mathcal{T}(E_{\text{NC}}) = \begin{bmatrix} \mathcal{T}(\lceil \Pi_{f_1,\alpha,C_1} \rceil) = \phi_{11}(\overrightarrow{\alpha}_1, \mathcal{T}(\lceil \overrightarrow{\Pi} \rceil)[\overrightarrow{b}_1/\overrightarrow{\alpha}_i]) \\ \cdots \\ \mathcal{T}(\lceil \Pi_{f_k,\alpha,C_n} \rceil) = \phi_{kn}(\overrightarrow{\alpha}_k, \mathcal{T}(\lceil \overrightarrow{\Pi} \rceil)[\overrightarrow{b}_k/\overrightarrow{\alpha}_i]) \end{bmatrix} \tag{35}$$

$$\mathcal{T}(E_{\text{SC}}) = \begin{bmatrix} \mathcal{T}(\lfloor \Pi_{f_1,\alpha,C_1} \rfloor) = \phi_{11}(\overrightarrow{\alpha}_1, \mathcal{T}(\lfloor \overrightarrow{\Pi} \rfloor)[\overrightarrow{b}_1/\overrightarrow{\alpha}_i]) \\ \cdots \\ \mathcal{T}(\lfloor \Pi_{f_k,\alpha,C_n} \rfloor) = \phi_{kn}(\overrightarrow{\alpha}_k, \mathcal{T}(\lfloor \overrightarrow{\Pi} \rfloor)[\overrightarrow{b}_k/\overrightarrow{\alpha}_i]) \end{bmatrix} \tag{36}$$

Vaatame Boole'i võrrandeid $\overrightarrow{\gamma}$ üle muutujate α_{ij}, mis ei sisalda mittejälgitavaid ega tagastusmuutujaid. Defineerime võre L Boole'i väärtuste vektoritest järgmiselt

$$\bot_{NC} = \overrightarrow{\text{false}}^{n*m} \qquad \bot_{SC} = \overrightarrow{\text{true}}^{n*m}$$

$$\top_{NC} = \overrightarrow{\text{true}}^{n*m} \qquad \top_{SC} = \overrightarrow{\text{false}}^{n*m}$$

$$\overrightarrow{\gamma}_1 \sqcup_{NC} \overrightarrow{\gamma}_2 = \langle \ldots, \gamma_{1l} \vee \gamma_{2i}, \ldots \rangle \qquad \overrightarrow{\gamma}_1 \sqcup_{SC} \overrightarrow{\gamma}_2 = \langle \ldots, \gamma_{1l} \wedge \gamma_{2i}, \ldots \rangle$$

Võre L on lõplik (loogilise ekvivalentsuseni), kuna muutujate α_{ij} arv on lõplik ja seega on loogiliselt erinevaid valemeid lõplik arv. Defineerime funktsioonid

$$F_{NC}(\overrightarrow{\gamma}_{NC}) = \langle \ldots, \phi_{ij}[\overrightarrow{\gamma}_{NC}/\mathcal{T}(\lceil \overrightarrow{\Pi} \rceil)], \ldots \rangle \tag{37}$$

$$F_{SC}(\overrightarrow{\gamma}_{SC}) = \langle \ldots, \phi_{ij}[\overrightarrow{\gamma}_{SC}/\mathcal{T}(\lfloor \overrightarrow{\Pi} \rfloor)], \ldots \rangle \tag{38}$$

mis asendavad tagastusmuutujad $\mathcal{T}(\overrightarrow{\Pi}_{\alpha,C_i})$ Boole'i võrranditega $\overrightarrow{\gamma}$. Arvutame E_{NC} vähima püsipunkti $\mathit{fix}(F_{NC}(\bot_{NC}))$ ja E_{SC} vähima püsipunkti $\mathit{fix}(F_{SC}(\bot))$. Püsipunktid leiduvad, kuna mõlemad süsteemid on monotoonsed vastavalt $\mathcal{T}(\lceil\overrightarrow{\Pi}\rceil)$ ja $\mathcal{T}(\lfloor\overrightarrow{\Pi}\rfloor)$

Näide 8. Kasutades näite 4 funktsiooni *f*, arvutame nõrgimat piisavat tingimust, et tagastatakse väärtus 1.

$$
\begin{aligned}
\mathrm{WSC}_0(\Pi_{f,x,1}) &= \mathsf{true}\\
\mathrm{WSC}_1(\Pi_{f,x,1}) &= (v_{x1} \vee \mathsf{true}) \vee (\neg(v_{x1} \vee \mathsf{true}) \wedge \ldots) \wedge\\
&\quad (v_{x1} \vee \mathsf{false}) \vee (\neg(v_{x1} \vee \mathsf{false}) \wedge (\mathrm{WSC}_0(\Pi_{f,x,1})[\mathsf{true}/v_{x1}] \vee \mathrm{WSC}_0(\Pi_{f,x,1})[\mathsf{false}/v_{x1}]))\\
&= \mathsf{true} \wedge\\
&\quad v_{x1} \vee (\neg(v_{x1} \wedge (\mathsf{true} \vee \mathsf{true})\\
&= v_{x1} \vee \neg v_{x1} = \mathsf{true}
\end{aligned}
$$

Näeme, et püsipunkt saavutub esimesel algoritmi iteratsioonil ja funktsioon *f* peab väljastama väärtuse 1. (Oletusel, et funktsioon termineerub.)

8 Võrdlus teiste lähenmistega

Mitmed mudelit kontrollivad analüsaatorid põhinevad ümberlükkava mudeli otsimisel[1, 5], kus iteratiivselt täpsustatakse abstraktsiooni, kuni vastav omadus on tõestatud või ümberlükatud. Sellised algoritmid ei pruugi lõpptulemuseni jõuda, kuna järjest täpsustatud abstraktsioon ei pea koonduma. Siin antud algoritm ei vaja iteratiivset täpsustamist ning töötab palju suurematel programmidel, kui ümberlükkavaid analüüsijaid. Samas ei pruugi mudelit kontrollival analüsaatoril olla lõpliku väärtustedomeedi ingimust.

Siiani on predikaatabstraktsiooni kasutavad analüsaatorid olnud väheskaleeruvad, kuna paljud predikaadid ei ole lõpptulemuse jaoks olulised. Craig'i interpolatsiooni[5] kasutades suudetakse leida lokaalselt tähtsaid predikaate. Siin kirjeldatud meetod üritab teha sama, kuid pole vaja otsida ümberlükkavat näidet ega mitut iteratsiooni.

Mitmed paremini skaleeruvad raja- ja kontekstitundliku analüüsi meetodid on olnud mittekorrektsed või mittetäielikud. Näiteks ESP[3], kus tingimusavaldise tõttu raja hargnemist vaadatakse eraldi, kui seisund on eristatav. ESP kasutab mitmeid heuristikaid ning ei pruugi avastada kõige paremaid rajatundlike tingimusi. Samuti analüsaator Saturn[6], mis pole üldiselt interprotseduraalselt rajatundlik – kasutab heuristikaid leidmaks rohkem tähtsaid predikaate.

9 Implementatsioon

Kirjeldatud meetod implementeeriti analüsaatoris Saturn, millega laiendati Saturni infrastruktuuri võimalusega pärida kliendi analüüsis funktsiooni tagastusväärtuse andmise tugevaimat vajalikku ja nõrgimat piisavait tingimust. Funktsiooni tagastusväärtused ja kõrvalefektid antakse edasi vabade muutujate kaudu.

Kuigi kirjeldatud tehnika jaoks on vajalik, et kõik võimalikud tagastusväärtused arvesse võetaks, ei pruugi väärtuste arv olla lõplik. Analüüsi jaoks on olulised need funktsioonide tagastusväärtused, mida programmis päritakse. Seepärast kogutakse kokku kõik funktsiooniväärtuste ja kõrvalefektide predikaadid, kus neid võrreldakse literaalidega. Näiteks, kui koodis esineb *if (foo(a)==3) ...*, siis arvutatakse $\Pi_{\text{foo},\alpha,3}$ ja $\neg\Pi_{\text{foo},\alpha,3}$ tugevaim vajalik ja nõrgim piisav tingimus. Selliselt toimides on võimalik leida lõplik hulk huvitavaid funktsioonide tagastusväärtusi. Paneme tähele, et selliselt käitudes kaotame täpsust, näiteks juhul kui võrreldakse kahe erineva funktsiooni f ja g tagastusväärtusi.

Antud algoritm arvutab vähima püsipunkti, kuid Saturnil võib ebaõnnestuda selle leidmine arvutusressurside puudumisel. See tooks kaasa mittekorrektse üldistuse. Sellepärast arvutatakse hoopis suurim püsipunkt, kuna siis võime arvutamise igal iteratsioonil peatada ja tulemus oleks võib olla ebatäpne kuid siiski korrektne. Suurima püsipunkti arvutamine kaotab täpsust mittedetermineerivate funktsioonide juures, näiteks

$$\textit{define } f(x) = \textit{if } (f(x) = c_1) \textit{ then } c_1 \textit{ else } c_2$$

puhul on f tagastamise c_1 suurim püsipunkt true tugevaima vajaliku tingimuse jaoks, kuid vähim püsipunkt tagastab samal juhul false.

Algoritmide esitamiseks kasutatud keeles on funktsioonidel üks tagastusväärtus. Sellise kitsendusega on turvaline elimineerida kõik mittejälgitavad muutujad. Kui funktsioonil on mitu tagastusväärtust nagu näiteks

```
int foo(int **p){
  int *x = malloc(sizeof(int));
  if (!x) return -1;
  *p = x;
  return 1;
}
```

puhul, võivad olulised saada seosed erinevate tagastusväärtuste vahelised seosed. Sellistel juhtudes Saturn loob muutujad, mis kirjeldavad väljund seisundeid väliselt jälgitavate seoste vahel.

10 Eksperimentaalsed tulemused

Antud meetodeid kasutades kontrueeriti kaks komplekti eksperimente. Esiteks koostati Saturni analüüs, mis väljastas tingimused, millal antud programmis loetakse viitadest andmeid või luuakse viitu. Selline eksperiment võiks olla hea test, kuna viitadest lugemine on C koodis üldlevinud operatsioon, ja samuti näidata tingimuste keskmist suurust. Lähtekoodina kasutati Linux kernelit, Sambat ja OpenSSH'd.

Saadi järgmised tulemused:

	Linux 2.6.17.1	Samba 3.0.23b	OpenSSH 4.3p2
Keskmine algsete valvurite suurus	3,00	4,45	3,02
Keskmine NC suurus (viitast lugemine)	0,75	1,02	0,75
Keskmine SC suurus (viitast lugemine)	0,48	0,67	0,50
Keskmine NC suurus (malloc)	2,39	2,82	1,39
Keskmine SC suurus (malloc)	0,45	0,49	0,67
Keskmine viida loomise ja lugemise kaugus	5,98	4,67	2,03
Koodi suurus (ridades)	6'275'017	515'689	155'660

Antud tabelis algsete valvuri suurus sisaldab mittejälgitavaid muutujaid ja tingimuste suuruse all mõeldakse Boole'i operatsioonide arvu. Viida loomise ja lugemise kaugus on kaugus väljakutseahelas.

Teine eksperimendina loodi Saturni *null* viitadest lugemise avastamise analüüsi. Konkreetne analüüs valiti, kuna see tihti nõuab väga keeruliste raja tingimuste jälgimist. Kõigepealt päritakse tugevaim vajalik tingimus g_1, et viit p on $NULL$, ja tugevaim vajalik tingimus g_2, et viidast p loetakse. Null viida viga on võimalik, kui $\mathrm{SAT}(g_1 \wedge g_2)$. Saturn teeb alt-üles analüüsi ja annab veahoiatuse, kui leitakse tee $NULL$ väärtusest kuni selle väärtusega viidast lugemiseni.

Eksperiment käivitati esmalt täielikult (interprotseduraalselt) rajatundlikult ning seejärel intraprotseduraalselt rajatundlikult ja interprotseduraalselt rajatundmatult Linuxi, Samba ja OpenSSH koodil. Järgnevas tabelis on ära toodud Saturni poolt antud hoiatused, kus ei ole tegemist massiivide ega rekursiivsete struktuuridega. Massiivide ja rekursiivsete struktuuride korralik käsitlemise probleem on ortogonaalne ja hetkel nendest tekkivaid veasituatsioone ei arvestata.

	interprotseduraalselt rajatundlik			intraprotseduraalselt rajatundlik		
	Linux 2.6.17.1	Samba 3.0.2.23b	OpenSSH 4.3.p2	Linux 2.6.17.1	Samba 3.0.2.23b	OpenSSH 4.3.p2
veahoiatusi	171	48	3	1'495	379	21
vigasid	134	17	1	134	17	1
valepositiive	37	25	2	1'344	356	20
uurimise all	17	6	0	17	6	0
hoiatusi / vigu	1,3	2,8	3	11,2	22,3	21

Interprotseduraalselt rajatundliku analüüs vähendab valepositiivide arvu peaaegu ühe suurusjärgu võrra ilma heuristikaid kasutamata ega korrektsusest loobumata. Paneme tähele, et valepositiivide olemasolu ei ole vastuolus sellega, et analüüs on täielik. Isegi lõplike domeenide korral võivad tekkida ebatäpsused analüüsi aja lõppemisega, *inline assembly* kasutusel ja implementatsioonivigade tõttu. Kui lõplike domeenide korral on tehnika täielik, lõpmatute domeenide korral ei pruugi antud tehnikaga olla võimalik arvutada tugevaimat vajalikku ja nõrgimat piisavat tingimust. Näiteks, kui rajatingimustes on kasutatud aritmeetikat.

11 Kokkuvõte

Töös on kirjeldatud uus programmianalüüsi meetod, millega saab arvutada täpsed vajalikud ja piisavad tingimused funktsiooni mingi konkreetse tagastusväärtuse jaoks. Kusjuures tingimus võib olla konteksti- ja rajatundlik ning seda isegi rekursiivsete funktsioonide korral. Algoritmi autorid on näidanud, et see antud meetod skaleerub suurtele programmidele, suutes oluliselt lihtsustada rajatundlike tingimusi. Meetod sakleerub isegi nii hästi, et sellega on võimalik uurida *NULL* viitade käitumis suurimates vabavaralistes C programmides. Toodud tehnika puudusteks saab välja tuua analüüsi tugeva sõltuvuse literaalidest ja ebatäpsused lõpmatute domeenidega tegelemisel – sealjuures aritmeetika kasutamisel tingimusavaldistes.

Siin kirjeldatud tehnikad implementeeriti olemasolevasse Saturn raamistikku, millega muudeti see raamistik üheks põnevaimaks omalaadsete seas. Järgnev töö võiks olla seotud Saturni täiustamisega, et see suudaks analüüsida ka rekursiivseid struktuure ja massiive.

Viited

[1] T. Ball and S.K. Rajamani. Automatically Validating Temporal Safety Properties of Interfaces. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 103–122, 2001.

[2] G. Boole. *An Investigation of the Laws of Thought*. Dover Publications, 1958.

[3] Manuvir Das, Sorin Lerner, and Mark Seigle. Esp: path-sensitive program verification in polynomial time. In *Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation*, pages 57–68, Berlin, Germany, 2002. ACM.

[4] Isil Dillig, Thomas Dillig, and Alex Aiken. Sound, complete and scalable path-sensitive analysis. *SIGPLAN Not.*, 43(6):270–280, 2008.

[5] T.A. Henzinger, R. Jhala, R. Majumdar, and K.L. McMillan. Abstractions from proofs. *ACM SIGPLAN Notices*, 39(1):232–244, 2004.

[6] Yichen Xie and Alex Aiken. Scalable error detection using boolean satisfiability. *SIGPLAN Not.*, 40(1):351–363, 2005.