Апстрактна интерпретација — Теоријске основе апстрактне интерпретације. Примери.

Семинарски рад у оквиру курса
Верификација софтвера
Математички факултет

Димитрије Шпадијер, 1084/2018
mm11021@alas.matf.bg.ac.rs

11. децембар 2018.

Апстракт

Рачунарски системи су постали незаменљиви део живота савременог човека. Скоро да не постоји област људске делатности у којој се не користе рачунари и неки софтвер специјализован за ту делатност. Неисправан софтвер може изазвати озбиљне проблеме, па је веома важно утврдити да ли је софтвер исправан. Тиме се бави верификација софтвера. Једна од техника верификације софтвера је апстрактна интерпретација. У овом раду биће приказана теоријска основа ове технике провере исправности софтвера, као и неки примери.

Садржај

1 Увод **2**

2 Скуп путања програма **2**

3 Апроксимација **3**
3.1 Уређење скупова 3
3.2 Галуаове везе 4

4 Апстрактна интерпретација **5**

5 Закључак **7**

Литература **7**

1 Увод

Верификација софтвера бави се провером да ли дати софтвер задовољава дату спецификацију, односно да ли резултат рада софтвера за дати улаз одговара спецификацији софтвера. У верификацији софтвера издвајају се технике статичке и динамичке верификације софтвера[5]. Технике динамичке верификације софтвера утврђују исправност софтвера покрећући га, извршавајући га и проверавајући да ли резултат одговара спецификацији. С друге стране, технике статичке верификације утврђују исправност софтвера не покрећући га, већ анализирајући његов код.

Статичка верификација софтвера може се вршити ручно — прегледањем изворног кода и његовом провером. Ручно верификовати софтвер је веома споро, па је пожељно аутоматизовати овај процес, уколико је то могуће. Неки од метода која омогућава аутоматизацију просеца верификације софтвера јесу формални методи верификације. Формални методи засновани су на томе да се услови исправности софтвера искажу математичким тврђењима на формалном језику неке математичке теорије. Метод апстрактне интерпретације је један формали метод верификације софтвера и заснован је на теорији апроксимације [1].

Најзанимљивија својства програма су неодлучива, тј. није могуће конструисати алгоритам који их утврђује потпуно прецизно. Познато је да је халтинг проблем[4] неодлучив. Већина својстава програма могу се свести на халтинг проблем. Ипак, неодлучивост халтинг проблема није потпуно ограничавајућа. Морамо се одрећи захтева да алгоритмом *потпуно прецизно* утврдимо да ли неко својство програма важи или не. Зато је неопходно вршити апроксимације.

Апроксимације које се врше морају испуњавати следећи услов: ако за конкретан улаз софтвер даје сигуран одговор, онда је у апроксимацији одговор потврдан. То значи да ако за неки улаз програм не даје одговор (између осталог због тога што се не зауставља), у апроксимацији одговор не може сигурно бити потврдан, па је онда одговор „можда". Да би се појам апроксимације формализовао, прво је потребно формализовати појам програма.

2 Скуп путања програма

Нека је дат једноставан програмски језик, чије су наредбе *halt*, додела, условни скок, улаз и излаз. Скуп ових наредби нека је означен са C.

Дефиниција 2.1. *Програм* је пресликавање $Prog : K \longrightarrow C$, где је $K = \{0, 1, \ldots, k-1\} \subset \mathbb{N}$.

Другим речима, програм представља низ наредби. Овај низ наредби је коначан. Приликом извршавања програма, гледа се вредност бројача (pc), који говори која се наредба следећа извршава. Нека су изрази који се појављују у програмску језику изрази с целим бројевима ($\mathbb{Z}$). Нека ρ представља пресликавање променљивих у вредности ($Var \longrightarrow \mathbb{Z}$). Оно за променљиву x може дати недефинисану вредност, ако је извршена наредба доделе израза $E/0$ променљивој x, где је E неки израз, или ако променљивој x није још увек додељена вредност. Недефинисана вредност биће означена са $\perp$. *Стање програма* је уређен пар (pc, ρ). Семантика израза се дефинише на следећи начин:

1. $[\![x]\!]_\rho = \rho(x)$,
2. $[\![E_1 + E_2]\!]_\rho = [\![E_1]\!]_\rho + [\![E_2]\!]_\rho$, итд.

Семантика наредби се дефинише као пресликавање стања програма (pc, ρ) у ново стање програма (pc', ρ') тако да важи:

1. ако је $Prog(pc)$ додела $x := E$, онда је $pc' = pc + 1$, а ρ' се добија од ρ променом $x \mapsto [\![E]\!]_\rho$,
2. ако је $Prog(pc)$ улаз *input* x, онда је $pc' = pc + 1$, а ρ' се добија од ρ променом $x \mapsto v$, за унето $v \in \mathbb{Z}$,
3. ако је $Prog(pc)$ улаз *print* E, онда је $pc' = pc + 1$, $\rho' = \rho$,
4. ако је $Prog(pc)$ условни скок $if\ E\ goto\ n$, онда је $pc' = n$, $\rho' = \rho$, ако је $[\![E]\!]_\rho \neq 0$, а иначе је $pc' = pc + 1$, $\rho' = \rho$.
5. ако је $Prog(pc) = halt$, онда је $pc' = k$, $\rho' = \rho$ (наредна наредба је ван домена пресликавања $Prog$, што означава његов крај)

Почетно стање програма је $(0, \rho)$, где је $\rho : x \mapsto \bot$. Извршавање програма је низ стања (pc, ρ) и тај низ се назива *путањом програма*. У зависности од улаза који задаје корисник, постоје различите путање програма. Зато ће семантика програма бити скуп његових путања, за све могуће улазе.

3 Апроксимација

Ако програм задовољава неко својство, доказивање тога своди се на доказивање чињенице да све путање програма задовољавају неки услов. Да би се то спровело, скуп путања програма се *апроксимира одозго* тако што се одабере скуп који је надскуп скупа путања. Речено је да због неодлучивости није могуће анализирати својства софтвера потпуно прецизно, па се поставља питање која је анализа прецизнија. Одговор је да је прецизнија она анализа која даје мањи надскуп скупа трагова. Свака апроксимација скупа путања је коректна ако је већа од најмање коректне апроксимације. Најмања коректна апроксимација је, наравно, када се уопште не врши апроксимација, већ се узима тачан скуп путања програма.

Како год да се представљају путање програма, односно стања програма, неопходно је постојање уређења на том скупу. Зато је важно познавати математичку теорију о уређености скупова.

3.1 Уређење скупова

Дефиниција 3.1 (Парцијално уређен скуп)**.** *Парцијално уређен скуп* је уређен пар $(S, \leqslant)$, где је $\leqslant$ бинарна релација за коју важи:

1. рефлексивност: $\forall a \in S, a \leqslant a$,
2. антисиметричност: $\forall a, b \in S, a \leqslant b \wedge b \leqslant a \Longrightarrow a = b$,
3. транзитивност: $\forall a, b, c \in S, a \leqslant b \wedge b \leqslant c \Longrightarrow a \leqslant c$.

Дефиниција 3.2. Елемент $c \in S$ је *горња граница* скупа $X \subseteq S$ ако је $\forall x \in X, x \leqslant c$. Елемент $c \in S$ је *доња граница* скупа $X \subseteq S$ ако је $\forall x \in X, c \leqslant x$. Елемент $c \in S$ је *најмања горња граница* скупа $X \subseteq S$ ако је c горња граница скупа X и за сваку горњу границу $c' \in S$ важи $c \leqslant c'$. Елемент $c \in S$ је *највећа доња граница* скупа $X \subseteq S$ ако је c доња граница скупа X и за сваку доњу границу $c' \in S$ важи $c' \leqslant c$.

Релација уређења $\leqslant$ не мора бити потпуна, тј. није неопходно да важи ниједна од релација $a \leqslant b$, $b \leqslant a$.

Дефиниција 3.3 (Решетка)**.** *Решетка* је парцијално уређен скуп $(L, \leqslant)$ такав да за свака два елемента $a, b \in L$ постоји најмања горња граница $a \sqcup b$ и највећа доња граница $a \sqcap b$. Решетка се означава са $(L, \leqslant, \sqcup, \sqcap)$.

Према дефиницији решетке, за сваки двочлани подскуп $X \subset L$ постоји најмања горња граница. Одатле следи да она постоји за сваки коначан подскуп елемената. Међутим, то не мора бити случај за бесконачне подскупове.

Дефиниција 3.4 (Потпуна решетка)**.** *Потпуна решетка* је парцијално уређен скуп $(L, \leqslant)$ такав да сваки подскуп $X \subseteq S$ има најмању горњу границу $\sqcup X$ и највећу доњу границу $\sqcap X$. Специјално, скуп L тада има највећи елемент $\top = \sqcup L$ и најмањи елемент $\bot = \sqcap L$. Потпуна решетка се означава са $(L, \leqslant, \bot, \top, \sqcup, \sqcap)$.

Релација $\leqslant$ има смисао одређивања прецизности, тј. ако је $a \leqslant b$, онда је a прецизнији од b. На пример, интервал $[a, b]$ је прецизнији од интервала $[a', b']$ ако је садржан у њему, тј. ако је $a \geqslant a'$ и $b \leqslant b'$, и тада пишемо $[a, b] \leqslant [a', b']$. На пример, у скупу целих бројева можемо посматрати потпуну решетку интервала

$$\{\emptyset\} \cup \{[a, b] |\ a \leqslant b, a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{+\infty\}\},$$

уз малу злоупотребу нотације да и неограничене интервале $]-\infty, b]$, $[a, +\infty[$ и $]-\infty, +\infty[$ означавамо са $[a, b]$. Ту је $\leqslant = \subseteq$, $\sqcup = \cup$, $\sqcap = \cap$, $\bot = \emptyset$ и $\top = [-\infty, +\infty]$.

Сакупљајућа семантика је семантика која рачуна скуп могућих путања. Формално се може дефинисати као

$$T_r = \{s_1 \to \cdots \to s_n |\ s_1 \text{ је почетно стање, } s_i \to s_{i+1} \text{ је дозвољено}\}.$$

Скуп достижних стања је $S_r = \{s |\ s_1 \to \cdots \to s \in T_r\}$. Скуп свих путања и стања су уређење релацијом инклузије ($\subseteq$) и чине комплетну решетку.

3.2 Галуаове везе

Појам Галуаове везе је од изузетног значаја за повезивање конкретних објеката са својим апстракцијама. Више о њима може се пронаћи у [2].

Дефиниција 3.5. Нека су $(L_1, \leqslant_1)$ и $(L_2, \leqslant_2)$ парцијално уређени скупови. *Галуаова веза* између L_1 и L_2 је уређени пар (α, γ), где су $\alpha : L_1 \longrightarrow L_2$ и $\gamma : L_2 \longrightarrow L_1$ пресликавања таква да важи

$$\forall x \in L_1, \forall y \in L_2, \alpha(x) \leqslant_2 y \Longleftrightarrow x \leqslant_1 \gamma(y).$$

Означава се са $(L_1, \leqslant_1) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (L_2, \leqslant_2)$. Парцијално уређен скуп $(L_1, \leqslant_1)$ је *конкретан*, а парцијално уређен скуп $(L_2, \leqslant_2)$ је *апстрактан*. Пресликавање α је *апстракција*, а пресликавање γ је *конкретизација*.

Особине $\alpha(x) \leqslant_2 y$ и њој еквивалентна особина $x \leqslant_1 \gamma(y)$ значе да је y добра апроксимација конкретног својства x. При томе је $\alpha(x)$ најпрецизнија апроксимација $x \in L_1$ у L_2, а елемент $\gamma(y)$ је најнепрецизнији елемент из L_1 који се може коректно апроксимирати елементом $y \in L_2$.

Пример 3.1. Нека је $\mathbb{Z}$ скуп целих бројева и нека је дата потпуна решетка $\mathcal{P}(\mathbb{Z})$. Једна апстракција [3] ове решетке је решетка садржи $\bot, c, \top$, за све $c \in \mathbb{Z}$, с уређењем $\bot \leqslant c \leqslant \top$. Галуаове везе су дате са:
$\alpha(\emptyset) = \bot$, $\alpha(\{c\}) = c$, $\alpha(S) = \top$
$\gamma(\bot) = \emptyset$, $\gamma(c) = \{c\}$, $\gamma(\top) = S$.

Пример 3.2. Парцијално уређени скуп $\mathcal{P}(\mathbb{Z})$ подскупова скупа целих бројева можемо апстраховати на основу знакова елемената [3]. У овом случају, конкретни скуп је $\mathcal{P}(\mathbb{Z})$, а апстрактни скуп је $\mathcal{P}(\{-, 0, +\})$. Галуаове везе су дате са:
$\alpha(\emptyset) = \emptyset$,
$\alpha(S) = \{+\}$, ако S садржи само позитивне бројеве,
$\alpha(\{0\}) = \{0\}$,
$\alpha(S) = \{-\}$, ако S садржи само негативне бројеве,
$\alpha(S) = \{0, +\}$, ако S садржи само нулу и позитивне бројеве,
$\alpha(S) = \{-, 0\}$, ако S садржи само нулу и негативне бројеве,
$\alpha(S) = \{-, +\}$, ако S садржи и позитивне и негативне бројеве, а не садржи нулу,
$\alpha(S) = \{-, 0, +\}$, ако S садржи и нулу и позитивне и негативне бројеве,
$\gamma(\emptyset) = \emptyset$, $\gamma(\{+\}) =]0, +\infty[$, $\gamma(\{0\}) = \{0\}$, $\gamma(\{-\}) =]-\infty, 0[$,
$\gamma(\{0, +\}) = [0, +\infty[$, $\gamma(\{-, 0\}) =]-\infty, 0]$, $\gamma(\{-, +\}) =]-\infty, 0[\cup]0, +\infty[$,
$\gamma(\{-, 0, +\}) = \mathbb{Z}$.

Пример 3.3. Парцијално уређени скуп $\mathcal{P}(\mathbb{Z})$ подскупова скупа целих бројева можемо апстраховати у затворене интервале. У овом случају је конкретни скуп је $\mathcal{P}(\mathbb{Z})$, а апстрактни скуп је потпуна решетка интервала, описана након дефиниције 3.4. Галуаове везе су дате са:
$\alpha(\emptyset) = \emptyset$, $\alpha(S) = [\min S, \max S]$ (ако S нема најмањи, тј. највећи елемент, онда је $\min S = -\infty$, односно $\max S = +\infty$),
$\gamma(S) = S$, за све S.

Став 3.1. *Пар (α, γ) је Галуаова веза ако и само ако су α и γ монотоне, $\alpha \circ \gamma(y) \leqslant_2 y$ и $\gamma \circ \alpha(x) \leqslant_1 x$.*

Доказ овог става може се пронаћи у [1]. Приметимо да се применом апстракције, па конкретизације губи прецизност ($\gamma \circ \alpha(x) \leqslant_1 x$).

Став 3.2. *Пресликавање α је „на" ако и само ако је γ „1–1" ако и само ако је $\alpha \circ \gamma = \mathrm{Id}$.*

4 Апстрактна интерпретација

У овом поглављу видећемо како изгледа апстрактна интерпретација програма написаних на једноставном програмском језику датом у поглављу 2. Преведимо прво семантику тог програмског језика у апстрактну семантику.

	Семантика	Сакупљајућа сем.	Апстракција	Апстр. сем.
Вредности	$\mathbb{Z}$	$(\mathcal{P}(\mathbb{Z}), \subseteq)$	$\underset{\alpha}{\overset{\gamma}{\leftrightharpoons}}$	$(L, \leqslant)$
Оператори	$+ : \mathbb{Z} \longrightarrow \mathbb{Z}$	$+ : \mathcal{P}(\mathbb{Z}) \longrightarrow \mathcal{P}(\mathbb{Z})$		$a^\sharp +^\sharp b^\sharp = \alpha(\gamma(a^\sharp) + \gamma(b^\sharp))$

Табела 1: Апстракција константи и константних израза

Табела 1 представља везу између конкретних врености и њихових апстракција. Коректност апстракције вредности x дата са $\alpha(x) \leqslant x^\sharp$,

или, еквивалентно, $x \leqslant \gamma(x^\sharp)$. Коректност апстракције оператора $+$ дефинисана је на следећи начин: $+^\sharp$ је коректна апстракција оператора $+$ ако и само ако из $\alpha(a) \leqslant a^\sharp$ и $\alpha(b) \leqslant b^\sharp$ следи $\alpha(a+b) \leqslant a^\sharp + b^\sharp$. Није тешко доказати да за оператор $+^\sharp$ дефинисан у табели 1 важи ово својство.

Следећи корак је апстраховати пресликавање $\rho : Var \longrightarrow \mathbb{Z}$ променљивих у вредности. Ово се врши у два корака. Први корак је:

$$\mathcal{P}(Var \longrightarrow \mathbb{Z}) \underset{\alpha_{R_1}}{\overset{\gamma_{R_1}}{\leftrightarrows}} Var \longrightarrow \mathcal{P}(\mathbb{Z}),$$

$\alpha_{R_1}(R_1) = \lambda x.\{\rho(x) |\ \rho \in R_1\}$, $\gamma_{R_1}(R_1^\sharp) = \{\lambda x.y |\ y \in R_1^\sharp(x)\}$. Наиме, с леве стране су скупови пресликавања променљивих у вредности, а њих апстрахујемо пресликавањима које свакој променљивој додељује скуп свих вредности које је та променљива имала у пресликавањима одговарајућег скупа с леве стране. Овом апстракцијом се губи веза између променљивих у истом стању програма.

Други корак је, наравно, прећи са $\mathcal{P}(\mathbb{Z})$ на одговарајућу апстракцију L. Ово се врши на следећи начин:

$$Var \longrightarrow \mathcal{P}(\mathbb{Z}) \underset{\alpha_{R_2}}{\overset{\gamma_{R_2}}{\leftrightarrows}} Var \longrightarrow L,$$

$\alpha_{R_2}(R_2) = \alpha \circ R_2$, $\gamma_{R_2}(R_2^\sharp) = \gamma \circ R_2^\sharp$. Галуаове везе између скупова стања и њихове финалне апстракције дате су са

$$\mathcal{P}(Var \longrightarrow \mathbb{Z}) \underset{\alpha_R}{\overset{\gamma_R}{\leftrightarrows}} Var \longrightarrow L,$$

$\alpha_R = \alpha_{R_2} \circ \alpha_{R_1}$, $\gamma_R = \gamma_{R_1} \circ \gamma_{R_2}$.

У табели 2 дат је опис апстракције променљивих и израза с променљивима.

Семантика	Сакупљајућа сем.	Апстр.	Апстр. сем.
$[\![E]\!]_\rho \in \mathbb{Z}$	$[\![E]\!]R \in \mathcal{P}(\mathbb{Z})$		$[\![E]\!]^\sharp R^\sharp \in L$
$[\![x]\!]_\rho = \rho(x)$			$[\![x]\!]^\sharp R^\sharp = R^\sharp(x)$
$[\![E_1 + E_2]\!]_\rho = [\![E_1]\!]_\rho + [\![E_2]\!]_\rho$			$[\![E_1 + E_2]\!]^\sharp R^\sharp = [\![E_1]\!]^\sharp R^\sharp +^\sharp [\![E_2]\!]^\sharp R^\sharp$

Табела 2: Апстракција израза с променљивима

При томе, $[\![E]\!]R$ представља скуп свих вредности израза E за вредности $\rho \in R$, тј. $[\![E]\!]R = \{[\![E]\!]_\rho |\ \rho \in R\}$. Коректност апстракције израза изражава се на следећи начин: ако је апстракција пресликавања ρ коректна, онда је резултат израза коректо апстрахован. Формално, ако је $\alpha_R(R) \leqslant R^\sharp$, онда је $\alpha_R([\![E]\!]R) \leqslant [\![E]\!]^\sharp R^\sharp$. Наравно, оператори се мењају одговарајућим апстракцијама (нпр. оператор $+$ мења се оператором $+^\sharp$).

Апстракција стања програма дата је у табели 3.

Семантика	Сакупљајућа сем.	Апстр.	Апстр. сем.
$s : PC \times (Var \longrightarrow \mathbb{Z})$	$S : \mathcal{P}(PC \times (Var \longrightarrow \mathbb{Z}))$	$\underset{\alpha_S}{\overset{\gamma_S}{\leftrightarrows}}$	$S^\sharp : PC \longrightarrow (\{\bot\} \cup (Var \longrightarrow L))$,

Табела 3: Апстракција израза с променљивима

Овде је $\alpha_S(S) = \lambda pc.\alpha'_R(\{\rho |\ (pc, \rho) \in S\})$, а $\alpha'_R(R)$ је исто што и $\alpha_R(R)$ ако је $R \neq \emptyset$, а $\alpha'_R(\emptyset) = \bot$. Наравно, онда је $\gamma'_R(R) = \gamma_R(R)$

за $R \neq \bot$ и $\gamma'_R(\bot) = \emptyset$. Конкретизација апстракције стања програма дата је са $\gamma_S(S^\sharp) = \{(pc, \rho) |\ \rho \in \gamma'_R(S^\sharp(pc))\}$. Преостаје још да се види апстракција наредби.

1. ако је $Prog(pc)$ додела $x := E$, онда је $pc' = pc+1$, а $R^{\sharp\prime}$ се добија од $R^\sharp$ променом $x \mapsto [\![E]\!]^\sharp R^\sharp$,
2. ако је $Prog(pc)$ улаз *input* x, онда је $pc' = pc + 1$, а $R^{\sharp\prime}$ се добија од $R^\sharp$ променом $x \mapsto \top$,
3. ако је $Prog(pc)$ улаз *print* E, онда је $pc' = pc + 1$, $R^{\sharp\prime} = R^\sharp$,
4. ако је $Prog(pc)$ условни скок $if\ x > 0\ goto\ n$, онда је $pc' = n$, а $R^{\sharp\prime}$ се добија од $R^\sharp$ променом $x \mapsto R^\sharp(x) \sqcap [1, +\infty[$, или је $pc' = pc+1$, а $R^{\sharp\prime}$ се добија од $R^\sharp$ променом $x \mapsto R^\sharp(x) \sqcap\ ]-\infty, 0]$.
5. ако је $Prog(pc) = halt$, онда је $pc' = k$, $R^{\sharp\prime} = R^\sharp$ (наредна наредба је ван домена пресликавања $Prog$, што означава његов крај)

Коректност апстракције изражава се на следећи начин: ако је $(pc, \rho) \mapsto (pc', \rho')$ и $\alpha(\{\rho\}) \leqslant R^\sharp$, онда постоји $R^{\sharp\prime}$ такав да је $(pc, R^\sharp) \mapsto (pc', R^{\sharp\prime})$ и $\alpha(\{\rho'\}) = R^{\sharp\prime}$.

Пример 4.1. Нека је дат програм

```
1: input x
2: if x <= 0 goto 4
3: print 10/x
4: input y
5: print y*y*x
6: halt.
```

Апстрактна интерпретација овог програма је

$$
\begin{array}{lcl}
1, R^\sharp & \longrightarrow & 2, R^\sharp[x \mapsto \top] \\
2, R^\sharp & \longrightarrow & 4, R^\sharp[x \mapsto R^\sharp(x) \sqcap\ ]-\infty, 0]\ ] \\
2, R^\sharp & \longrightarrow & 3, R^\sharp[x \mapsto R^\sharp(x) \sqcap [1, +\infty[\ ] \\
3, R^\sharp & \longrightarrow & 4, R^\sharp \\
4, R^\sharp & \longrightarrow & 5, R^\sharp[y \mapsto \top] \\
5, R^\sharp & \longrightarrow & 6, R^\sharp \\
6, R^\sharp & \longrightarrow & 7, R^\sharp.
\end{array}
$$

5 Закључак

У овом раду описана је теоријска основа апстрактна интерпретација програма. Формално је описан појам програма, наредбе, формално је уведена семантика једноставног програмског језика у домену целих бројева, као и појам скупа путања. Описано је како се, због ограничења која су последица неодлучивости халтинг проблема, врши апроксимација, како би се вршила верификација софтвера с делимично умањеном прецизношћу. Затим су дефинисани математички појмови парцијално уређеног скупа, решетке и потпуне решетке, као и Галуаове везе два парцијално уређена скупа. На крају је описано како се врши апстракција програма написаних на датом програмском језику.

Литература

[1] Bruno Blanchet. Introduction to abstract interpretation. 2002.

[2] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. pages 269–282.

[3] Xavier Rival. Abstract interpretation — semantics and applications to verification. École Normale Supérieure.

[4] A. M. Turing. On Computable Numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[5] Милена Вујошевић Јаничић. Верификација софтвера — материјали са предавања. Математички факултет, Универзитет у Београду, Србија, 2018.