Метода аналитичких таблоа у класичној исказној логици

Антоније Суботић, Стефан Миленковић

4. октобар 2021.

Сажетак

Циљ рада је да се практично реализује метода аналитичких таблоа у класичној исказној логици. Тако имплементиран програм би био једноставнији доказивач теорема ове логике и у суштини би испитивао да ли је нека формула таутологија и евентуално генерисао њене контрамоделе уз помоћ таблоа. Већина рада заснована је на књизи [1].

1 Увод

1.1 Класична исказна логика

Сам наслов одељка, данашњем свету математике и рачунарства веома је познат. Класична исказна логика, као подобалст математичке логике, бави се проучавањем формалног језика, теорија изграђених на њему и поступцима доказивања тврђења у тим теоријама. Међутим, ми се нећемо детаљно удубљивати у ову област (за више детаља видети [1]), већ ћемо увести основне појмове и дефиниције, јер без знања истих, читање поглавља која следе било би знатно отежано.

У свакодневној комуникацији људи користе просте и сложене реченице. За математичку логику, посебно су привлачне оне које могу имати вредност тачно или нетачно. Такве реченице називају се *исказима*. На пример, "Лопта је плава" је исказ, док "Поздрав", није исказ. Претходни исказ, не може се раставити на простије исказе. Такви искази називају се *атомски* искази. Уобичајена пракса је да се атомски искази означавају исказним словима (a, b, p, a_1 ...). Сложени искази граде се комбиновањем исказних слова, логичких везника и отворених и затворених заграда. Наведимо логичке везнике:

- *Негација* исказа p је "Није p", у ознаци $\neg p$.
- *Конјункција* исказа p и q је "p и q", у ознаци $p \wedge q$.
- *Дисјункција* исказа p и q је "p или q", у ознаци $p \vee q$.
- *Импликација* исказа p и q је "Ако p, онда q", у ознаци $p \to q$.
- *Еквиваленција* исказа p и q је "p ако и само ако q" у ознаци $p \leftrightarrow q$.

Пример 1.1. *Пример сложеног израза је "Ако је данас киша, сутра је понедељак или уторак."*

Преведимо на формални језик појам исказне формуле, увођењем одговарајуће дефиниције:

Дефиниција 1.1. *Исказне формуле* се дефинишу на следећи начин:

- Исказна слова су*атомске формуле.*
- Атомске формуле су исказне формуле.
- Ако су P и Q исказне формуле, тада су $\neg P$, $\neg Q$, $P \wedge Q$, $P \vee Q$, $P \to Q$ и $P \leftrightarrow Q$ исказне формуле.
- Исказне формуле се граде само коначном применом претходних правила.

$I(P)$	$I(Q)$	$I(\neg P)$	$I(P \wedge Q)$	$I(P \vee Q)$	$I(P \to Q)$	$I(P \leftrightarrow Q)$
$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\top$

Табела 1.1: Основне операције - Истинитосна таблица

Осврнимо се сада на пример 1.1. Ако изразу "Данас је киша" доделимо исказно слово p, изразу "Сутра је понедељак" слово q и изразу "Сутра је уторак" слово r, сложени исказ можемо написати као $p \to (q \vee r)$.

Као што смо већ рекли, при раду са исказним формулама занимају нас њихове истинитосне вредности. У случају класичне исказне логике, истинитосна вредност је један од два израза: тачно или нетачно. $\top$ означава истинитосну вредност тачно, а $\bot$ означава истинитосну вредност нетачно.

Дефиниција 1.2. *Интерпретација* I је пресликавање које исказним словима додељује једну од вредност из скупа $\top, \bot$. Вредност формуле P при интерпретацији I, у ознаци $I(A)$, је:

- Ако је $P = a$ исказно слово (атомска формула), онда је $I(P) = I(a)$.
- Ако су већ израчунате вредности вредности $I(P)$ и $I(Q)$ за исказне формуле A и B, вредности $I(\neg P)$, $I(P \wedge Q)$, $I(P \vee Q)$, $I(P \to Q)$ и $I(P \leftrightarrow Q)$ рачунамо на основу табеле 1.1.

Дефиниција 1.3. Формула P је *задовољива* ако постоји интерпретација I таква да је $I(P) = \top$. Скуп формула $\{P_1, P_2, ...P_n\}$ је *задовољив* ако постоји интерпретација за коју је $I(P_i) = \top$ за све $i \in \{1, 2, ...n\}$.

Математичарима су кроз векове објекти који имају специфичне особине увек били посебно занимљиви. Логика није изузетак. Стога ћемо дефинисати посебне типове формула, које су, у смислу броја интерпретација при којима су тачне, дефинитивно вредне дискусије. Шта више, оне су од круцијалног значаја за овај рад, јер се готово цео наставак истог, "интересује" баш за њих.

- Формула је *таутологија* (*ваљана формула*) ако је тачна при свакој интерпретацији.
- Формула је *контрадикција* ако је нетачна при свакој интерпретацији.
- Интерпретација I је *модел* за формулу P ако је $I(P) = \top$; I је *контрамодел* за P ако је $I(P) = \bot$.

Пример 1.2. *Посматрајмо формулу* $\Delta = (p \vee q) \to (p \wedge q)$. *Формула је задовољива јер избором интерпретације* $I_1(p) = \top$ *и* $I_1(q) = \top$, *важи* $I_1(\Delta) = \top$, *стога је* I_1 *модел за* Δ *и формула није контрадикција. Један контрамодел за* Δ *је интерпретација* $I_2(p) = \top$ *и* $I_2(q) = \bot$, *јер је* $I_2(\Delta) = \bot$, *те формула није таутологија. Пример таутологије би била формула* $(p \vee q) \leftrightarrow (q \vee p)$, *а пример контрадикције* $p \wedge \neg\, p$.

2 Теоријске основе

2.1 Аналитички таблои

Овај део ћемо посветити методи аналитичких метода у исказној логици. Главни циљ методе је да за дату формулу провери њену ваљаност. Коришћењем исте, настојимо да направимо контрамодел за неку формулу, па ако се у томе не успе, метода гарантује да је формула ваљана. Како тражимо контрамоделе? Формула је ваљана ако и само ако њена негација није тачна ни при једној интерпретацији, стога се природно намеће посматрање негације формуле, и тражење интерпретација у којима је она тачна. Сада ћемо увести и нотацију коју користимо у наставку поглавља(такозвана уједначавајућа нотација).

α	α_1	α_2
$TA \wedge B$	TA	TB
$FA \vee B$	FA	FB
$FA \to B$	FA	TB
$T\neg A$	FA	FA
$F\neg A$	TA	TA

β	β_1	β_2
$FA \wedge B$	FA	FB
$TA \vee B$	TA	TB
$TA \to B$	FA	TB

Табела 2.1: α и β формуле

За исказни језик који се састоји од пребројиво много исказних слова, заграда и логичких везника $\neg$, $\wedge$, $\vee$, $\to$ дефинишимо нове формалне симболе T и F (асоцирају нас на true, односно false) који се не налазе у исказном језику. За исказну формулу A, формуле TA и FA су *означене формуле*. *Конјугат* формуле TA је FA, и обрнуто. Знак T испред формуле тумачимо као "тачно је", док знак F тумачимо као "није тачно". Сходно томе, за произвољну интерпретацију I је $I(TA) = I(A)$ и $I(FA) = I(\neg A)$.

Све неатомске означене формуле(у случају класичне исказне логике, атомске формуле су исказна слова) се деле на α и β формуле. Такве формуле имају одговарајуће компоненте α_1 и α_2, односно β_1 и β_2 (Табела 2.1).

Није тешко уочити да се α формуле могу тумачити као конјункција својих компоненти. На пример, "тачно је"$A \wedge B$ ако и само ако је "тачно"A и ако је "тачно"B. Слично, β формуле тумачимо као дисјункцију одговарајућих компоненти. Ову константацију оправдава следећа теорема:

Теорема 2.1. *За произвољну интерпретацију I важи:*

- $I(\alpha) = \top$ *акко* $I(\alpha_1) = I(\alpha_2) = \top$
- $I(\beta) = \top$ *акко* $I(\beta_1) = \top$ *или* $I(\beta_2) = \top$
- $I(TA) = \top$ *акко* $I(FA) = \bot$*, за означену формулу* A

Доказ. У зависности од типа формуле лако се доказује тражено. На пример, $I(FA \to B) = \top \Leftrightarrow I(A \to B) = \bot \Leftrightarrow I(A) = \top$ и $I(B) = \bot \Leftrightarrow I(TA) = I(FB) = \top$. Аналогно доказујемо и за остале случајеве. □

Опишимо сада поступак формирања аналитичког таблоа. За формулу A, аналитички табло представља бинарно дрво у чијим чворовима се налазе означене подформуле формуле A које се конструише на следећи начин:

- У корен дрвета поставља се формула FA
- Нека је L лист[1] неке гране[2] θ до тог момента конструисаног дрвета. У зависности од формула које се налазе на грани θ, наставак конструкције се одвија у два могућа смера:

  α *правило*: Ако се α формула налази на θ, тада се грана продужава са два нова чвора у којима се налазе формуле α_1 и α_2 редом. (Потомак, односно дете листа L је чвор који садржи α_1, а дете чвора α_1 је чвор који садржи формулу α_2).

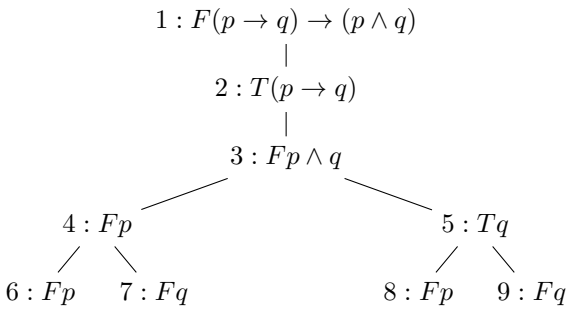
  β *правило*: Ако се β формула налази на θ, тада се θ у чвору L грана, при чему леви потомак садржи формулу β_1, а десни потомак формулу β_2.

Важно је напоменути да ако кроз неки чвор пролази више грана, примена правила конструкције на тај чвор се одражава на све гране(*). Преостаје нам да видимо када и како се завршава конструкција таблоа. Стога, дефинишимо пар појмова које ћемо користити у наставку и који нам директно из дефиниције дају одговор на оно што тражимо:

- Грана таблоа је *затворена* ако се на њој налазе формула и њен конјугат. Табло је *затворен* ако је свака грана таблоа затворена.

[1] *лист* је чвор дрвета који нема ниједног потомка

[2] *грана* представља пут у дрвету који почиње у корену, а зарвшава се у неком листу



Слика 1

- Грана таблоа је *завршена* ако су одговарајућа правила конструкције примењена на све чворове са те гране.
- Табло је *завршен* ако му је свака грана затворена или завршена.
- Грана је *задовољива* ако постоји интерпретација I при којој су све означене формуле у чворовима те гране тачне. Табло је *задовољив* ако постоји грана која је задовољива.

Пример 2.1. *Конструишимо табло за формулу $\Delta = (p \to q) \to (p \wedge q)$(Слика 1). Сада ћемо детаљно описати поступак конструкције. Иницијално, у чвор 1 постављамо означену формулу $F\Delta$. Применом α правила формирају се чворови 2 и 3. Даље, применом β правила на чвор 2, формирају се човорови 4 и 5. Коначно, применом β правила на чвор 3, формирамо чворове 6 и 7, односно 8 и 9 (сетимо се (*), правило се примењује на све гране које пролазе кроз чвор, у нашем случају гране $1-2-3-4$ и $1-2-3-5$), чиме је табло завршен. Грана $1-2-3-5-9$ је затворена (Конјугати Tq и Fq се налазе на њој), али преостале три гране нису, стога табло није затворен.*

Право смисао затвореног таблоа наћи ћемо у наредном одељку, када ће се испоставити да је формула Δ ваљана ако и само ако је табло у чијем корену је $F\Delta$ затворен. Као што смо рекли на почетку, ова метода настоји да генерише контрамоделе. Неуспех у конструкцији контрамодела се огледа у затварању једне гране с обзиром на то да скуп формула са те гране не може бити задовољив (на грани се налазе означена формула и њен конјугат). Поред овога, постоје питања која се природно намећу (на пример, да ли се овом методом генеришу сви контрамодели?) на која је углавном лако одговорити, али су свакако вредна дискусије и на њих ћемо одговорити у наредном одељку, где ћемо и формализовати претходне константације.

2.2 Коректност и комплетност

Посматрамо бинарно дрво T_1, у чијим чворовима се налазе означене формуле. Тада за бинарно дрво T_2 кажемо да је његово *директно проширење* ако се T_2 добија из T_1 применом α или β правила. Такође, *доказ* за неозначену формулу A је затворен табло за A.

Теорема 2.2. *Ако је T_1 задовољив табло, и ако је T_2 његово директно проширење, тада је и табло T_2 задовољив.*

Доказ. Уочимо грану θ која је задовољива у таблоу T_1 при некој интерпретацији I. Ако је T_2 добијен правилом које не мења θ, онда је очигледно и T_2 задовољив. У случају да је примењено α правило, на основу теореме 2.1 имамо да ће новодобијена грана и табло T_2 бити задовољиви. Слично и за β правило, при чему ће бар једна од новодобијених грана (проширења од θ), а тиме и табло T_2, бити задовољиви. □

Теорема 2.3. ***(Коректност)*** *Ако формула A има табло доказ, онда је она ваљана.*

Доказ. Претпоставимо да формула A има контрамодел. Тада постоји интерпретација I, при чему је $I(FA) = \top$. То би значило на основу теореме 2.2 да постоји грана која је задовољива у табло доказу. Међутим, то је немогуће јер је свака његова грана затворена, те не може бити задовољива. Одавде следи да је A ваљана. □

Очигледно затворена грана таблоа не може бити задовољива, па нам једино преостаје "потрага" за контрамоделима у завршеним (незатвореним) гранама. Очекиван одговор наводи следећа теорема:

Теорема 2.4. *Свака завршена грана која није затворена је задовољива.*

Доказ. Нека је θ завршена грана и Ω_θ скуп означених формула са те гране. За свако исказно слово p, највише једна од формула Tp и Fp се налази на θ. Ако α формула припада Ω_θ, онда компоненте α_1 и α_2 придају Ω_θ. Ако β формула припада Ω_θ, онда β_1 или β_2 припадају Ω_θ (**). Сада тражимо интерпретацију при којој ће θ бити задовољива. Како током формирања таблоа, у сваком кораку упрошћавамо формуле, завршене гране ће за чланове имати и формуле облика Tp или Fp, где је p исказно слово. Стога је природно дефинисати интерпретацију при којој је $I_\theta(p) = \top$ ако и само ако $Tp \in \Omega_\theta$ за свако исказно слово p. Индукцијом по сложености формуле лако се доказује да I_θ задовољава све формуле из Ω_θ. Наиме, доказ за исказна слова је тривијалан и следи из дефиниције I_θ. На основу (**) и 2.1, ако $\alpha \in \Omega_\theta$ тада $\alpha_1, \alpha_2 \in \Omega_\theta$, а на основу индуктивне хипотезе $I_\theta(\alpha_1) = I_\theta(\alpha_2) = \top$, односно $I_\theta(\alpha) = \top$ за неку α формулу. Слично следи и за β формулу, чиме се комплетира наш доказ. □

Теорема 2.5. ***(Комплетност)*** *Ако је формула A ваљана, сваки завршени табло за A мора бити затворен.*

Доказ. Претпоставимо супротно, односно нека је табло завршен, али не и затворен. Тада постоји грана која је завршена, али не и затворена. На основу претходне теореме, постоји интерпретација при којој је она задовољива, односно при којој је $I(FA) = \top$. Међутим, како је A таутологија, долазимо до контрадикције.

□

У теореми 2.4 ми смо за дату завршену грану дефинисали интерпретацију I_θ чиме смо обезбедили задовољивост гране. Међутим, како постоје исказна слова p чије означене формуле Fp и Tp се не налазе на грани (ниједна од њих), и како вредности тих слова при некој интерпретацији I не утичу на задовољивост гране при I (уз услов да је $I(a) = \top$ ако Ta припада грани, и $I(a) = \bot$ ако Fa припада грани, за исказно слово $a \in \Omega_\theta$), постоји више интерпретација, поред I_θ, при којима је грана задовољива. Прецизније, за свако исказно слово које не припада Ω_θ, постоје две могућности за вредност при интерпретацији, стога број оваквих интерпретација експоненцијално зависи баш од броја овако изабраних исказних слова.

Да ли смо овим поступком "покрили" све контрамоделе? Одговор је ДА. Наиме, сваки контрамодел одређује интерпретацију I за коју је $I(FA) = \bot$ (за неозначену формулу A). На основу доказа теореме 2.2, затворен табло за A ће бити задовољив, шта више постојаће грана која је задовољива при I, а на основу основу претходног начина "генерисања" контрамодела на једној грани, обезбедили смо постојање интерпретације I.

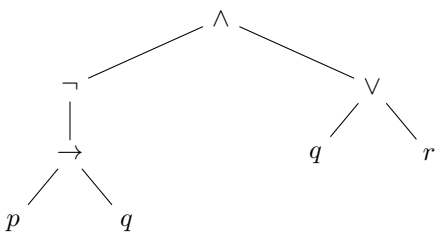
3 Програмска реализација

У овом делу се прво преводи унета ниска карактера, која треба да представаља формулу класичне исказне логике, у бинарно дрво, а затим се од тако учитане формуле формира табло и врши се претрага његових грана.

Претварањем унете формуле (у виду ниске карактера) у бинарно дрво добија се једноставнији облик формуле за чување и читање и омогућава лакше формирање одговарајућег таблоа. Паралелно томе врши се и провера исправности самог уноса због природе прављења дрвета. То се ради на следећи начин. Ако имамо формулу класичне исказне логике облика $\alpha * \beta$, где је $*$ логички бинарни симбол, $*$ се поставља у корен дрвета, у лево подстабло се ставља одговарајуће стабло формуле α, а у десно подстабло одговарајуће стабло формуле β. Уколико је формула облика $\neg\alpha$, додаје се један чвор који садржи негацију и на њега се надовезује други који садржи формулу α. Поступак се понавља док не дођемо до исказних слова.

Тако је на пример за формулу $\neg(p \to q) \wedge (q \vee r)$ одговарајуће дрво приказано на слици 2.

То је имплементирано на следећи начин. За сваки бинарни симбол формуле се проверава да ли су у левој и десној подформули добро постављене заграде (да ли је свака спарена са одговарајућом) и уколико јесте, уклањају се спољне заграде и врши се рекурзивна провера за обе



Слика 2

подформуле. Поступак се понавља док се не дође до исказног слова. Уколико се у неком тренутку примети неправилно спаривање заграда, нелегални бинарни симбол или нелегално исказно слово, извршавање се прекида и формула се проглашава неисправном. Детаљнији опис дат је наредним псеудокодом 1.

Алгоритам 1 Исправност уноса и формирање дрвета

Улаз: Ниска карактера s

Излаз: Провера да ли је унета ниска формула класичне исказне логике и у потврдном случају се формира одговарајуће дрво t

```
1: избацивање бланко знакова из ниске s
2: додавање отворене и затворене заграде на крајеве ниске s

3: функција formirajDrvo(s)
4:     уклони спољне заграде у s ако постоје
5:     ако s се састоји из само једног симбола онда
6:         ако је тај симбол мало или велико латинично слово онда
7:             врати чвор који садржи то слово
8:         иначе
9:             врати NULL
10:    за све бинарне симболе b у ниски s ради
11:        ако лева и десна подниска имају спарене заграде онда
12:            l ← formirajDrvo(лева подниска)
13:            d ← formirajDrvo(десна подниска)
14:            ако l и d нису NULL и s је обргрљен заградама онда
15:                c ← нови чвор који садржи b
16:                c показује лево на l
17:                c показује десно на d
18:                врати c
19:    ако s почиње негацијом онда
20:        избаци негацију из s
21:        t ←formirajDrvo(s)
22:        ако t није NULL онда
23:            врати чвор са негацијом који показује на t
24:    врати NULL

25: врати formirajDrvo(s)          ▷ враћа одговарајуће стабло ако је фомула коректна
```

Овде је t показивач на корен дрвета, и дрво је реализовано тако да сваки чвор садржи, поред одговарајућег симбола, и показиваче на леву и десну грану. Ради лакшег формирања таблоа, претходном алгоритму се додаје да разликује случај еквиваленције коју одмах раздваја на две импликације.

Временска сложеност овог алгоритма је $O(n^2 \log n)$, а просторна је $\log n$, где је n дужина ниске s.

Наредни део се састоји од прављења таблоа за овако формирано дрво. Креће се од корена

дрвета и у зависности од симбола у њему примењује се α или β правило. Тако се у случају α правила додају по две нове гране чвору са одговарајућим формулама и симболима, док се за β правило дода једна грана са два чвора. Онда се проналазе сви листови дрвета и за сваки од њих се поступак понавља.

Да би се поступак убрзао, а и да би се уједно проверило да ли је формула таутологија, може се већ током формирања таблоа вршити провера да ли је нека грана затворена или не. Једна од имплементација такве провере може бити тако што се у току проласка кроз табло издваја посебан низ у меморији који ће чувати податке о броју појављивања исказних слова или њихових негација. Кроз табло се пролази претрагом у дубину и када се током претраге дође до исказног слова, повећа се одговарајући елемент низа који чува број појављивања, док се током рекурзивног враћања тај број смањује. На тај начин се обезбеђује да се посматрају понављања исказних слова само са исте гране. Уколико у неком тренутку дође до појављивања истог исказног слова и са F и са T симболом (што видимо преко помоћног низа), онда се чворови те гране означавају затвореним и неће се даље проверавати, чиме се добија уштеда времена. Поступак се наставља све док нема више незатворених чворова или док су сва правила формирања таблоа примењена. Уколико су сви чворови затворени, формула јесте таутологија и то се исписује, док су у негативном случају од интереса отворене гране које се касније посматрају за исписивање контрамодела.

Овај поступак се може описати и алгоритмом 2.

Алгоритам 2 Формирање таблоа

Улаз: Формула A класичне исказне логике у форми дрвета
Излаз: Аналитички табло формуле A

```
1:  функција proveriTablo(t)
2:      ако t показује на симбол s онда
3:          повећај појављивање симбола s са одговарајућом ознаком из t
4:          ако већ постоји појављивање симбола s са супротном ознаком онда
5:              означи t као затворен
6:      proveriTablo(лева грана од t)
7:      смањи појављивање симбола са одговарајућим знаком из леве гране од t
8:      proveriTablo(десна грана од t)
9:      смањи појављивање симбола са одговарајућим знаком из десне гране од t
10:     ако су чворови из обе гране чвора t затворени онда
11:         означи t као затворен

12: креирај чвор t који садржи симбол F и корен фомуле A
13: док постоји непосећен чвор c ради
14:     узети за c онај незатворени чвор најмање дубине у дрвету      ▷ c је корен незатворених
15:     за све незатворене листове l дрвета t ради
16:         у зависности од симбола у c применити α или β правило на l
17:         proveriTablo(t)
18: врати t
```

На овај начин провера да ли је формула таутологија своди се само на упит да ли је корен таблоа t затворен или не. Уколико јесте, онда је и цео табло затворен и формула је таутологија. Уколико није, постоје контрамодели формуле и они могу да се добију проласком кроз незатворене гране таблоа.

Проналажење контрамодела се одвија на следећи начин. Како је у интересу да се испитују гране појединачно, претрага се врши у дубину и током ње се у меморији чува низ који памти тренутан број појављивања исказних слова са одговарајућим знаком (F или T) у таблоу. Када се дође до листа таблоа, на основу тог помоћног низа може се установити коју истинитосну вредност треба да добију одређени симболи. Уколико се у некој грани не појаве сва исказна слова која учествују у формули, тада њихова истинитосна вредност није одређена кроз пролазак и може да буде произвољна. Зато се за те преостале симболе генеришу све варијације могућих вредности. Детаљнији опис дат је алгоримом 3.

Алгоритам 3 Генерисање контрамодела

Улаз: Табло t формуле A класичне исказне логике
Излаз: Контрамодели формуле A

```
1: функција KONTRAMODELI(t)
2:     ако је t исказно слово онда
3:         повећај број појављивања симбола са одговарајућим знаком из t
4:     ако је t лист онда
5:         додели истинитосне вредности за исказна слова која се појављују у грани
6:         генериши све могуће варијације за преостала исказна слова
7:         испиши те контрамоделе
8:     иначе
9:         KONTRAMODELI(лева грана од t)
10:        смањи појављивање симбола са одговарајућим знаком из леве гране од t
11:        KONTRAMODELI(десна грана од t)
12:        смањи појављивање симбола са одговарајућим знаком из десне гране од t
```

Композицијом ових функција за полазну унету ниску карактера која треба да представља формулу класичне исказне логике испитује се да ли је ваљана или не, а у негативном случају се одређују и сви њени контрамодели.

4 Резултати

Како сложеност алгоритама 2 и 3 не зависи само од дужине унете ниске, већ и од броја исказних слова која учествују у формули, као и од саме конструкције формуле (који симболи су у њој и на који начин су комбиновани), није баш најједноставије прецизно је одредити.

За већину формула са мање од 15 карактера (не рачунајући белине), извршавање траје неколико десетина секунди, док за оне мало дуже, ситуација може бити доста сложенија.

На пример, за формулу $a \vee b \vee c \vee d \vee e \vee f \vee g$ извршавање програма траје 0.79 секунди, док за формулу $a \leftrightarrow b \leftrightarrow c \leftrightarrow d \leftrightarrow e \leftrightarrow f \leftrightarrow g$ извршавање траје 11.86 секунди.

За формулу, од око 75 карактера,

$$\neg(((p \vee (q \wedge r)) \leftrightarrow ((p \vee q) \wedge (p \vee r))) \wedge (((p \rightarrow r) \wedge (q \rightarrow r)) \leftrightarrow ((p \vee q) \rightarrow r))) \vee ((p \rightarrow q) \rightarrow ((p \vee r) \rightarrow (q \vee r)))$$

која је таутологија, је потребно 7.82 секунде.

Уколико последње исказно слово r заменимо неким новим, на пример a, за новодобијену формулу, са истим бројем карактера као претходна,

$$\neg(((p \vee (q \wedge r)) \leftrightarrow ((p \vee q) \wedge (p \vee r))) \wedge (((p \rightarrow r) \wedge (q \rightarrow r)) \leftrightarrow ((p \vee q) \rightarrow r))) \vee ((p \rightarrow q) \rightarrow ((p \vee r) \rightarrow (q \vee a)))$$

која овога пута није таутологија, је потребно 54.36 секунди.

Програм на крају генерише и LaTeX документ са скицом таблоа и начином на који су гране добијене. Пример једног целокупног поступка може се видети на формули $(p \vee q) \rightarrow p$, где се на неколико десетина секунде добија табло, да формула није таутологија и њени контрамодели. Генерисани документ изгледа овако:

(1) $F\ (p \vee q) \Rightarrow p$ $\quad |\boldsymbol{\alpha}:\ \mathbf{(2,\ 3)}|$

(2) $T\ p \vee q$ $\quad |\boldsymbol{\beta}:\ \mathbf{(4,\ 5)}|$

(3) $F\ p$

(4) $T\ p$ (5) $T\ q$

Овде је једна грана затворена, а друга није, па програм враћа да формула није таутологија и исписује контрамоделе.

```
I(p) = 0; I(q) = 1;
```

Дакле, формула није тачна у само једном случају и то је за нетачно p и тачно q.

5 Закључак

Резултати показују да је овако имплементирана метода аналитичих таблоа применљива за краће формуле или оне мало једноставније, док извршавње може да потраје за оне компликованије.

Ово се може потеницијално решити тако што се код претраге за затвореност таблоа посматрају целе формуле и њихове негације уместо само исказних слова. У тренутној имплементацији, грану таблоа обележавамо затвореном само ако приметимо да се на њој појављују чворови облика p и $\neg p$, за неко исказно слово p, а она се може прогласити затвореном и раније, уколико се јави нека формула α на грани заједно са својом негацијом $\neg\alpha$. Овакав додатак би значајно допринео временској сложености.

Још унапређења се може додати и на почетку, при учитавању. Тренутно програм за исказна слова може да учита само мала и велика латинична слова абецеде, па се у једној формули може јавити највише 52 различита исказна слова. Могло би се додати могућност учитавања исказних слова са индексом нпр. a_1 као a1. Ово не би побољшало време извршавања, али би омогућило већу слободу при избору формула, односно већи домен уноса.

Такође, код слободних исказних слова у генерисању контрамодела, током рачунања свих могућих варијација у неким ситуацијама долази до понављања контрамодела. На пример, ако имамо формулу $p \wedge q$, долази до двоструког исписивања контрамодела у коме су и p и q нетачни. Ово се јавља јер се у листовима таблоа јављају $F\ p$ и $F\ q$, и онда се за случај када је p нетачно исписују сви могући случајеви за q, од којих је један нетачно p и нетачно q. Међутим, тај случај се добија и када се пролази кроз другу грану, где је q нетачно и када се генеришу све могућности за p. Као ни претходно, ни ово унапређење не би значајно допринело временској или просторној сложености, али би омогућило краћи и компактнији испис.

Литература

[1] Ненад Крцавац Зоран Огњановић. *Увод у теоријско рачунарство.* 2004.