Др Предраг Цветковић,[*]
Редовни професор,
Правни факултет, Универзитет у Нишу

ОРИГИНАЛНИ НАУЧНИ РАД
10.5937/zrpfn0-34462

UDK: 339:004.658.2
004.421
Рад примљен: 15.10.2021.
Рад прихваћен: 17.11.2021.

УГОВОР КАО АЛГОРИТАМ: УВОДНА РАЗМАТРАЊА[**]

Апстракт**:** *Рад се бави могућношћу претварања уговора у форми текста у алгоритамски облик. „Алгоритмизација уговора" требало би да омогући конверзију уговорних одредби у алгоритам у циљу њиховог коришћења код тзв. „дигитално унапређених уговора" попут паметних уговора (е. Smart contracts), а да се при томе обезбеди правна валидност и ефикасност уговорних одредби. Алгоритам је план решавања одређеног проблема кроз предузимање појединачних корака. У правном контексту, алгоритам је посебан начин управљања понашањем уговорних страна. Описано управљање спроводи се одабиром информација о понашању страна, те контролом и усмеравањем спровођења уговора у складу са тим понашањем. Увођење алгоритма као форме дефинисања уговорних одредби утиче на формат и суштину уговорног права. Алгоритам уговора може бити у облику псеудокода, симбола формалне логике и дијаграма тока. Разумевању алгоритмизације уговора у облику псеудокода од помоћи су механизми и правила формалне логике. Ова правила омогућавају систематско „уситњавање" правног текста кроз логичку матрицу, чиме се олакшава процес алгоритмизације кроз кораке. Развој принципа за алгоритмирање уговора процес је који траје. Кључна питања која траже одговор су, између осталог, основ и садржина нормативности уговора у форми кода и њихова подобност да буду квалификовани као правно ваљани формат.*

Кључне речи*: уговор, алгоритам, паметни уговори, псеудокод, формална логика.*

---

[*] pepi@prafak.ni.ac.rs

[**] Рад је представљен на међународној научној конференцији „Право и дигитализација" одржаној на Правном факултету Универзитета у Нишу, априла 2021. године.

1. Увод

Правне норме садржане у уговору (такође у статутима, по законима) и написане природним језиком могу бити предмет алгоритамске конверзије у одређеним фазама трајања уговора (спровођење, праћење, контрола, тумачење). Коришћењем блокчејн концепта као структурног обрасца[1] отвара се могућност стварања уговора са аутоматизованим извршењем: „паметног" уговора[2] (Цветковић, 2020: 127–144; о паметном уговору видети

1 *Blockchain* је сложеница речи *Block* (блок) и *chain* (ланац). Ради се о концепту заснованом на коришћењу криптографски заштићеног ланца трансакционих блокова. Трансакције се пакују у блокове, а блокови се везују у ланац. Блокови се везују криптографски, кроз Хеш (е. *Hash*) функцију: садржај блока не може да се промени а да се не измени садржај свих других блокова који му претходе. Дакле, блокчејн је датотека која информације складишти у блокове. Сваки је блок везан за следећи блок коришћењем криптографске сигнатуре. Ово омогућава да блокчејнови буду коришћени као деловодна књига која може да се дели (е. *share*) и потврђује од стране сваког са одговарајућом дозволом да то чини. Концепт верификације дигиталних података праћењем кроз блокове идентичан је деловној књизи: блокови функционишу као књиговодствени улошци дигиталног деловодника. Блокчејн омогућава складиштење и дељења информација кроз блокове у *peer to peer* мрежи (мрежи кроз коју учесници комуницирају без посредника). Идентичне копије блокова (који су функционално књиговодствени улошци) заједнички верификују чланови мреже. Верификована информација је садржана у блоковима који су додати у хронолошком ланцу постојећих и одобрених блокова коришћењем криптографске сигнатуре. Сваки нови блок има временски печат који кореспондира уносу нових информација (података): такав нови блок садржи информације о претходном блоку тако да сваки покушај измене једног блока захтева измену сваког раније евидентираног блока (који и сам садржи податке о блоковима који су му претходили). Суштински, сваки блок има учитане податке о свим претходним блоковима (трансакцијама) унутар једног блокчејна. Значај технологије блокчејна је што осигурава аутентичност дигиталних података: поверење у класичном правном односу замењено је верификацијом кроз податке у блоковима на горе наведени начин. Блокчејн концепт је транспарентан и омогућава ефикасну (брзу и јефтину) трансмисију информација у информатичким мрежама. Блокчејн ће за трансакције бити оно што је интернет за комуникацију: оно што је започело као средство поделе информација трансформисаће индустрију једном када се његова примена прошири.

2 На другом полу од „паметног" („аналогног") стоји традиционални (даље и „дигитални" уговор). Концизна дефиниција паметног уговора је следећа: паметни уговор је уговор повезан са компјутерским протоколом, написан рачунарским програмским језиком, који аутоматски извршава програмиране функције као одговор на испуњење одређених услова. Описани концепт није нов: ипак, интегрисан са блокчејн технологијом гради потенцијал паметних уговора да аутоматизују и гарантују извршење великог броја различитих уговорних обавеза без потребе постојања централног ауторитета, правног система или спољашњег механизма спровођења. У овим случајевима, паметни уговори доносе јасноћу, предвидљивост, могућност контроле/ревизије/ и олакшавају извршење уговорних обавеза уз смањење ризика који су повезани са људским учешћем.

више у Цветковић, 2021: 61–71). Алгоритмирање се у контексту овог рада схвата као процес који омогућава да се текст уговора преведе у формат који је разумљив за програмере који развијају програмски код који би требало да омогући извршавање уговора: са тим разумевањем могло би да се приступи програмирању уговора у форми кода. У том циљу предлаже се коришћење следећих методологија: псеудокода, формалне логике и дијаграма тока (е. *flowchart*). [3] Алгоритам би требало да објасни логичку структуру обавеза и услова одговорности страна на начин који обезбеђује валидност и ефективност уговора који је предмет кодирања. [4]

2.1. Тешкоће интегрисања права у код

Проблеми интегрисања права у код су вишеслојни.

Прво, програмери не познају право.

Друго, разликује се динамика анализе програмера и правника. Програмер примењује *a priori* знање о програмским језицима и алгоритмима: ово је знање објективизирано програмским језиком независним од искуства. Синтакса и семантика програмских језика је стриктна и не дозвољава слободу формулација. [5] Правна норма се пак фокусира на интеракцију социјалних елемената посматрану из правне перспективе. Примењује се уз *a posteriori* знање (коришћење искуства, емпиријске доказе и резултате опсервација).

Правно програмирање подразумева интегрисање перспектива права и програмирања како би се обезбедили заједнички обрасци функционисања правног и програмских језика. Ово интегрисање омогућавају две чињенице.

Прво, заједнички именитељ правног и програмског језика јесте структура резоновања: и један и други језик базирани су на правилима. Програмски језик прати стриктна правила програмирања; језик правне прозе пак заснован је на језичким принципима природног језика у функцији формулисања правног правила.

---

3 О дијаграму тока видети више *supra* у тач. 4. 3.

4 Алгоритмизација се у овом раду анализира кроз визуру претварања уговора у програмски код као de lege lata процеса који је видљив и подобан за праћење и истраживање. То не искључује могућност да у блиској будућности буде могуће кодирање прописа и закона. Видети пример Новог Зеланда у Barraclough, Fraser, Barnes, 2021: 1–175.

5 Програмирање, поред познавања програмског језика у питању, укључује примену логике, коришћење математике и друге методе које иду изван простог учења програмског језика.

Друго, сличност правног правила и програмирања је што њихов процес настанка има дефинисану структуру и правила контроле. Програмер такође процесуира одређене податке према правилима програмског кода, поступајући у складу са задатим правилима структуре и формата (*Koch*, 2005: 238).

2.2. Трансформација уговора из прозе у алгоритам: техничка питања

Правни документи пишу се на природном језику. Компјутери не разумеју овај језик, чак и када тај језик користи концизне сентенце писане у структуираној правној прози. Природан језик је експресиван али инхерентно двосмислен и неодређен. Са друге стране, програмски језици имају за циљ управо елиминисање неодређености.

Конвертовање правила из уговора на „природном језику" (е. *Natural Language*) у програмски код захтева да се превазиђе разлика између ова два начина изражавања. Технике „конвертовања природног језика" (е. *Natural Languague Processing*) ослањају се на статистичке и стохастичке анализе а не на установљавање механизама за одређивање смисла. Илустрације ради: у програмском језику, реч „испорука" посматра се само као низ симбола (е. *string*), а не назив за обавезу продавца да преда робу купцу. Илустрације ради, реч „испорука" је у програмском језику објекат који захтева формализоване инструкције и дефинисање секвенци поступања усмерених ка предаји робе од продавца купцу. Рачунар анализира правну прозу (не чита га у контексту писаног текста) тако што је дели, детектује обрасце и дефинише статистичке податке о анализираном тексту (*Ashley*, 2017: 132–135).

За алгоритмирање уговора важно је разумети циљ кодирања (коме претходи алгоритмизација): код би требало да буде функционални еквивалент правног правила које је предмет интегрисања.

Интеграција уговора и кода одвија се у следећим фазама:

фаза 1: алгоритмизација уговора написаног на природном језику (путем псеудокода/формалне логике/дијаграма тока):

фаза 2: конверзија алгоритамског израза у изворни код;[6]

6 У рачунарству, изворни код (енгл. *Source code*) је колекција компјутерских инструкција написана на неком, за људе разумљивом програмском језику. Изворни код програма је специјално дизајниран за рад програмера који прецизира акције које ће обављати са компјутера писањем кодова. Изворни код је често трансформисан од стране програмског преводиоца (видети *infra* напомену 8) у код ниског нивоа који је разумљив

фаза 3: конверзија изворног кода у компилатор (преводилац програма)[7]

фаза 4: конверзија компилатора у машински код[8] (видети: Јовановић, 2005: 51–91).

Пут од реченице правне прозе до инструкције извршивог машинског кода је дуг: свака фаза интеграције права у код значи и удаљавање од природног језика, мању разумљивост за уговараче и самим тим мање контроле над текстом уговора. Описани процес носи ризике у свакој фази са којима се има рачунати приликом трансформације уговора у програмски код. [9]

2.3. Систем правне анализе (е. Legal Expert System).

Успешно претварање правне прозе у код захтева сталну, холистичку и ефикасну сарадњу између правника и програмера. Интеграција уговорне одредбе у програмски код захтева интерпретацију правних правила: тумачи правних правила су правници. Циљ тумачења је да код буде дефинисан на начин који је у складу са правним правилима. Оквир те сарадње је установљавање тзв. „Система правне експертизе". (е. *Legal Expert System*). У основи, овај систем генерише снагу „правног инжињеринга": две експертизе које имају за циљ да обезбеде успешну дводимензионалну анализу ангажованих питања (правну и програмерску). Синтагма ЛЕС означава програм који су конципирали правници и који имитира решавање проблема од стране правника (*Susskind*, 1986: 172). Циљ ЛЕС јесте: стицање знања програмера о уговору као правном институту; интеграција тог знања у ЛЕС; контрола и тестирање програмског кода кроз сарадњу програмера и правника. Експерти који учествују у описаним активностима

компјутеру. Изворни код може да буде слободно доступан корисницима: било ко може да преузме изворни код, да га модификује и да дистрибуира његову модификовану верзију у неограниченом броју копија. Тада се ради о отвореном изворном коду. Не постоје новчане надокнаде за лиценцу или било која друга ограничења. Детаљнија и технолошки разрађена дефиниција дата је на веб страници *Open Source Initiative*; https://opensource.org/osd. Приступљено 14. 05. 2020.

7 Компилатор или програмски преводилац (енгл. *Compiler*) је рачунарски програм (или низ програма) који трансформише код једног програмског језика у други програмски језик. Код који се преводи обично се зове изворни код, а код добијен трансформацијом машински код.

8 Машински језик или машински код (бинарни код, машинац) је систем инструкција и података које централни процесор у рачунару непосредно извршава. Машински језик је у одређену руку примитиван програмски језик: јавља се у бинарном облику, што значи да се користе само 2 елемента (0 и 1).

9 Детаљнији преглед овог ризика подразумева технолошко-правну анализу која би својим обимом и суштином била изван оквира овог рада.

називају се „инжењерима правног процеса". Они могу да буду правници и/или програмери; вероватније је да појам обухвата најмање два лица са различитом експертизом: примарном (право или програмирање) и секундарном (основна знања из права или програмирања).

3. Уговор као предмет алгоритмирања

Алгоритмирање уговора требало би да конвертује уговор из прозе у код, а да у исто време очува његову валидност и ефикасност. Тешкоћа у овом процесу је следећа: текст уговора има ограничени капацитет да буде презентован у алгоритму (и даље у програмског коду). Основно ограничење паметних уговора је смањена могућност да се комплексни правни стандарди и концепти конвертују у програмски код. Аналогно услужним машинама код којих се извршење ослања на математичку калкулацију (нпр. да ли је уплаћена довољна сума новца да би се испоручила роба), паметни уговори се ослањају на прецизну и унапред дефинисану логику извршења. Како у том контексту верификовати понашање које је у складу са стандардима попут „разумности", „најбољих напора" или других који се у традиционалним уговорима користе да би се обезбедила флексибилност? Њихово транслатовање у код или редуцирање у формулу алгоритма је тешко, уколико је уопште и могуће. Паметни уговори у овом тренутку не могу да решавају комерцијално комплексне сценарије: програмски код не може да обезбеди све могуће одговоре на сва могућа питања која се могу поставити у вези са одређеним пословним односом *(von Haller Grønbæk,* 2016). Стога је *de lege lata* примена паметних уговора као потпуне замене за традиционалне уговоре искључена.

Описани недостатак превазилази се прихватањем следеће концепције: уговор је испуњен онда када су извршене обавезе њиме предвиђене. Циљ алгоритмираног уговора је да обезбеди да се програмирањем аутоматизује извршење уговором предвиђених обавеза: циљ није анализа текста или интерпретација. Инструкције програмерима морају да буду прецизне и недвосмислене. Аутоматизује се извршење одређених задатака садржаних у уговорним обавезама (плаћање, испорука, слање обавештења о пријему итд.). Аутоматизација извршења праћена је гаранцијом да ће извршење у питању бити учињено управо како је уговором усаглашено. Перфектно извршење елиминише могућност кршења уговора и стога се у обавезу извршавања програмског кода стапа и питање решавања спорова и испуњења уговора (Werbach, Cornell, 2017: 352). Уговорни документ у форми кода више није отелотворење споразума већ алат за његово испуњење. За разлику од традиционалног уговора, који разликује текст уговора и његово

извршење, програмски код допушта да текст контракта у програмском језику буде истовремено и акција његовог испуњења. Тиме се потенцијално изједначава уговорно регулисање и испуњење уговора (самим постанком кода – нормирањем – оно што би требало да буде учињено као резултат уговора већ је извршено с обзиром на непроменљивост програмског кода и његову затвореност за интервенцију посредника). У стварности нема друге опције осим примене правила које представља норму у форми паметног уговора.

Стога се говори о валидности програмског кода паметног уговора у форми блокчејна: валидност програмског кода претпоставља да је код паметног уговора (за разлику од уговора у писаној форми, чија се валидност процењује како у моменту настанка, тако и у фази примене) та процена могућа искључиво у *ex ante* фази (у моменту настанка – дизајна кода). Дизајн програмског кода, дакле, има нормативни ефекат (*Goldoni*, 2015: 128).

3.1. Карактеристике уговорних одредби као објекта алгоритмирања

Потенцијални објект алгоритмирања су тзв. примарне инструкције усмерене на извршење карактеристичне престације уговора. То нису клаузуле које служе решавању питања као што су избор меродавног права уговора, решавање спорова и слично. [10]

Када се ради о карактеру уговорне одредбе која има карактер примарне инструкције, она мора да буде дељива у кораке (секвенце) и да буде описана у условима коначног, бинарног резултата. Пример клаузула подобних за алгоритмирање су рецимо одредбе о плаћању: оне могу да се алгоритмирају тако што се функционисање алгоритма веже за објективне информације из рачуноводственог/књиговодственог система: тиме се обезбеђује независни параметар испуњења обавезе плаћања.

Нису сви параметри текста уговора до краја рашчлањиви (ово је рашчлањивање важно јер омогућава да се параметри преведу у алгоритам и учитају у програмски код). Прво, одређена права и обавезе имају форму стандарда (разумни напори, разумно лице итд.). Друго, текст може да има празнине и да захтева имплицирање из постојећег текста да би се дефинисала воља страна; треће, уговор упућује и на контекст трансакције. Све ово захтева додатну интерпретацију.

10 Питање решавања спорова проистеклих из функционсања дигиталног окружења је предмет посебних академских анализа (видети. Kaulartz, 2019: 73–85).

Један од начина на који је могуће управљати овим неодређеностима кроз код је да се у самом тексту уговора ови стандарди преформулишу на начин који их чини подобним за алгоритмирање и касније кодирање. На пример, обавеза предузмања разумних напора може да се објективизира дефинисањем акција чије предузимање значи испуњење ове обавезе (коришћењем кључних индикатора извршења – е. *Key Performances Indicator*). Даље, обавеза извршења одређење радње „без непотребног одлагања" може да се замени оквирним датумом извршења. Тиме обавеза поступања према стандардима постаје обавеза чије је извршавање објективно мерљиво. Чини се да би свест актера уговора да постоји могућност његове алгоритмизације и кодирања могла да обезбеди повратни утицај и утиче на њихов „алгоритамски приступ" самом тексту уговора (да би ово била тенденција, алгоритмирање уговора мора да постане тржишно пожељно понашање чиме би се уговарачи мотивисали да поступају на напред наведени начин).

Циљ паметних уговора је да се аутоматизира извршење а не да се осигура синтаксичка или семантичка верност кодиране верзије тексту уговора. Перфектно транслатовање уговорне прозе у код је немогуће због разлике у природи програмског и природног језика. Али је стога могућа функционална конверзија уговорних клаузула у акцију алгоритма/кода: конвертује се уговор као форма засведочавања споразума у активан алгоритам који покреће компјутерске операције (Surden, 2012, 667). Уместо питања да ли линија кода одражава текстуално значење клаузуле, значајније је да ли извршење кода обезбеђује резултат који су стране дефинисале уговором.

Велики број спорова базираних на писаним документима има свој корен у пропусту креатора да јасно дефинишу њихово значење: некада је то пропуштање намерно (да би се обезбедила флексибилност функционисања норме у датом социјалном, политичком, економском или другом правно релевантном контексту). Садашњи ниво развоја вештачке интелигенције није у стању да све клаузуле уговора преведе на језик програмског кода. Могући одговор је примена тзв. Рикардијанског уговора: Рикардијански уговор кључне уговорне одредбе дефинише у форми програмског кода, док су комплексније одредбе које се не могу конвертовати у алгоритам садржане у додатним инструкцијама које су део Рикардијанског уговора – нпр. линкове који обезбеђују текст ових комплексних одредби или пун текст споразума (видети: Цветковић, 2021: 65–75).

4. Методе алгоритмизације уговора

Превазилажење јаза између уговорне прозе и кода могуће је уколико је уговор формулисан на начин који омогућава његово алгоритмирање. Могућност алгоритмирања отвара простор за представљање уговора у форми која је ближа програмском коду него текст уговора.

Алгоритам је процедура која се састоји од коначног сета недвосмислених правила (инструкција) која спецификују коначне секвенце акција које обезбеђују решење проблема. У констексту трансформације уговора у код, алгоритам може да буде потенцијални формат уговора који обезбеђује конвертовање уговорне прозе у симболе одређеног програмског језика: суштински би он био припрема текста уговора за конвертовање у програмски језик. Адекватан алгоритам којим је презентован текст уговора претпоставка је кодирања уговорних одредби на начин који обезбеђује њихову валидност и ефекат.

Алгоритам је план решавања одређеног проблема кроз предузимање појединачних корака: у контексту уговора, алгоритмизација је метод управљања понашањем уговорних страна. Описано управљање спроводи се одабиром информација о понашању страна, те контролом и усмеравањем спровођења уговора у складу са тим понашањем.

Три су методе алгоритмирања уговора.

Први је коришћење псеудокода.

Други је коришћење правила формалне логике.

Трећи је представљање уговора кроз дијаграм тока (е. „Flowchart“).

4.1. Алгоритмирање уговора коришћењем псеудокода

Програмски језици имају специфичну синтаксу која се користи како би програм исправно функционисано. Псеудокод није функционални програмски језик у смислу да омогућава извршење програма: реч је (посматрано из визуре програмера) о неформалном систему изражавања којим се изражавају идеје настале током процеса развоја алгоритма. Псеудокод користи изразе (углавном на енглеском језику као *lingua franca* нашег доба) који имају прихваћена значења у највећем броју програмских језика. Реч је о следећим изразима.

INPUT – израз који означава уношење одређеног податка у алгоритам;

OUTPUT – израз који означава резултат обраде унетог податка кроз функцију алгоритма;

WHILE – израз за петљу (е. loop) извршења функције која има услов на почетку; док је услов испуњен, алгоритам се понавља;

FOR – израз који означава услов чије се постојање проверава: уколико је испуњен, алгоритам активира одређену функцију;

REPEAT – UNTIL – израз за петљу која понавља функцију и проверава да ли је испуњен услов за њено окончање: када се то деси, функција престаје да се извршава;

IF – THEN – ELSE – израз за доношење одлуке о избору одређеног понашања; у зависности од избора, активира се одређена функција алгоритма.

Следе две илустрације коришћења псеудокода.

Први је алгоритмирање уговора који за предмет има клађење. У алгоритму овог уговора, *IF-THEN- ELSE* принцип се ставља у алгоритам (рашчлањује у кораке) на следећи начин:

Почетни баланс у уговору: узети новац од А
Нови баланс: узети новац од Б
Примити и ускладиштити резултат утакмице као сигнал за испуњење услова
Уколико (резултат = “Тим х је победио“) {Новац пренети на рачун А }
У сваком другом случају {Пренети новац на рачун Б }

Други пример је уговор о пружању услуга оптимизације претраживања на интернету (е. *Service level agreement*).[11] Код овог уговора, компанија А нуди услугу оптимизације претраживања. У ту сврху креирала је уговор у форми блокчејна, на основу кога нуди услуге оптимизације за 1000 евра. Друга компанија (Б) која купује ову услугу депонује захтевану суму (1000 евра) у уговор у циљу добијања услуге оптимизације за свој домен. Паметни

11 *Service Level Agreement* (или скраћено СЛА) је део уговора (мада може да буде и посебан уговор) у ком се дефинише врста и ниво услуге између понуђача услуге оптимизације претраживања клијента на интернет претраживачима. Као што и сам назив каже, у питању је **ниво услуге**. Овим уговором, купац се осигурава да ће имати могућност одржавања система, као и могућност додатне надоградње већ развијеног система.

уговор требало би да процени да ли ће А бити у стању да домен компаније Б буде међу три најпопуларнија резултата претраживања у одређеном временском року (нпр. до 15. децембра). Уколико је овај услов испуњен, компанија А ће добити депоновану суму. У супротном, сума ће се вратити компанији Б. Паметни уговор би могао да изгледа на следећи начин.

IF domain http://www. example. com/,

RANKS between 1 to 3,

USING SEARCH TERM example, IN SEARCH ENGINE http://www. google. rs/,

BY 2016-12-15,

THEN Term 1 IS MET, and

DEPOSIT is transferred to X.

OTHERWISE Term 1 NOT MET, and

DEPOSIT is returned to Y (*Lauslahti, Mattila, & Seppala,* 2017: 18-19).

Документ са описаним алгоритмом дат је у псеудокоду и сличан је уговору. И субјекат А и субјекат Б изразили су вољу да буду обавезани уговором: субјекат А тиме што је припремио документ са псеудокодом и послао га Б, а субјекат Б тиме што је депоновао унапред дефинисану суму.

4.2. Алгоритмирање коришћењем правила формалне логике

Логика спада у каталог методолошких инструмената сваког правника. Правници су одувек користили логику. Сама конструкција норме (диспозиција и санкција) погодује представљању у форми логичког исказа. Правници логику схватају интуитивно: ипак, њено коришћење за формално представљање (па и рашчлањивање) правног правила захтева додатна знања. Ова знања су тим пре неопходна што постоје правила која нису подобна за логичко представљање: празнине, двосмисленост или нејасноћа неретко су свестан избор креатора норме (па и уговора) како би се обезбедила маргина флексибилности функционисања уговора.

Структура уговора (корелација права и обавеза праћена одговорношћу за неизвршење) погодује представљању у логичким исказима.

Пионир у овом смислу је Лејман Ален (е. *Layman E. Allen*). Он је развио метод формулисања уговорних одредби (е. *Writings in signs*) симболима формалне логике (видети: Allen, 1956: 833). Ален користи формалну логику за експресију уговора употребом шест симбола:

импликација (*IFTTT*-е. "*If this, than that*": „ако је ово, онда је то");

коимпликација (коимпликација се дефинише као коњукција две посебне импликације: на пример, коимпликацијом се сматра коњукција две импликације: прве да „P" имплицира „Q" і друге да непостојање „P" имплицира непостојање „Q");

коњукција која повезује два исказа везником „и"

ексклузивна дисјункција (тврдња која је тачна уколико су први или други исказ тачни, али не и оба; „P" или „Q" али не и „P" и „Q");

инклузивна дисјункција (тврдња која је тачна уколико су обе премисе тачне или уколико је тачна само једна од њих: P или Q & P и Q), и

негација (Allen, 1956: 833–854).

Норма се претвара у алгоритам путем системског рашчлањивања (пулверизације) правног текста кроз кроз логичку матрицу чиме се омогућава процес алгоритмизације уговора кроз кораке. Однос примарних елемената уговора дефинише логички однос између њих коришћењем шест наведених оператера.

Ален је анализом силогизама правних текстова обезбедио основ за савремено истраживање о начинима на које се програмски језици могу користити за дефинисање правних докумената (пре свега уговора): тиме се омогућава раздвајање конститутивних елемената у правном документу и сведено дефинисање логичког односа између њих. Предност описане анализе је могућност детектовања и контрадикторности текста, као и различитих опција за интерпретацију норме (има ставова да је систем изражавања дефинисан правилима формалне логике прецизан и да допушта једноставно секвенцирање, али је за право недовољно експресиван (*Kuhn*, 2014: 121, 127).

Ево примера конвертовања текста у форми логичких симбола. Члан 2-714 Једнообразног трговачког законика Сједињених Америчких Држава предвиђа да када је купац прихватио робу, може да тражи надокнаду штете за губитак који је настао због било које несагласности испуњења, уколико је та штета настала током редовног тока ствари као последица кршења обавезе од стране продавца, под условом да га је купац о томе

обавестио. Наведена одредба се конвертује у алгоритам кроз употребу симбола формалне логике на следећи начин:

∀р∀к∀п Ш(Рр & Кк & Пп & Арк & Окп &-Ср) → Нкп

где је

Р=роба

П=продавац

С=сагласност испоручене робе са условима уговора

К=купац

А=акцептирање робе од стране купца

О=обавештавање продавца од стране купца о несагласности робе условима уговора

Д=надокнада штете продавца купцу због несагласности робе са условима уговора (Видети: Lipshaw, 2018: 34–38)

Корак у смислу примене логичких симбола на дефинисање правних докумената је тзв. Декларативно логичко програмирање.

Логички програмски језици су категорија декларативних програмских језика. Оно што разликује логичко програмирање од других декларативних програма је да је основна јединица програмирања логичко правило, а не линија кода дата у форми програмског језика. [12] Пример за програмски језик заснован на декларативном логичком програмирању је Пролог. Реч је о језику развијеном седамдесетих година прошлог века (Van Em-

12 Декларативно програмирање дефинише се кроз разлике према императивном програмирању. Императивно програмирање утврђује кораке које би програм требало да изврши ради остварења одређеног циља, те да објасни на који начин сваки корак мења стање података којим програм манипулише. Декларативни програмски језици, уместо спецификовања инструкција за извршење, дефинишу циљ који би програмом требало да се изврши. За разлику од императивног програмирања, декларативни језик дозвољава рачунару да дефинише инструкције у форми сопственог императивног програма за извршење одређене акције (императивног програма који зависи од проблема који се решава, а није унапред дат као код императивних програмских језика). Инструкције код декларативног програмирања не морају да се прате у одређеном редоследу. Битно је да имају жељени резултат: компјутер генерише сопствену императивну процедуру. Декларативно програмирање се, дакле, може разумети као симплификовани начин програмирања на језику који утврђује циљеве, не процесе.

den, Kowalski, 1976: 733–742). Врло брзо је добио примену у кодирању правног резоновања (пример је кодирање Закона о држављанству Велике Британије (Sergot, Sadri, Kowalski, R Kriwaczek, Hammond, & Cory, 1986: 370–386). Декларативно логичко програмирање омогућава да се уговорна норма преформулише на начин који је разумљив правницима. То би значило да би правник могао да чита и разуме (уз знање формалне логике и језика заснованог на симболима формалне логике, попут Пролога) линије алгоритма на начин који му омогућава да процени да ли је алгоритам израз воље уговарача. Тиме се ствара простор за међукорак у коме се стварају јасни обрасци односа између уговора писаног на природном језику („аналогног“) и уговора у форми алгоритма (Ashley, 2017: 63).

4.3. Коришћење дијаграма тока (e. Flowchart) као метода програмирања

Метод дијаграма тока подразумева да се алгоритам презентује у геометријским облицима повезаном стрелицама. Дијаграм тока је метод представљања процеса у контексту програмирања (Brookshear, 2021: 267–268).

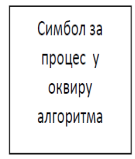
Дијаграм тока састоји се од геометријских облика (симбола) повезаних стрелицама.

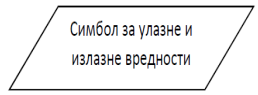
Сваки облик представља корак у процесу, а стрелице редослед тих корака.

Дијаграм тока комбиновањем симбола фигуративно приказује функционисање алгоритма. [13]

Постоји 6 базичних симбола дијаграма тока (унутар симбола уписано је њихово значење):

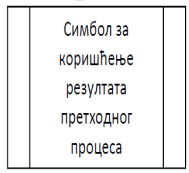
13 Данас је овај метод превазиђен. Користи се углавном за презентацију програма али сасвим ретко као метод за алгоритмирање.

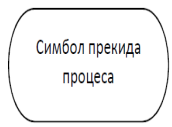
1. 

2. 

3. 

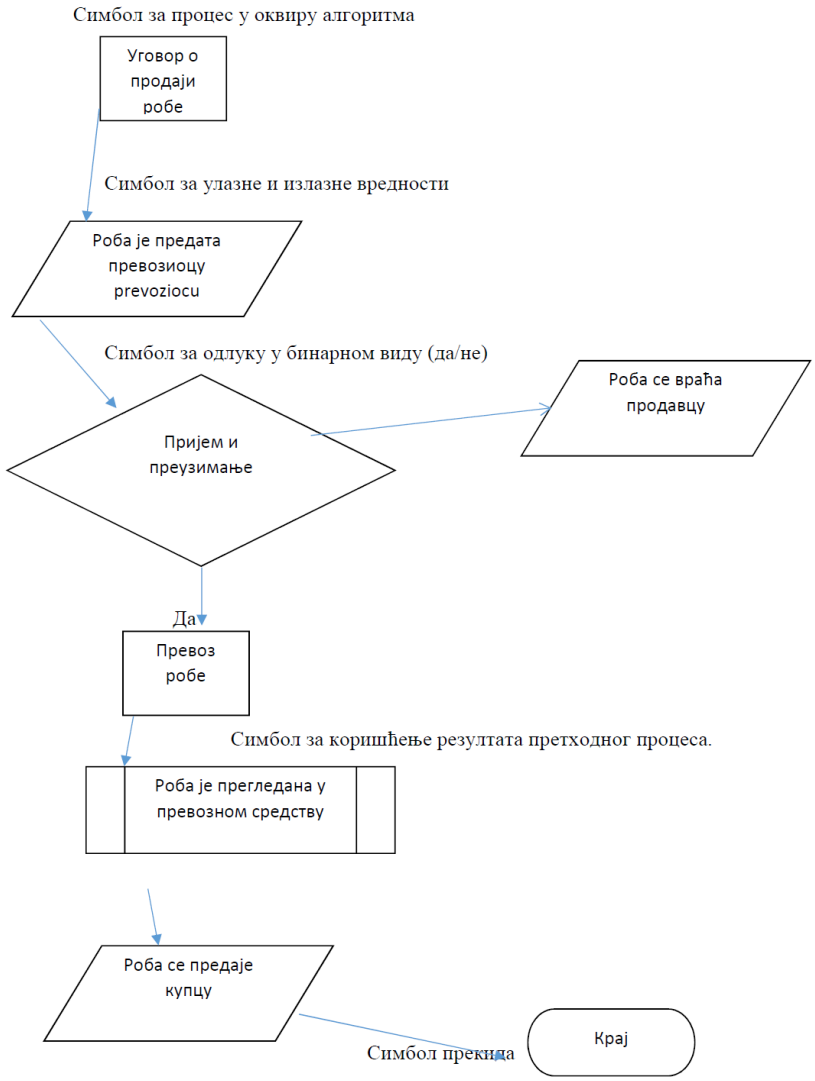
4. 

5. 

6. 

Стрелице показују правац кретања.

Ево примера илустрације алгоритма уговора о продаји кроз дијаграма тока



5. Алгоритмизација: отворена питања

Програмски код не само чини сајберспејс тиме што јесте, већ га и регулише тако што омогућава дигиталне активности. Заступници изједначавања програмског кода и права описану чињеницу узимају као аргумент за ново разумевање појма регулативе: оно је могуће искораком изван

традиционалних схватања права и његових елемената. Носилац новог разумевања регулативе је нови регулатор: програмски код (*Lessig*, 2006: 5). Наведено схватање, мада се *de lege lata* може сматрати екстремним, делом је већ данас вредно пажње због процеса рада на механизмима и технолошким алатима за алгоритмирање уговора (кроз појам паметних уговора).

У вокабулар права (као део мултидисциплинарности његовог проучавања условљеног упливом технологије у дефинисање, спровођење и контролу правне норме) улази појам дигиспруденције (*Diver*, 2019: 304). Дигиспруденција је појам који дефинише услове нормативности кода. Нормативни значај (и легитимност) кода требало би да се осигура већ у моменту настанка правне норме: то значи да би правници у фази дефинисања садржине правног правила требало да као полазну претпоставку имају да ће то правило бити предмет дигитализације, алгоритмирања и кодирања (када је реч о уговорним одредбама, ради се о некој од фаза трајања уговорног односа-закључење, извршење, контрола). Тиме би и право уважило чињеницу да дигитализација више није само фрагмент друштвене стварности који има интеракцију са правом: она је постала општи друштвени оквир (вештачка интелигенција се сматра технологијом опште примене – е. *General Purpose Technology*). Право функционише у дигитализованом оквиру. Стога би уговор (као прва степеница у процесу уједначавања норме и програмског кода) морао да настаје (тамо где је могуће, када се ради о уговорима чије одредбе омогућавају алгоритмирање) уважавајући реалност уплива технологије у настанак, функционисање и конгролу уговорних одредби.

Концепција алгоритмизације уговора предмет је критичке пажње академске и пословне јавности: у том смислу, постоје упозорења у погледу применљивости овог концепта у области уговорног права. У техничкој литератури указује се да уговор који би требало да се алгоритмира мора да дефинише коначне, узајамно искључиве и исцрпне односе између страна. Он се не може свести на логичке операције: уговор да би био извршив мора да буде комплетан. Питање је да ли алгоритам може да обезбеди такву потпуност. Илустративан је пример покушаја да се дефинише логичка структура споразума о зајму (видети: Flood, Goodenough, 2017: 19–24). Дефинисање структуре је требало да: 1. омогући да се уговор креира и спроводи на основу доступних финансијских рачунарских алата; 2. докаже да фундаментална правна структура адекватно формулисаних финансијских уговора прати логику развоја уговора на начин да се та логика математички формализује. Описани процес је условио да уговор о зајму буде симплификован до мере која га чини комерцијално нереалним.

У пракси нема таквих уговора: комплексност уговора у форми природног језика одражава комплексност трансакције која је њиме регулисана. Та се комплексност не може увек и у потпуности конвертовати у алгоритам и пренети у код. Стога паметни уговори морају да буду реално структуирани: не да буду „примери играчке", већ да допринесу практичној употреби.

Алгоритмизација уговора је процес који траје и ту је да остане. Значај овог процеса је неспоран: његов домет, динамика и претпоставке су и даље делимично дефинисане и тестиране. Потребан (али не и довољан услов) је да се алгоритамски/стохастички начин размишљања легитимизује у процесу дефинисања уговорних одредби. Даљи развој ће зависити од функционисања других елемената у окружењу у коме би кодирани уговори функционисали, а можда понајвише од комерцијалног одговора на читав процес: алгоритмирање уговора мора да постане тржишно исплативо понашање да би добило ширу примену.

Литература и извори

Allen, L. E. (1956). Symbolic logic: A razor-edged tool for drafting and interpreting legal documents. *Yale LJ*, *66*, 833.

Ashley, K. D. (2017). *Artificial intelligence and legal analytics: new tools for law practice in the digital age*. Cambridge University Press.

Barraclough, T., Fraser, H., & Barnes, C. (2021). Legislation as Code for New Zealand.

Brookshear, J. G. (2012). *Computer science: an overview*. Boston: Addison-Wesley.

Christodoulou, M., Szczygieł, E., Kłapa, Ł., & Kolarz, W. (2018). Algorithmic and Programming.

Cvetković, P. (2021). Синтеза правног текста и програмског кода: случај рикардијанског уговора. *Зборник радова Правног факултета у Нишу*, (90), 61–76.

Cvetković, P. (2020). Блокчејн као правни феномен: уводна разматрања. *Зборник радова Правног факултета у Нишу*, (87), 127–144.

Diver, L. E. (2019). Digisprudence: the affordance of legitimacy in code-as-law. University of Edinburgh, doctoral thesis.

Flood, M. D., & Goodenough, O. R. (2017). Contract as automaton: the computational representation of financial agreements. *Office of Financial Research Working Paper*, (15-04). Приступљено: 10. 09. 2021. https://ccl.yale.edu/sites/default/

files/files/OFRwp-2015-04_Contract-as-Automaton-The-Computational-Representation-of-Financial-Agreements.pdf

Goldoni, M. (2015). The politics of code as law: towards input reasons. In: Reichel, J. and Lind, A.S. (eds.) *Freedom of Expression, the Internet and Democracy.* Brill: Leiden, pp. 115–133.

Jovanović, N. (2005). *Uvod u programiranje*. Viša poslovna školam str. 51–91.

Kaulartz, M. (2019). Smart Contract Dispute Resolution in Fries, Martin, and Boris P. Paal. *Smart contracts*. Mohr Siebeck,

Koch, K. L. (2005). A Multidisciplinary Comparison of Rules-Driven Writing: Similarities in Legal Writing, Biology Research Articles, and Computer Programming. *Journal of Legal Education*, *55*(1/2), 234–251.

Kuhn, T. (2014). A survey and classification of controlled natural languages. *Computational linguistics*, *40*(1), 121–170.

Lauslahti, K., Mattila, J., & Seppala, T. (2017). Smart contracts–How will blockchain technology affect contractual practices?. *Etla Reports*, (68).

Lawrence Lessig, *Code: Version 2.0* (Basic Books 2006)

Lipshaw, J. M. (2018). The Persistence of "Dumb" Contracts. Приступљено 10. 07. 2020. https://www.researchgate.net/profile/Jeffrey_Lipshaw/publication/326475422_The_Persistence_of_'Dumb'_Contracts/links/5c3b4db7a6fdccd6b5a9e41f/The-Persistence-of-Dumb-Contracts.pdf

Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., Hammond, P., & Cory, H. T. (1986).

Surden, H. (2012). Computable contracts. *UCDL Rev.*, *46*, 629.

Susskind, R. E. (1986). Expert systems in law: A jurisprudential approach to artificial intelligence and legal reasoning. *The modern law review*, *49*(2), 168–194.

The British Nationality Act as a logic program. *Communications of the ACM*, *29*(5), 370–386. Приступљено: 20. 09. 2021. http://www.doc.ic.ac.uk/~rak/papers/British%20Nationality%20Act.pdf

Van Emden, M. H., & Kowalski, R. A. (1976). The semantics of predicate logic as a programming language. *Journal of the ACM (JACM)*, *23*(4), 733–742.

Von Haller Grønbæk, M. (2016). *Blockchain 2.0, Smart Contracts And Challenges.* Приступљено 02. 03. 2020. https://www.twobirds.com/~/media/pdfs/in-focus/fintech/blockchain2_0_martinvonhallergroenbaek_08_06_16.pdf

Werbach, K., & Cornell, N. (2017). Contracts ex machina. *Duke LJ*, *67*, 313.

Predrag Cvetković, LL.D.,
Full Professor,
Faculty of Law, University of Niš

CONTRACT AS AN ALGORITHM: INTRODUCTORY CONSIDERATIONS

Summary

Legal norms contained in text-driven contracts (as well as in statutes and bylaws), which are written in natural language, can be subject of algorithmic conversion in certain phases of the contract circle (implementation, monitoring, control, interpretation). The application of the blockchain concept as a structural pattern opens the possibility of creating a code-driven contract with automated execution: a "smart" contract. Algorithmization is understood as a process which enables the text of the contract to be translated into a format that is understandable to software developers. To this end, the use of the following methodologies is proposed: design of a pseudocode, application of formal logic symbols and the use of flowcharts. Successful conversion of legal prose into a code calls for cooperation between lawyers and programmers. The framework of that cooperation is the establishment of the so-called "Legal Expert System" (LES). Originally conceived by lawyers, LES is a program which allows the algorithm to solve the problem of contract execution. Contract algorithmization should convert contracts from prose to a code, while preserving contract validity and efficiency. For the time being, smart contracts cannot regulate commercially complex scenarios; thus, the de lege lata application of smart contracts as a complete replacement for traditional (analogous) contracts is excluded. A potential object of algorithmization are the primary instructions aimed at executing the characteristic performance of the contract. Contract algorithmization is an ongoing process, which is here to stay. The significance of this process is indisputable but its scope, dynamics and assumptions are still only partially defined and tested. The necessary condition (but hardly a sufficient one) is to legitimize the conception of a contract as an algorithm in the process of defining contractual provisions. Further development of this concept will depend on the functioning of other elements in the environment where code-driven contracts would be used and, above all, on the commercial response to the entire process of contract algorithmization. In effect, in order to be widely applied, contract algorithmization must become a commercially viable activity.

Keywords*: contract, algorithm, smart contracts, pseudocode, formal logic.*