УНИВЕРЗИТЕТ У БЕОГРАДУ

Математички факултет

Катедра за рачунарство и информатику

Игре парности и њихово решавање свођењем на SAT

-МАСТЕР РАД-

Студент:
Игор Родић

Ментор:
др Предраг Јаничић

Комисија:
др Филип Марић
др Марко Маликовић

Београд,

2020.

Садржај

1. Увод

Игре парности представљају игре за два играча на коначном усмереном тежинском графу. Сваком чвору се додељује целобројна тежина (приоритет). Игра се тако што играчи у сваком потезу померају жетон од чвора до чвора графа.Током бесконачне игре неки чворови ће бити посећени више пута. Ако је највећа тежина свих чворова кроз које се пролази бесконачно пута у току (бесконачне) партије паран број, онда је победник први играч, а иначе је победник други играч.

Решавање игара парности представља одређивање који од два играча има победничку стратегију (низ потеза који њему осигуравају победу) у односу на дату стартну позицију. Доказано је да је решавање игара еквивалентно провери модела у модалном μ-рачуну, па због тога игре парности могу имати примене у пољу аутоматске верификације софтвера и хардвера. За њихово решавање још увек није пронађен алгоритам полиномске сложености и самим тим оне представљају веома изазован и важан проблем у теоријском рачунарству.

Убрзаним развојем све бољих SAT решавача, као и све моћнијих рачунара, многи проблеми високе комплексности могу на задовољавајући начин да се реше техником свођења на SAT, што важи и за овај проблем. Циљ овог рада је да се развије систем за решавање игара парности свођењем на SAT проблем. Биће анализирана временска и просторна сложеност поступка решавања. Рад садржи и преглед игара парности и постојећих стратегија за њихово решавање. Стратегије за решавање игара парности су имплементиране у програмском језику C++ који је у стању да погодно опише и ефикасно реши тај проблем.

Овај рад се у прегледним деловима ослања на постојеће прегледе игара парности [47], модалног μ-рачуна [36], и свођења игара парности на SAT [16].

2. Игре парности

Игре парности први пут су поменуте почетком 70-их година прошлог века у раду Стивена Кола (енг. *Steven Cole*) [6], док експанзију доживљавају у другој половини 90-их година пре свега радом Марцина Јурџинског (енг. *Marcin Jurdziński*). Пре него што почнемо са дубљом анализом игара парности, која претходи њиховом свођењу на SAT проблем, прво ћемо у овом поглављу представити основне информације везане за игре: шта су оне, како је играју, како се одређује победник, које су стратегије за играче, као и која је сложеност њиховог решавања.

2.1 Основе

Дефиниција 2.1.1. (Стандардна игра парности). Стандардне игре парности *састоје се од усмереног графа $G = (V, E)$, где је V скуп чворова, који се састоји од два подскупа V_0 и V_1, а $E \subseteq V \times V$ скуп грана и функције приоритета $\alpha : V \to \mathbb{N} \setminus 0$ која додељује вредности приоритета сваком чвору. Такву игру означавамо са $\mathcal{G} = (V, E, \alpha)$. Игру играју два играча, P_0 коме припада подскуп чворова V_0, и P_1 коме припада подскуп чворова V_1. Претпоставка је да сваки чвор мора да има бар једну излазну грану.*

Игра се тако што играчи померају жетон од чвора до чвора графа почевши од чвора $v_0 \in V$. Играч коме припада чвор v_0 започиње игру и бира једну од излазних грана (v_0, v_1) тог чвора и помера жетон до чвора v_1. Игру по истом принципу наставља играч коме припада чвор v_1. Играчи не морају нужно да изаберу исту грану у истом чвору. Играјући игру играчи конструишу бесконачну партију (путању) $\pi = v_0, v_1, v_2, \ldots$ где је $\alpha(v_0)$, $\alpha(v_1)$, $\alpha(v_2)$,... низ приоритета одговарајућих чворова, а $\mathrm{Inf}(\pi)$ је скуп $\{v \in V \mid v$ се појављује бесконачно много пута у $\pi\}$. Победника у игри одређује највећи приоритет у скупу приоритета тј. $max\{\alpha(v) \mid v \in \mathrm{Inf}(\pi)\}$): ако је он паран, побеђује P_0, у супротном побеђује P_1. Победника аналогно може

одредити и најмањи приоритет, у ком случају играч P_0 *побеђује ако је* $min\{\alpha(v) \mid v \in \mathrm{Inf}(\pi)\}$ *паран, док у супротном побеђује играч* P_1.

Стандардна игра парности
Улаз: игра парности $\mathcal{G} = (V, E, \alpha)$ и чвор $v \in V$.
Излаз: да ли играч P_0 има победничку стратегију из v у $\mathcal{G}$.

Остаје да дефинишемо још неколико параметара и термина које ћемо користити. За игру $\mathcal{G} = (V, E, \alpha)$ константе n, m и d дефинишемо на следећи начин:

- $n = |\mathrm{V}|$ број чворова у графу G
- $m = |\mathrm{E}|$ број грана у графу G
- $d = |\alpha(\mathrm{V})|$ број различитих приоритета у игри $\mathcal{G}$

Са $V^p = \{v \in \mathrm{V} \mid \alpha(v) = \mathrm{p}\}$ означавамо све чворове приоритета p. Слично, са $V^{\geq p}$ означавамо скуп чворова приоритета бар p, а са $V^{\leq p}$ скуп чворова приоритета највише p.

Функцију припадности чвора $v \in \mathrm{V}$ дефинишемо са:

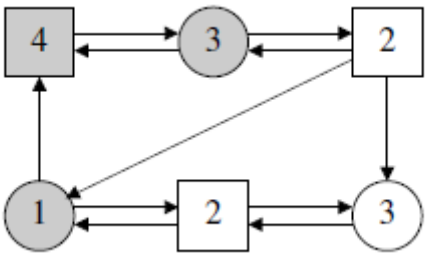
$$o(v) = \begin{cases} 0 & \text{ако } v \in V_0 \\ 1 & \text{ако } v \in V_1 \end{cases}$$

Дефиниција 2.1.2. (Подигра игре парности). *Подигру игре* $\mathcal{G} = (V, E, \alpha)$ *обележавамо са* $\mathcal{G}' = (V', E', \alpha')$, *ако је граф* $G' = (V', E')$ *подграф графа* $\mathcal{G} = (V, E)$ *и за сваки чвор* $u \in V'$ *постоји чвор* $v \in V'$ *тако да* $(u, v) \in E'$.

Пример 2.1.1. *Слика 3.4 приказује пример игре парности. Задати граф садржи шест чворова, чворови означени круговима припадају играчу* P_0, *док чворови означени квадратима припадају играчу* P_1. *Бројеви записани унутар чворова означавају њихове приоритете.*

У овом примеру, играч P_0 *може да оствари победу ако се жетон нађе на неком од осенчених чворова. Заиста, ако* P_0 *форсира прелазак у чвор са приоритетом 4, из својих чворова са приоритетима 1 и 3, играч* P_1 *нема избора него да настави игру преласком у чвор са приоритетом 3 у ком случају ће први играч вратити жетон назад у чвор приоритета 4. На тај начин игра ће се бесконачно одвијати између чворова са приритетима 3 и 4, а у том случају, како је највећи приоритет који се појављује у тој игри паран (4), победник је* P_0. *Слично претходном, играч* P_1 *може да оствари победу из неосенчених чворова тако што ће из својих чворова приоритета два форсирати противников чвор приоритета 3, тиме стварајући бесконачну*

игру 2, 3, 2, ... Како је највећи приоритет који се бесконачно појављује у тој игри непаран (3), победник је P_1.



Слика 2.1: Пример игре парности

2.2 Стратегије

Игра може да поседује одређену стратегију, тј. низ акција које играча воде до победе (или до најбоље могуће позиције), а самим тим постоји и настојање да се та стратегија пронађе. У наставку ћемо дефинисати стратегију за игре парности, као и шта одређену стратегију карактерише као победничку.

Дефиниција 2.2.1. (Стратегија играња). Стратегија δ_0 *за* P_0 *(тј.* δ_1 *за* P_1*) је функција* $\delta_0 : V^*V_0 \to V$ $(\delta_1 : V^*V_1 \to V)$ *која сваком чвору* $v \in V^*V_0$ $(\in V^*V_1)$ *у партији* π *додељује чвор* w *такав да* $(v, w) \in E$. *Играч* P_i *користи стратегију* δ_i *у партији* $\pi = \pi_1\pi_2...\pi_k...$, *ако* $\pi_{k+1} = \delta_i(\pi_1...\pi_k)$ *за сваки чвор* $\pi_k \in V_i$. *Стратегија* δ_i *је* победничка *за играча и одређени почетни чвор* $v \in V$ *ако тај играч побеђује у свакој партији која почиње из чвора* v *и користи се стратегија* δ_i. *Све игре које почињу из чвора* v *означавамо са* $\mathcal{G}(v)$.

Стратегија, дакле, одређује следећи чвор у који ће играч померити жетон на основу дотадашњих потеза, тј. путање $v_0, ..., v_k$ сачињене од низа посећених чворова. *Позициона стратегија* не узима у обзир целу путању већ само последњи чвор v_k. Оваква стратегија назива се још и *стратегија без памћења* зато што за њену примену није потребно памтити претходне чворове, потребно је знати само у ком се чвору тренутно налази жетон. Победничка позициона стратегија за играча P_0 из чвора $v \in V$ представља позициону стратегију у којој играч P_0 побеђује у свакој игри која почиње из чвора v користећи ту стратегију.

Дефиниција 2.2.2. (Услов победе). *За фиксирани почетни чвор v кажемо да играч P_i побеђује у игри $\mathcal{G}(v)$ ако постоји стратегија δ уз чије коришћење P_i може да победи у свакој партији која почиње у чвору v, без обзира на стратегију и потезе противника.* Решавање игре $\mathcal{G}$ *представља проналажење победника игре $\mathcal{G}(v)$ за сваки чвор $v \in V$. Слично, кажемо да играч побеђује у игри $\mathcal{G}$ ако постоји стратегија δ таква да коришћењем те стратегије он побеђује у игри $\mathcal{G}(v)$ за сваки чвор $v \in V$.*

Победнички регион или победнички скуп за играча P_i представља скуп чворова из којих P_i има победничку стратегију. Победнички регион у игри $\mathcal{G}$, за играча P_i, обележавамо са $W_i(\mathcal{G})$.

Дефиниција 2.2.3. (Решавање игре парности). Решавање игре парности $\mathcal{G}$ *за почетни чвор $v \in V$ представља проверу да ли играч P_0 има победничку стратегију из v у $\mathcal{G}$.*

Дефиниција 2.2.4. (Коначна игра парности). Коначна игра парности $\mathcal{G}' = (V, E, \alpha)$ *представља игру парности која се игра све док се неки чвор не посети два пута, у ком тренутку се прекида.*

Победник у коначној игри парности одређује се на следећи начин: када дођемо до неког чвора други пут тиме можемо да сматрамо да смо затворили циклус који је почео његовим првим појављивањем, а завршио се другим, дакле, циклус би садржао све чворове од његовог првог појављивања до другог. Највећи приоритет који се јавља у том циклусу одређује победника, ако је паран побеђује играч P_0, ако је непаран побеђује играч P_1. Разог за ово јесте чињеница да ће се тај циклус, будући да играчи играју игру по непроменљивим стратегијама, након свог првог појављивања јављати бесконачно много пута, изнова и изнова, јер ће играчи бирати исте потезе у истим ситуацијама.

Следећа теорема доказује одлучивост проблема решавања игара парности, тј. да један од играча мора да има победничку стратегију. Доказ је заснован на редукцији стандардне игре парности на коначну игру парности која, будући да је коначна, представља игру са нултом сумом и савршеним информацијама (енг. *finite zero-sum game of perfect information*) и као таква потпада под Цермелову теорему [49] (енг. *Zermelo's theorem*) теорије игара.

Теорема 2.2.1. *За сваку игру парности $\mathcal{G}$ и за сваки почетни чвор v, или играч P_0 има победничку стратегију из v, или играч P_1 има победничку*

стратегију из v, тј. $W_0(\mathcal{G}) \cup W_1(\mathcal{G}) = V$. Такође, ако играч има победничку стратегију у коначној игри парности $\mathcal{G}' = (V, E, \alpha)$ из чвора $v \in V$, онда има победничку стратегију и у стандардној игри парности $\mathcal{G} = (V, E, \alpha)$ из чвора $v \in V$.

Доказ. Претпоставимо да играч P_0 има победничку стратегију у коначној игри парности $\mathcal{G}' = (V, E, \alpha)$ из чвора $v \in$ V. Сада играмо игру $\mathcal{G} = (V, E, \alpha)$, а паралелно у игри $\mathcal{G}'$ вучемо исте потезе. Када дођемо до циклуса бришемо га из памћења игре $\mathcal{G}'$ и настављамо да играмо победничку стратегију из $\mathcal{G}'$ из последњег посећеног чвора у циклусу, али тако да се не вратимо у тај исти циклус. Будући да играч P_0 игра по победничкој стратегији у $\mathcal{G}'$ сваки циклус има паран максимални приоритет. Што даље значи да је и највећи приоритет у $\mathcal{G}$ паран јер је он максимум максималних приоритета у бесконачно много циклуса. Дакле играч P_0 побеђује и у игри $\mathcal{G}$.

Чињеница да један од играча мора да има победничку стратегију у игри $\mathcal{G}$ из чвора $v \in V$ следи из првог дела доказа. Ако играч има победничку стратегију у игри $\mathcal{G}$ из чвора v, онда има и победничку стратегију на њеној коначној верзији $\mathcal{G}'$ из чвора v. Игру почињемо у чвору v и играмо све док се у њега не вратимо. На овај начин формирали смо циклус од v до v чији највећи приоритет, будући да се вредности приоритета узимају из скупа $\mathbb{N} \setminus 0$, мора бити паран или непаран. На таквом циклусу или је победник играч P_0, ако је највећи приоритет паран, или играч P_1, ако је највећи приоритет непаран. □

Теорема 2.2.2. *У игри парности $\mathcal{G}$, ако играч има победничку стратегију из неког чвора $v \in V$, онда има и победничку позициону стратегију из тог чвора.*

Доказ. Доказ се врши индукцијом по броју чворова који имају више од једне излазне гране. Нека је $z \in V$ случајно одабран чвор који има више од једне излазне гране. Ако у чвору z постоји победничка стратегија желимо да докажемо да је она и победничка позициона стратегија, а онда индукцијом да то докажемо за све чворове.

Слично доказу теореме 2.2.1, дефинишемо коначну игру $\mathcal{G}'_z$ која, када се дође у чвор z, заборавља све до тада одиграно потезе, а завршава се када се вратимо у почетни чвор v. Оваква игра такође је коначна јер треба да обиђемо највише $2n$ чворова да бисмо је завршили, тј. највише n да дођемо до z или да направимо циклус, а ако смо дошли до z, онда још највише n да

направимо циклус. Слично доказу теореме 2.2.1, можемо доказати да ако у $\mathcal{G}'_z$ постоји победничка стратегија за једног од играча, она постоји и у $\mathcal{G}$.

Даље, ако претпоставимо да је $z \in V_0$, онда је било која победничка стратегија за играча P_0 позициона из чвора z у $\mathcal{G}'_z$ јер се тај чвор обилази највише једанпут и брише се из памћења игре када се то деси. Будући да P_0 игра игру $\mathcal{G}$ исто као и игру $\mathcal{G}'_z$ може се закључити да је z победничка позициона стратегија и у $\mathcal{G}$. И коначно, ако се излазне гране из z које нису изабране обришу, онда по индуктивној хипотези P_0 има победничку позициону стратегију која имплицира победничку позициону стратегију у $\mathcal{G}$. □

2.2.1 Принудни скупови

Ако имамо скуп чворова $S \subseteq V$, принудни скуп (енг. *force set*) скупа S за играча P_i је скуп свих чворова из којих играч P_i може да исфорсира прелазак у скуп S [33]. Други назив за принудне скупове који се често користи у литератури јесте привлачни скупови (енг. *attractor sets*)

Дефиниција 2.2.5. (Принудни скуп). *За играча P_i и $S \subseteq V$ дефинишемо* принудни скуп $F_i(S)$ *као фиксну тачку следећег система једначина:*

$$F_i^0(S) = S$$
$$F_i^{k+1}(S) = F_i^k(S) \ \cup$$
$$\{u \in V_i \mid \exists v \in F_i^k(S).(u,v) \in E\} \ \cup$$
$$\{u \in V_{1-i} \mid \forall v \in V.(u,v) \in E \implies v \in F_i^k(S)\}$$

Израчунавање почињемо од нашег почетног скупа S и одатле рекурзивно градимо принудни скуп додавањем нових чворова. Ако чвор припада играчу P_i онда га додајемо у текући скуп само ако постоји чвор у који може из њега да се пређе (повезани су гранама) а који је већ у нашем текућем скупу. Ако чвор припада другом играчу, P_{1-i}, онда га додајемо у текући скуп ако се сваки чвор у који из њега може да се пређе већ налази у текућем скупу. Параметар k означава број корака који нам је потребан да би смо дошли до фиксне тачке наше итерације. Фиксна тачка је достигнута када дођемо до ситуације да је $F_i(S) = F_i^{k+1}(S)$, тј. да у последњој итерацији нисмо додали ниједан нови чвор у текући скуп.

Теорема 2.2.3. *Нека је $\mathcal{G} = (V, E, \lambda)$ игра парности и $S \subseteq W_i(\mathcal{G})$ скуп који је део победничког региона играча P_i. Онда је $F_i(S) \subseteq W_i(\mathcal{G})$, а $\mathcal{G}' = \mathcal{G} \backslash F_i(S)$ је подигра игре $\mathcal{G}$ и $w \in W_i(\mathcal{G}) \iff w \in W_i(\mathcal{G}')$.*

Доказ. Чињеница да је $F_i(S) \subseteq W_i(\mathcal{G})$ је очигледна и следи из чињенице да по дефиницији принудног скупа играч P_{1-i} не може да напусти скуп $W_i(\mathcal{G})$.

Ако $\mathcal{G}'$ није подигра игре $\mathcal{G}$, онда мора да постоји чвор $v \in V(\mathcal{G}')$ који нема следбеника у $V(\mathcal{G}')$. Нека је j најмањи индекс тако да v има све следбенике у $F_i^j(S)$, по дефиницији принудног скупа $v \in F_i^{j+1}(S)$ што значи да такав чвор не постоји.

Претпоставимо да је $w \in W_i(\mathcal{G})$. Из тога следи да постоји победничка стратегија δ таква да не постоји противников циклус у $\mathcal{G}$, али по дефиницији принудног скупа F_i није могуће да $v \in V(\mathcal{G}')$ и $\delta(v) \in F_i(S)$, тако да је δ победничка стратегија у $\mathcal{G}'$. Други смер доказује се аналогно. □

2.3 Уопштења игара парности

Поред игара парности постоје и игре које су општије и једноставније од њих. У овом поглављу приказаћемо две такве игре које су нам важне из следећих разлога:

1. Сложеност решавања ових игара је у истој класи сложености као и решавање игара парности
2. Проблем решавања игара парности може се свести на проблем решавања ових игара.

Те игре су игре средње вредности и једноставне стохастичке игре.

2.3.1 Игре средње вредности

Игре средње вредности (енг. *mean payoff game*) увели су Еренфјухт (енг. *Ehrenfeucht*) и Мичелски (енг. *Mycielski*) [10] 1979. године, а Цвик (енг. *Zwick*) и Патерсон (енг. *Paterson*) [51] доказали су да проблеми проналажења приоритета чворова и стратегија за играче припадају класи сложености NP∩coNP. Свођење игара парности на игре средње вредности открили су независно Јерум (енг. *Jerrum*) [44] і Пури (енг. *Puri*) [40]. Преко тог свођења, у раду Јурџинског [21], доказано је да и игре парности припадају класи сложености UP∩coUP.

Класа UP представља класу недвосмислених недетерминистичкх проблема решивих у полиномском времену (енг. *unambiguous non-deterministic polynomial-time*) које препознају недвосмислене Тјурингове машине (енг. *unambiguous Turing machines*). Чињеница да проблем припада класи NP значи да за задато решење можемо лако проверити да ли је исправно, док

припадност класи coNP значи да за задато решење можемо лако проверити да ли је неисправно. Код класа UP и coUP ситуација је слична само што се не може проверити свако решење већ само кратки јединствени сертификати (енг. *unique short certificates*) што значи да решење сме да има само једну јединствену путању која задовољава критеријуме исправног решења (или неисправног у случају класе coUP). Класа NP садржи класу UP.

Дефиниција 2.3.1. (Игра средње вредности). Игра средње вредности $\mathcal{G} = (V, E, \omega, \nu)$ *састоје се од усмереног графа* $G = (V, E)$, *где је* V *скуп чворова, који се састоји од два подскупа* V_0 *и* V_1, *функције приоритета* $\omega : E \to \{-w,...,0,...,w\}$ *која додељује целобројну вредност између* $-w$ *и* w *свакој грани из* $\mathcal{G}$, *и праћа вредности* $\nu \in \mathbb{N}$. *Игра се игра исто као и игра парности, једино се разликује услов победе: играч* P_0 *побеђује у бесконачној партији* $\pi = \pi_1\pi_2...$ *ако*

$$\liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \omega((\pi_i, \pi_{i+1})) \geq \nu$$

2.3.2 Једноставне стохастичке игре

Једноставне стохастичке игре (енг. *simple stochastic games*) су, за разлику од игара парности и игара средње вредности, игре вероватноће. Увео их је Шапли (енг. *Shapley*) [43] 1953. године, а Кондон (енг. *Condon*) [7] је доказао да и оне припадају класи сложености NP∩coNP. Свођење игара средње вредности на једноставне стохастичке открили су Цвик и Патерсон [51].

Дефиниција 2.3.2. (Једноставна стохастичка игра). Једноставна стохастичка игра $\mathcal{G} = (V, E, \nu_0, \nu_1)$ *састоји се од графа* $G = (V, E)$ *који садржи два специјална чвора* ν_0, $\nu_1 \in V$ *који се зову 0-понор и 1-понор и немају излазне гране. Скуп чворова* $V \setminus \{\nu_0, \nu_1\}$ *дели се на 3 дела* V_0, V_1 *и* $V_{1/2}$. *Као и у претходним случајевима, играч* P_0 *контролише скуп чворова* V_0, *играч* P_0 *скуп чворова* V_1, *а скуп чворова* $V_{1/2}$ *зове се* скуп просечних чворова *и сваки чвор који му припада има тачно две излазне гране. Просечни чворови разликују се од обичних јер не припадају ниједном играчу већ када се дође до просечног чвора следећа грана се бира случајно са вероватноћом 1/2 за сваку од грана. Играч* P_0 *побеђује у игри ако долази до 0-понора са вероватноћом од најмање 1/2.*

2.4 Комплексност

Као што је поменуто у претходном поглављу, решавање игара парности спада у класу сложености UP∩coUP, што је доказао Јурџински [21] 1998. године преко свођења игара парности на игре средње вредности. Доказ Јурџинског превазилази домен овог рада те неће бити детаљно обрађен, уместо њега приказан је једноставнији интуитивни доказ који има за сврху приказ основних параметара који утичу на високу комплексност игара парности.

Лема 2.4.1. *Игра парности $\mathcal{G}$ за једног играча, тј. игра где други играч не поседује ниједан чвор, може да се реши у времену $O(n^3)$.*

Доказ. Будући да један играч (нпр. P_0) поседује све чворове, он побеђује у игри из неког почетног чвора v ако и само ако постоји циклус до ког може да се дође из v, а чији највећи приоритет је паран.

Проверу да ли постоји такав циклус започињемо уклањањем чворова који сигурно нису садржани у њему. Прво уклањамо чворове до којих не може да се дође из v. Затим проверавамо да ли је највећи приоритет у $\mathcal{G}$ паран, ако није, уклањамо све чворове највећег приоритета, јер будући да су они непарни сигурно нису садржани у нашем траженом циклусу. Ако приоритет јесте паран, проверавамо да ли је бар један чвор који га садржи део циклуса, ако такав циклус постоји значи да P_0 има победничку стратегију из v. У супротном, ако такав циклус не постоји понављамо исту процедуру док нам не остане ниједан чвор, у ком случају закључујемо да P_0 нема победничку стратегију у $\mathcal{G}$, што даље значи да је има други играч P_1.

У сваком кораку уклања се најмање један чвор, а провера да ли је чвор у циклусу захтева $O(m)$ времена, дакле $O(mn)$, док је укупно време извршавања $O(n^3)$.

□

Теорема 2.4.1. *Стандардна игра парности припада класи сложености NP∩coNP.*

Доказ. Недетерминистички можемо да пробамо да погодимо победничку позициону стратегију δ за играча P_0 (у теореми 2.2.2 доказали смо да онда P_0 има и победничку стратегију) а затим можемо да је проверимо у игри $\mathcal{G}_\delta$. С обзиром да је $\mathcal{G}_\delta$ игра за једног играча, P_0 не може да бира куда ће се жетон кретати и самим тим не може да утиче на игру. Будући да је

овај алгоритам недетерминистички, овим смо доказали да решавање игара парности припада класи NP.

Слично, ако играч P_0 нема победничку стратегију има је играч P_1, што подразумева исти алгоритам само овог пута тражећи контрапример, тј. да играч P_0 нема победничку стратегију. Ова чињеница сврстава решавање игара парности и у класу coNP. □

3. Модални μ-рачун

Важна тема у проучавању игара парности јесте модални μ-рачун. Будући да је доказано да је решавање игара еквивалентно проверавању модела у модалном μ-рачуну [11], игре парности налазе своју примарну примену у пољу аутоматске верификације софтвера и хардвера.

У овом поглављу прво ћемо видети шта представљају модална и темпорална логика, затим шта је проверавање модела и како може да се користи у играма, шта је модални μ-рачун, као и како проблем решавања игара парности може да се сведе на проблем провере модела μ-рачуна.

3.1 Модална логика

Модална логика (као формализам модала) омогућава коришћење појмова који се природно користе у језику а означавају *могућност* и *нужност*. Тиме она проширује описну изражајну моћ стандардних логика. До модалне логике долазимо проширивањем стандардних логика са два унарна оператора:

- $\Box$ - квадрат (означава "нужно") и
- $\diamond$ - дијамант (означава "могуће")

Оператори се могу изразити један преко другог на следећи начин:

- $\diamond\alpha =_{def} \neg\Box\neg\alpha$
- $\Box\alpha =_{def} \neg\diamond\neg\alpha$.

Пример 3.1.1. *Реченицу "Могуће је да ће данас падати киша" можемо означити преко модалне логике са $\diamond k$, где k представља реченицу "Киша ће падати". Док реченицу "Данас ће нужно падати киша" означавамо са $\Box k$. Ако кажемо да је "могуће да ће падати киша", значи да није "нужно да неће падати киша", и обратно. Овакву реченицу није могуће описати стандардним логикама.*

3.1.1 Темпорална логика

Када се говори о верификацији реактивних система (система који су у непрестаној интеракцији са околином) преко провере модела, најчешће се не користи основна модална логика већ, њен посебан случај, темпорална логика. Темпорална логика (енг. *temporal logic*) интерпретира модалне операторе у времену, па истинитост формуле зависи и од тренутка у коме се она валуира, што је чини погодном за верификацију реактивних система који се непрестано мењају [12]. Такође, темпорална логика даје нам нова тумачења оператора $\Box$ и $\diamond$. Нова тумачења су:

- $\diamond\alpha = \alpha$ ће важити у неком тренутку времена t
- $\Box\alpha = \alpha$ важи у сваком тренутку времена t.

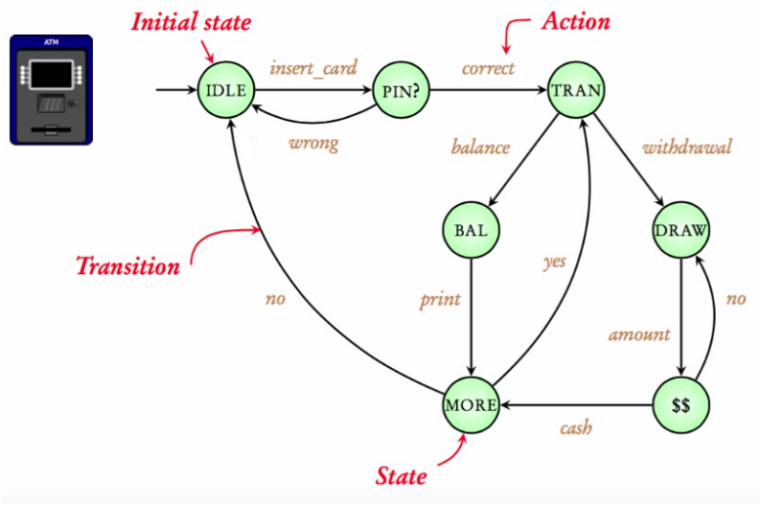
Уз помоћ ових проширења лакше је описати променљиво понашање система који треба верификовати.

3.2 Проверавање модела

Проверавање модела (енг. *model checking*) представља начин верификације софтверских и хардверских система кроз валидацију и тестирање особина самог система. Како наши животи и њихова угодност и функционалност тако и напредак човечанства свакодневно зависе од поузданости критичних компјутерских система који нас окружују, поставља се питање да ли постоји начин да се рад тих система и формално провери, тј. да ли можемо да будемо сигурни да ће њихов рад бити без грешака. Примери оваквих система су контролери у авионима, контроли лета, беспилотним летелицама и метроима, нуклеарним електранама, свемирским летилицама, сателитима, системима за одржавање животних функција пацијента, као и дозирања медикамената, системима за сигурност рачуна у банкама, берзе и банкомата. Један од начина за верификацију критичних система јесте кроз проверу модела у терминима темпоралне логике.

Проблем се описује на следећи начин: ако систем представимо апстрактним моделом $\mathcal{M}$, проверити да ли систем поседује задато својство φ. Ово можемо записати уз помоћ оператора $\models$ који у контексту провере модела означава чињеницу да неки модел $\mathcal{M}$ задовољава неко својство φ. Формула гласи $\mathcal{M} \models \varphi$. Систем је обично моделован усмереним графом стања који детаљно описује његово понашање. На слици 3.1 може се видети пример моделовања система банкомата, док својство представља формулу

темпоралне логике. Предност апстрактног модела лежи у чињеници да сам систем не мора бити конструисан да би се проверио његов рад, већ се он симулира на моделу. Наравно, модел не може да прикаже све аспекте система и самим тим никада не може у потпуности да га симулира.



Слика 3.1: Пример моделовања система банкомата

Један од начина да се прикажу својства система јесте Хенеси-Милнерова логика [17] (енг. *Hennessy–Milner logic*) која представља врсту темпоралне логике. Она се користи при провери модела да опише својства обележеног система прелазака (енг. *labeled transition system*) који је заправо наш граф модела $\mathcal{M}$. Моћ Хенеси-Милнерове логике лежи у могућности описивања односа између стања и прелаза између њих јер њени централни оператори $\Box$ и $\diamond$ означавају да се након неке акције мора доћи у одређено стање, и да је након неке акције могуће доћи у неко стање.

Дефиниција 3.2.1. (Обележени систем прелазака [26]). Обележени систем прелазака *представља тројку* $\mathcal{T} = (S, \Lambda, \rightarrow)$ *где је:*

- S *скуп стања*
- Λ *скуп лабела*
- $\rightarrow \subseteq S \times \Lambda \times S$ *транзитивна релација којој је додељена лабела из скупа* Λ

Лабеле могу да представљају различите ствари, на пример услов који мора да буде задовољен да би се обавио прелазак из стања у стање или акцију која се извршава у току преласка.

Дефиниција 3.2.2. (Хенеси-Милнерова логика). *Скуп формула* Хенеси-Милнерове логике *дефинише се на следећи начин:*

$$\Phi ::= tt \mid ff \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid [a]\Phi \mid \langle a\rangle\Phi$$

где $a \in \Lambda$ припада скупу лабела Λ и представља прелазе између стања система, док $[a]\Phi$ означава да за сваки прелаз a важи формула Φ, а $\langle a\rangle\Phi$ да за неке прелазе a важи формула Φ.

У примеру 3.2.1 можемо видети пример провере модела система банкомата уз помоћ Хенеси-Милнерове логике.

Пример 3.2.1. *Показаћемо пример коришћења Хенеси-Милнерове логике у проблему провере модела на примеру система банкомата. Модел система дат је на слици 3.1. Рецимо да желимо да проверимо својство "Увек када корисник одабере да жели извештај о стању на рачуну, банкомат мора тај извештај да му одштампа". На почетку реченице стоји реч "увек-што значи да формулу почињемо оператором нужности $\Box$. Ако акцију одабира извештаја означимо са a, а акцију која мора да се деси после ње са $b \in Act$, почетак формуле записујемо са $[a]b$. Формула b у нашем случају означава да мора да се одштампа извештај и за њено записивање користимо конструкцију која се често користи а означава да се у следећем кораку рада система мора десити одређена акција $c \in Act$. Ово записујемо као могућност да се деси акција c и немогућност да се деси било која друга акција, што значи да је $b = (\langle c\rangle$тачно $\wedge\, [Act\backslash c]$нетачно$)$. У нашем случају акцију c представља штампање извештаја. Када спојимо ове две формуле које смо дефинисали добијамо коначну формулу својства која гласи:*

$$[\text{извештај}](\langle\text{штампање извештаја}\rangle\text{тачно} \wedge [Act\backslash\text{штампање извештаја}]\text{нетачно})$$

Провером задате формуле на графу увиђамо да је она тачна јер након што корисник у кораку трансакције одабере извештај једина могућа акција, тј. прелаз у графу система, је штампање извештаја. Што даље значи да систем задовољава дато својство.
Приметимо такође да слично својство не важи и при одабиру подизања новца и уношења жељеног износа јер може да се деси да корисник на рачуну нема толико расположивог новца, у ком случају систем неће извршити исплату.

Сада ћемо проверити својство које систем не задовољава. Формула гласи:

$$\langle\text{неисправан пин код}\rangle\langle\text{извештај} \vee \text{подизање новца}\rangle\text{тачно}$$

Формула почиње дијамантом што означава могућност, дакле чињеница да после акције $a \in Act$ може да се деси акција $b \in Act$ описана је формулом $\langle a \rangle b$. У нашем случају акција a представља уношење неисправног пин кода, док је акција b нова формула. Дањом анализом формуле b видимо да она означава могућност да се деси једна од две акције: или акција штампања извештаја, или акција подизања новца. Када објединимо анализирано, долазимо до закључка да ако је ова формула задовољива то би значило да корисник понекад може, чак и уз неисправан пин код, да добије приступ средствима и стању на рачуну што би имало озбиљне последице по систем. Провером формуле на графу увиђамо да то није могуће и да само у случају уношења исправног пин кода корисник добија приступ рачуну.

3.3 μ-рачун

Модални μ-рачун увео је Козен [28] (енг. *Kozen*) као логику фиксних тачака и он представља проширење Хенеси-Милнерове логике са два оператора фиксних тачака:

- ν - оператор највеће фиксне тачке
- μ - оператор најмање фиксне тачке

Нови оператори уведени су да би постојала могућност да се изрази рекурзија. Уз њихову помоћ можемо се кретати кроз систем прелазака, тачније рекурзивно ићи од чвора до чвора, док не дођемо до фиксне тачке.

Модални μ-рачун је веома експресиван и моћан али за његово коришћење потребно је искуство. Да би се лакше разумели и користили у пракси, нови оператори ν и μ често се сматрају да означавају сигурности (енг. *safety*) и живост (енг. *liveness*). Такође, оператори се могу посматрати и као уопштење класичних квантификатора $\forall$ и $\exists$. Ове операторе можемо дефинисати на следећи начин:

- $\nu(\varphi)$ означава сигурност, у датом извршавању формула φ је свуда задовољива, уопштење универзалног квантификатора $\forall$ јер је формула увек задовољива
- $\mu(\varphi)$ означава живост, свако извршавање ће неизбежно доћи до стања у ком је формула φ задовољива, уопштење егзистенцијалног квантификатора $\exists$ јер постоји стање у ком је формула задовољива

Дефиниција 3.3.1. (Фиксна тачка). *Теорија фиксних тачака [27] (енг. fixed point theory) дефинише* фиксну тачку *функције као тачку коју функција*

пресликава на саму себе. То значи да ако је x фиксна тачка функције f онда је f(x) = x. Најмања фиксна тачка *има најмању вредност од свих фиксних тачака, док* највећа фиксна тачка *има највећу.* Префиксна тачка *је свака тачка x за коју важи f(x) ≤ x, док је постфиксна она за коју важи x ≤ f(x).*

Дефиниција 3.3.2. (Модални μ-рачун). Модални μ-рачун *гради се над скупом формула:*

$$\varphi ::= tt \mid ff \mid X \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid [a]\varphi \mid \langle a\rangle\varphi \mid \nu X.\varphi \mid \mu X.\varphi$$

где $X \in Var$ представља променљиву из пребројивог скупа променљивих.

Као што смо видели раније у случају Хенеси-Милнерове логике, као модел на коме се примењују логичке формуле коришћен је систем прелазака, слично је и са модалним μ-рачуном само што се овог пута користи *необележени систем.*

Дефиниција 3.3.3. (Необележени систем прелазака [26]). Необележени систем прелазака *представља пар $\mathcal{T} = (S, \rightarrow)$ где је:*

- *S скуп стања*
- *$\rightarrow \subseteq S \times S$ транзитивна релација*

Дакле, систем прелазака није ништа више него усмерени граф.

Дефиниција 3.3.4. (Валуација у модалном μ-рачуну). *Валуација модалног μ-рачуна $\mathcal{V}$ представља пресликавање $\mathcal{V} \rightarrow 2^S$ које свакој променљивој X додељује скуп стања, а валуација $\mathcal{V}[\mathcal{X} := T]$, где је $T \subseteq S$, иста је као валуација $\mathcal{V}$ сем што за променљиву X важи $\mathcal{V}[\mathcal{X} := T](X) = T$.*

Ако је $\mathcal{T} = (S, \rightarrow)$ систем прелазака, $\mathcal{V}$ валуација, а φ формула μ-рачуна онда кажемо да је формула φ задовољива у стању s система $\mathcal{T}$ при валуацији $\mathcal{V}$ (обележава се са $(\mathcal{T}, s) \models_{\mathcal{V}} \varphi$) ако је $s \in [\![\varphi]\!]^{\mathcal{T}}_{\mathcal{V}}$.

Као што смо видели у дефиницији најмање и највеће фиксне тачке 3.3.1 између фиксних тачака постоји релација поретка помоћу које можемо да поредимо тачке и, сходно томе, утврдимо која је најмања, а која највећа. Нама је сада потребно да дефинишемо ту релацију поретка за скупове, јер, као што смо видели у дефиницији 3.3.4, променљиве модалног μ-рачуна које се валуирају представљају скупове стања. Као релацију поретка узећемо операцију инклузије скупова.

Дефиниција 3.3.5. (Инклузија скупова) Кажемо да је скуп A подскуп скупа B, у ознаци $A \subseteq B$, ако је сваки елемент скупа A уједно и елемент

скупа B. За скуп A кажемо да је прави подскуп скупа B, у ознаци $A \subset B$ ако је $A \subseteq B$ и $A \neq B$. Бинарну релацију $\subseteq$ зовемо и инклузијом, а бинарну релацију $\subset$ зовемо и строгом инклузијом.

Дакле, за сваки скуп S, релација инклузије је парцијално уређење на скупу 2^S свих подскупова од S, што се слаже са нашом дефиницијом 3.3.4 где валуација пресликава променљиве у скупове стања ($\mathcal{V} \to 2^S$).

Дефиниција 3.3.6. (Семантика модалног μ-рачуна). Семантика модалног μ-рачуна *дефинише се на следећи начин:*

- $[\![tt]\!]^\tau_\mathcal{V} = S$
- $[\![\mathrm{ff}]\!]^\tau_\mathcal{V} = \emptyset$
- $[\![X]\!]^\tau_\mathcal{V} = \mathcal{V}(X)$
- $[\![\varphi_1 \wedge \varphi_2]\!]^\tau_\mathcal{V} = [\![\varphi_1]\!]^\tau_\mathcal{V} \cap [\![\varphi_2]\!]^\tau_\mathcal{V}$
- $[\![\varphi_1 \vee \varphi_2]\!]^\tau_\mathcal{V} = [\![\varphi_1]\!]^\tau_\mathcal{V} \cup [\![\varphi_2]\!]^\tau_\mathcal{V}$
- $[\![[a]\varphi]\!]^\tau_\mathcal{V} = \{s \in S \mid \forall_{t \in S, a \in \{a\}} s \xrightarrow{\mathrm{a}} t \Rightarrow t \in [\![\varphi]\!]^\tau_\mathcal{V}\}$
- $[\![\langle a\rangle\varphi]\!]^\tau_\mathcal{V} = \{s \in S \mid \exists_{t \in S, a \in \{a\}} s \xrightarrow{\mathrm{a}} t \wedge t \in [\![\varphi]\!]^\tau_\mathcal{V}\}$
- $[\![\nu X.\varphi(X)]\!]^\tau_\mathcal{V} = \bigcup \{T \subseteq S \mid T \subseteq [\![\varphi]\!]^\tau_{\mathcal{V}[\mathcal{X}:=T]}\}$
- $[\![\mu X.\varphi(X)]\!]^\tau_\mathcal{V} = \bigcap \{T \subseteq S \mid [\![\varphi]\!]^\tau_{\mathcal{V}[\mathcal{X}:=T]} \subseteq T\}$

3.3.1 Основни примери

Након што смо видели како се дефинише и шта представља модални μ-рачун, време је да видимо како он може да се користи кроз примере. Примери који следе преузети су из књиге *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)* [4].

Пример 3.3.1. *Ако желимо да опишемо чињеницу да нека формула ϕ важи увек у неком систему прелаза $\mathcal{T}$ то можемо урадити на следећи начин преко својства X:*

$$X \Rightarrow \phi \wedge [-]X$$

што значи да ако је X тачно онда је тачна и формула ϕ, и у које год даље стање да одемо, преко произвољног прелаза, X ће да важи и у том стању ([-]X). Ако би смо ову формулу желели да опишемо преко модалног μ-рачуна то би смо урадили преко фиксне тачке јер формула ϕ зависи од својства X (када је X тачно, тачно је и ϕ), што се уклапа са својством фиксне тачке да је $\phi(X) = X$. Питање које остаје јесте да ли ћемо да узмемо најмању или највећу фиксну тачку. Будући да тражимо формулу која важи увек у

систему узећемо највећу фиксну тачку јер нам она враћа највећи могући број стања система који задовољавају својство фиксне тачке. Коначна формула гласи:

$$\nu X.\phi \wedge [-]X$$

Пример 3.3.2. *Мало компликснији пример је описивање формуле која означава да једна формула важи све док друга формула не важи. Значи желимо да опишемо да формула P важи све док не важи формула Q, то можемо урадити следећом формулом:*

$$\nu Z.Q \vee (P \wedge [a]Z)$$

која означава да све док Q није тачно тачно је P и у које год стање даље да одемо преко прелаза a цела формула је тачна $(P \wedge [a]Z)$. Тако ћемо се кретати кроз систем преко прелаза a или заувек у бесконачној петљи, или док не дођемо до стања у коме важи формула Q.

Пример 3.3.3. *Супротно прошлим примерима, чињеницу да формула P важи након неке акције, тј. потребна је живост система да се деси нека акција да би формула била задовољива, можемо записати уз помоћ оператора μ. Ова формула гласи:*

$$\mu Z.P \vee [a]Z$$

што значи да смо или дошли до стања у коме је формула P задовољива (P) или кретањем низ прелаз a долазимо до стања у коме је цела формула $(\mu Z.P \vee [a]Z)$ задовољива. Ово укратко значи да смо или у стању у коме је P задовољиво или ћемо се кретањем низ прелазе a доћи до таквог стања.

Пример 3.3.4. *Овај пример сличан је примеру 3.3.2 с тим што сада на почетку формуле имамо оператор најмање фиксне тачке, уместо највеће:*

$$\mu Z.Q \vee (P \wedge [a]Z)$$

формула има исто значење као и формула из примера 3.3.2 с тим што у овом случају не можемо да уђемо у бесконачну петљу и формула Q мора пре или касније да буде задовољива у неком стању система.

Пример 3.3.5. *Сада ћемо видети како формуле модалног μ-рачуна могу да се користе при моделовању система. У примеру можемо видети моделовање*

једноī концепīта коју се чесīто јавља у рачунарским сисīемима. Рецимо да желимо да запишемо формулу преко које проверавамо да ли у сисīему може да дође до засīоја (енī. deadlock). Појам засīоја јавља се у конкуренīном раду и њеīова елиминација предсīавља важан проблем у рачунарсīву. Формула мора да испуни услов да не посīоје сīања без излазних īрана. У синīакси модалноī μ-рачуна īа формула īласи:

$$\nu Z.\left(\bigvee_{a\in A}\langle a\rangle\top \wedge \bigwedge_{a\in A}[a]Z\right)$$

и означава да мора да посīоји бар једна излазна īрана из сīања ($\bigvee_{a\in A}\langle a\rangle\top$) и да када пређемо из īренуīноī сīања низ било коју од прелаза а долазимо до сīања у коме је формула īакође задовољива ($\bigwedge_{a\in A}[a]Z$).

3.3.2 Алгоритам за проверу задовољивости формула

Кључно питање које се природно намеће након дефинисања синтаксе и семантике модалног μ-рачуна јесте како проверити која стања система прелазака задовољавају неку формулу, и да ли је могуће конструисати алгоритам за такву проверу. Као што смо раније рекли, најмања и највећа фиксна тачка омогућавају нам да се рекурзивно крећемо кроз систем прелазака и ту чињеницу можемо искористити при конструкцији нашег алгоритма. Оно што нам је потребно да проверимо јесте да ли је формула свуда задовољива, што постижемо преко највеће фиксне тачке, или да ли постоји стање у систему у коме је формула задовољива, што постижемо преко најмање фиксне тачке. Оба проблема можемо да решимо коришћењем рекурзије. Провера да ли сва стања система задовољавају формулу захтева да рекурзивно прођемо кроз сва стања, обилазећи стања у која можемо тренутно да пређемо у сваком кораку, а затим понављајући тај поступак док не обиђемо сва стања система. Слично томе, ако желимо да проверимо да ли постоји стање у ком је формула задовољива морамо у сличном маниру рекурзивно да обиђемо скуп стања и да утврдимо да ли такво стање постоји.

Питање које нам сада остаје јесте одакле да кренемо, тј. из којих стања да започнемо нашу претрагу. Највећу фиксну тачку разматрамо јер нам омогућава да кренемо из било којег стања у систему, тј. из целог скупа S, и да се из њих даље рекурзивно крећемо, од стања до стања, кроз систем преко прелаза из задате формуле. Када дођемо до фиксне тачке, тј. не постоје нова стања у која можемо да пређемо а која задовољавају задату формулу,

онда стајемо. Али шта ће се десити ако неко од стања из ког кренемо нема ниједан прелаз? Испоставља се да нам није довољно да кренемо из свих стања система преко највеће фиксне тачке јер тако обилазимо само стања која имају прелазе у друга стања, али не и она која немају ниједан излазни прелаз, због тога поред највеће разматрамо и најмању фиксну тачку. Најмања фиксна тачка омогућава нам да кренемо од стања која немају излазне прелазе, тј. скуп стања у која може из њих да се пређе је празан. Разматрањем обе фиксне тачке покривамо све могуће случајеве рекурзивног проласка кроз скуп стања.

Један од начина да конструишемо алгоритам јесте преко функционала као што је приказано у раду Кеирена [25] (енг. *Keiren*). Оператори ν и μ могу се дефинисати са:

- $[\![\nu X.\varphi]\!]^{\tau}_{\mathcal{V}} = \nu T \subseteq S.\Phi_{\mathcal{V}}(T)$
- $[\![\mu X.\varphi]\!]^{\tau}_{\mathcal{V}} = \mu T \subseteq S.\Phi_{\mathcal{V}}(T)$

где је $\Phi_{\mathcal{V}} : 2^S \to 2^S$ функционал дефинисан са $\Phi_{\mathcal{V}}(T) = [\![\varphi]\!]^{\tau}_{\mathcal{V}[X:=T]}$, νT означава највећу, а μT најмању фиксну тачку.

Уз помоћ нових дефиниција оператора пробаћемо да дефинишемо итеративни алгоритам за израчунавање стања у којима је нека формула φ задовољива. Нека је $\Phi^0(T) = T$ и $\Phi^{n+1}(T) = \Phi(\Phi^n(T))$, будући да је Φ монотона функција и S је коначан скуп стања можемо изменити дефиниције оператора ν и μ:

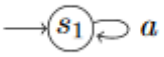
- $[\![\nu X.\varphi]\!]^{\tau}_{\mathcal{V}} = \bigcap \Phi^i_{\mathcal{V}}(S)$
- $[\![\mu X.\varphi]\!]^{\tau}_{\mathcal{V}} = \bigcup \Phi^i_{\mathcal{V}}(\emptyset)$

Ове две формуле представљају наше фиксне тачке, прва највећу, а друга најмању. Алгоритам почињемо од Φ^0 и извршавамо га кроз итерације све док се извршавање не стабилизује, прво за најмању, а затим и за највећу фиксну тачку. Почињемо од $i = 0$ а као улаз за најмању фиксну тачку узимамо празан скуп $\emptyset$. Након тога израчунавамо скуп стања у којима важи формула Φ^0, применом формула које смо дефинисали у семантици модалног рачуна. Затим у сваком наредном кораку алгоритма инкрементирамо i за 1 и израчунавамо скупове стања одговарајућих формула $(\Phi^1, \Phi^2...)$. Када излази две узастопне итерације буду исти, као излаз смо добили скуп стања система прелазака која задовољавају задату формулу φ. Након што смо завршили израчунавање за најмању фиксну тачку исти поступак примењујемо и на највећу фиксну тачку, с тим што у њеном случају улаз прве итерације представља цео скуп S. На овај начин дефинисали смо итеративни алгоритам за проверу формула модалног μ-рачуна.

Дефиниција 3.3.7. (a-путања). a-путања *у систему прелаза* $\mathcal{T}$, *где је* $a \in Act$ *прелаз између стања система, представља путању из једног стања система* $s_1 \in S$, *низ прелаз* a, *до другог стања система* $s_2 \in S$.

У наставку ће бити приказана два примера провере задовољивости формуле μ-рачуна. Примери су преузети из рада *Modal μ-calculus* [25].

Пример 3.3.6. *Систем прелазака* $\mathcal{T}$ *задат је на слици 3.2, проверићемо да ли у њему важе формуле* $\mu X.\langle a\rangle X$ *и* $\nu X.\langle a\rangle X$.



Слика 3.2: Систем прелазака 1

Почећемо са формулом $\mu X.\langle a\rangle X$. *Рецимо да је* Φ *функционал и да је* $\Phi(T) = [\![\langle a\rangle X]\!]^{\tau}_{[X:=T]}$. *Сада треба да пронађемо најмању фиксну тачку функционала* Φ *и то ћемо урадити уз помоћ итеративног алгоритма који смо раније дефинисали. Будући да израчунавамо оператор* μ *полазимо од празног скупа и пролазимо кроз итерације док не стигнемо до најмање фиксне тачке:*

$$\begin{aligned}
\Phi^0(\emptyset) &= \emptyset \\
\Phi^1(\emptyset) &= [\![\langle a\rangle X]\!]^{\tau}_{[X:=\emptyset]} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \ \wedge \ t \in [\![X]\!]^{\tau}_{[X:=\emptyset]}\} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \ \wedge \ t \in [X := \emptyset](X)\} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \ \wedge \ t \in \emptyset\} \\
&= \emptyset
\end{aligned}$$

Нашли смо фиксну тачку одмах у првом кораку алгоритма и видимо да је скуп стања која задовољавају почетну формулу празан.

Сада ћемо проверити формулу $\nu X.\langle a\rangle X$. *Поступак је сличан као и у претходном случају с тим што у случају оператора* ν *почињемо од целог скупа стања* S *и пролазимо кроз итерације док не дођемо до највеће фиксне тачке:*

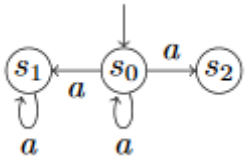
$$\begin{aligned}
\Phi^0(S) &= S \\
\Phi^1(S) &= [\![\langle a\rangle X]\!]^{[X:=S]}_{L} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \ \wedge \ t \in [\![X]\!]^{[X:=S]}_{L}\} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \ \wedge \ t \in [X := S](X)\} \\
&= \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \ \wedge \ t \in S\} \\
&= S
\end{aligned}$$

Опет смо нашли фиксну тачку одмах у првом кораку алгоритма и видимо да сва стања у систему задовољавају почетну формулу.

Пример 3.3.7. *За комплекснији пример узећемо систем прелазака $\mathcal{T}$ задат на слици 3.3. Треба проверити да ли постоји путања кроз систем која ће нас довести до стања у коме прелаз a не постоји, тј. да ли постоји стање које задовољава услов недоступности прелаза a. Ово се може проверити формулом:*

$$[a]false \vee \langle true \rangle X$$

која означава да смо или дошли до жељеног стања, тј. стања у коме не постоји излазни прелаз a ([a]false) или идемо даље кроз систем преко произвољног прелаза $\langle true \rangle$ до следећег стања X. Након што је формула одређена само је потребно видети да ли услов недоступности прелаза a задовољава највећа или најмања фиксна тачка.



Слика 3.3: Систем прелазака 2

Почећемо са најмањом фиксном тачком. Функционал Φ задат је са $\Phi(T) = [\![[a]false \vee \langle true \rangle X]\!]^{\tau}_{[X:=T]}$. Слично прошлом примеру почињемо од празног скупа и извршавамо алгоритам итеративно:

$$\begin{aligned}
\Phi^0(\emptyset) &= \emptyset \\
\Phi^1(\emptyset) &= [\![[a]false \vee \langle true \rangle X]\!]^{\tau}_{[X:=\emptyset]} \\
&= [\![[a]false]\!]^{\tau}_{[X:=\emptyset]} \cup [\![\langle true \rangle X]\!]^{\tau}_{[X:=\emptyset]} \\
&= \{s \in S \mid \forall_{t \in S, a \in \{a\}} s \xrightarrow{a} t \Rightarrow t \in \emptyset\} \\
&\quad \cup \{s \in S \mid \exists_{t \in S, a \in \{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]^{\tau}_{[X:=\emptyset]}\} \\
&= \{s_2\} \cup \emptyset \\
&= \{s_2\} \\
\Phi^2(\emptyset) &= [\![[a]false \vee \langle true \rangle X]\!]^{\tau}_{[X:=\{s_2\}]} \\
&= [\![[a]false]\!]^{\tau}_{[X:=\{s_2\}]} \cup [\![\langle true \rangle X]\!]^{\tau}_{[X:=\{s_2\}]} \\
&= \{s_2\} \cup \{s \in S \mid \exists_{t \in S, a \in \{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]^{\tau}_{X:=\{s_2\}}\} \\
&= \{s_2\} \cup \{s_0\} \\
&= \{s_0, s_2\} \\
\Phi^3(\emptyset) &= [\![[a]false \vee \langle true \rangle X]\!]^{\tau}_{[X:=\{s_0,s_2\}]} \\
&= [\![[a]false]\!]^{\tau}_{[X:=\{s_0,s_2\}]} \cup [\![\langle true \rangle X]\!]^{\tau}_{[X:=\{s_0,s_2\}]} \\
&= \{s_2\} \cup \{s \in S \mid \exists_{t \in S, a \in \{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]^{\tau}_{[X:=\{s_0,s_2\}]}\} \\
&= \{s_2\} \cup \{s_0\}
\end{aligned}$$

$$= \{s_0, s_2\}$$

Нашли смо фиксну тачку након 3 итерације алгоритма и видимо да су стања која задовољавају почетну формулу s_0 *и* s_2.

Сада ћемо израчунати највећу фиксну тачку:

$$\Phi^0(S) = S$$
$$\Phi^1(S) = [\![[a]false \vee \langle true\rangle X]\!]^{\tau}_{[X:=S]}$$
$$= [\![[a]false]\!]^{\tau}_{[X:=S]} \cup [\![\langle true\rangle X]\!]^{\tau}_{[X:=S]}$$
$$= \{s_2\} \cup \{s \in S \mid \exists_{t\in S, a\in\{a\}} s \xrightarrow{a} t \wedge t \in [\![X]\!]^{\tau}_{[X:=S]}\}$$
$$= \{s_2\} \cup \{s_0, s_1\}$$
$$= S$$

Нашли смо фиксну тачку одмах у првом кораку алгоритма и видимо да сва стања у систему задовољавају почетну формулу.

Након што смо нашли најмању и највећу фиксну тачку питање је која нам од њих више одговара. Услов недоступности прелаза a који треба да задовољимо испуњен је у стању s_2*, док је стање* s_1 *непожељно јер му једини излазни прелаз води у самог себе преко a. Тако да, будући да нам стање* s_1 *не одговара, бирамо најмању фиксну тачку која садржи стања* s_0 *и* s_2.

У прошлим примерима могли смо да видимо како се користи итеративни алгоритам за проверу формула, а сада ћемо погледати неке примере формула μ-рачуна које се често јављају у пракси. Примери су преузети су из рада *Modal μ-calculus* [25].

3.4 Веза са играма парности

Везу између игара парности и модалног μ-рачуна први су открили Емерсон (енг. *Emerson*) и Јутла (енг. *Jutla*) [11] 1991. године. У њиховом раду приказан је начин да се проблем решавања игара парности сведе на проблем провере модела модалног μ-рачуна. Нека је $\mathcal{G} = (V, E, \alpha)$ игра парности, а функција приоритета узима вредности α(V) = {1,...,n} док је највећи приоритет n паран (да је непаран формула би почела са $\mu Z_n.\nu Z_{n-1}...$). Ако граф $G = (V, E)$ заменимо системом прелазака $\mathcal{T}$ имамо формулу:

$$\varphi = \nu Z_n.\mu Z_{n-1}...\mu Z_1.\Big(\bigvee_{i\leq n}(Y_0 \wedge X_i \wedge \langle -\rangle Z_i) \vee \bigvee_{i\leq n}(Y_1 \wedge X_i \wedge [-]Z_i)\Big) \tag{3.1}$$

На крају, валуација $\mathcal{V}$ задовољава:

- $\mathcal{V}(X_i) = \{v \in V \mid \alpha(v) = i\}$

- $\mathcal{V}(Y_0) = V_0$
- $\mathcal{V}(Y_1) = V_1$

Теорема 3.4.1. *Нека је $\mathcal{G} = (V, E, \alpha)$ игра парности, $\mathcal{T}$ систем прелазака, $\mathcal{V}$ валуација, а φ формула μ-рачуна која је задата изнад. Онда*

$$v \in W_0(\mathcal{G}) \iff (\mathcal{T}, v) \models_{\mathcal{V}} \varphi$$

Дакле, проблем провере да ли се неки чвор v налази у победничком региону неког играча и проблем провере да ли је формула μ-рачуна задовољива у задатом систему прелазака и валуацији, из истог почетног чвора v, су еквивалентни.

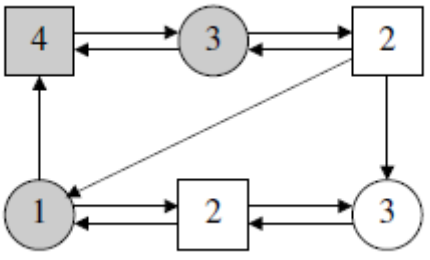
Да би се илустровала веза између игара парности и модалног μ-рачуна, тј. његове формуле коју су дефинисали Емерсон и Јутла 3.1, у наставку ће бити приказана детаљна анализа те формуле кроз пример.

Пример 3.4.1. *Нека је $\mathcal{G} = (V, E, \alpha)$ игра парности задата графом приказаним на слици 3.4, α функција приоритета која узима вредности $\alpha(V) = \{1,\ldots,n\}$, код које је највећи приоритет n паран, Z_i, $i \in \{1,\ldots,n\}$, променљиве придружене одговарајућим приоритетима, а систем прелазака τ граф игре парности $\mathcal{G}$. Тада формула μ-рачуна која одговара тој игри гласи:*

$$\varphi = \nu Z_4.\mu Z_3.\nu Z_2.\mu Z_1.\left(\bigvee_{i\leq n}(Y_0 \wedge X_i \wedge \langle - \rangle Z_i) \vee \bigvee_{i\leq n}(Y_1 \wedge X_i \wedge [-] Z_i)\right)$$

где валуација $\mathcal{V}$ задовољава:

- $\mathcal{V}(X_i) = \{v \in V \mid \alpha(v) = i\}$
- $\mathcal{V}(Y_0) = V_0$
- $\mathcal{V}(Y_1) = V_1$



Слика 3.4: Пример графа игре парности

Сада ћемо се бавити детаљнијом анализом приказане формуле. Уочавамо да се састоји од две потформуле везане дисјункцијом. Прва

потформула важи када се налазимо у стању система које припада играчу P_0, док друга важи ако тренутно стање система припада играчу P_1. Прва потформула гласи:

$$\bigvee_{i \leq n} (Y_0 \wedge X_i \wedge \langle - \rangle Z_i)$$

Y_0 означава скуп стања која припадају играчу P_0, X_i означава скуп стања која имају приоритет i, док $\langle - \rangle Z_i$ означава да само у неким стањима у која можемо да пређемо из тренутног стања важи формула Z_i. Овде можемо увидети паралелу са играма парности јер када проверавамо да ли играч P_0 има победничку стратегију из неког почетног чвора, за чворове који припадају играчу P_0 гледамо било које излазне гране, тј. оне које су у складу са његовом стратегијом.

Друга потформула гласи:

$$\bigvee_{i \leq n} (Y_1 \wedge X_i \wedge [-] Z_i)$$

Y_1 означава скуп стања која припадају играчу P_1, X_i означава скуп стања која имају приоритет i, док $[-] Z_i$ означава да у свим стањима у која можемо да пређемо из тренутног стања важи формула Z_i. Овде паралелу са играма парности представља чињеница да за другог играча, тј. играча P_1, морамо да разматрамо све гране, јер не знамо по којој он стратегији игра и због тога морамо размотрити све његове могућности.

Оно што даље можемо да приметимо јесте да су променљиве Z_i које имају парне приоритете везане оператором највеће фиксне тачке, док су оне непарних приоритета везане оператором најмање фиксне тачке. Највећа фиксна тачка, у складу са својом одликом сигурности, означава да за стања парних приоритета важи да ако формула важи у неким (у случају играча P_0) или свим (у случају играча P_1) стањима у која може да се пређе из тренутног стања, онда формула важи и у тренутном стању у ком се налазимо. Аналогно, најмања фиксна тачка, у складу са својом одликом живости, означава да за стања непарних приоритета важи да ако формула пре или касније важи у неким (у случају играча P_0) или свим (у случају играча P_1) стањима у која може да се дође полазећи из тренутног стања и крећући се кроз систем прелазака, онда формула важи и у тренутном стању у ком се налазимо.

Након што смо дефинисали све правилности које се јављају у формули 3.1 можемо их проверити на нашем графу игре задатом на слици 3.4.

Стања означена кругом припадају играчу P_0, док стања означена квадратом припадају играчу P_1. Осенчена стања означавају победнички регион играча P_0. Почетно стање је горње лево стање приоритета 4, и њега ћемо прво размотрити. То је стање парног приоритета које припада играчу P_1. То значи да посматрамо највећу фиксну тачку и оператор $[-]$. Формула важи у тренутном стању ако важи у сваком стању у које може из њега да се пређе. У стању десно од њега, приоритета 3, формула важи ако ће се преко било ког прелаза кад тад доћи до стања у коме формула важи. У стању десно од њега, приоритета 2, формула важи ако важи у сваком стању у које може из њега да се пређе. Аналогно можемо проверити сва стања система, чиме добијамо скуп стања у којима важи формула φ, који је уједино и победнички регион играча P_0.

4. Стратегије решавања

Важан аспект игара парности представља временска сложеност одређивање њиховог победника, под условом да оба играча користе оптималне стратегије. Један од првих алгоритама прилагођених играма парности представио је Зиелонка [50] (енг. *Zielonka*) 1998. године, алгоритам је био експоненцијалне сложености и биће први представљен у наставку поглавља.

4.1 Досадашњи радови

Након појаве првих алгоритама уследила су детаљнија истраживања и покушаји да се утврди да ли су игре парности решиве у полиномском времену. Јурџински [21] је 1998. доказао да игре парности припадају класи UP∩coUP што је сугерисало да вероватно постоји алгоритам бржи од експоненцијалног. Та претпоставка се испоставила као тачна јер је ускоро осмишљено више алгоритама субекспоненцијалне сложености. Један од првих таквих јесте случајни алгоритам Петерсона (енг. *Petersson*) и Воробјова (енг. *Vorobyov*) [37], а нешто касније и детерминистички алгоритам Јурџинског, Патерсона и Цвика [23] сложености $n^{O(\sqrt{n})}$.

Питање које се затим јавило јесте да ли је могуће осмислити алгоритам који ће бити ефикасан у практичном решавању игара. Ови алгоритми претежно су били засновани на хеуристикама [2, 22, 39]. У свом раду Фридман (енг. *Friedmann*) и Ланге (енг. *Lange*) [14] доказали су да је у пракси ипак и даље био најефикаснији првобитни Зиелонкин алгоритам.

Још једна идеја која је имплементирана при изради алгоритама јесте увођење параметра m, који представља број различитих приоритета које чворови могу да имају. Мекнаутон (енг. *McNaughton*) је у свом раду [33] из 1993. године показао да игре парности могу да се реше у времену $n^{m+O(1)}$. Касније унапређења дала су сложеност $n^{m/2+O(1)}$ [42], а Шеве (енг. *Schewe*) је у свом раду [41] спустио сложеност на $n^{m/3+O(1)}$ и утврдио да је употреба

параметра m веома важна у анализи временске сложености решавања игара парности, и да је чак природна, као и да је параметар m у највећем броју случаја експоненцијално мањи од n.

Дефиниција 4.1.1. (ω-аутомат). *ω-аутомати представљају варијанту коначних аутомата која уместо коначних ниски прима као улаз бесконачне ниске. Због чињенице да прихватају бесконачан улаз ω-аутомати морају да дефинишу услове прихватања (енг. acceptance conditions) уместо прихватљивих завршних стања.*

Дефиниција 4.1.2. (Буки аутомат). *Буки аутомати (енг. Büchi automaton) представљају потклasu ω-аутомата у којима је услов прихватања дефинисан над потскупом F аутомата G. Услов прихватања је задовољен за све улазне речи ρ за које је $Inf(p) \cap F$ непразно тј. постоји прихватљиво стање које се појављује бесконачно много пута у ρ.*

Дефиниција 4.1.3. (Аутомат парности). *Аутомати парности представљају још једну потклasu ω-аутомата. У аутоматима парности услов прихватања за улазну реч ρ је да је најмањи број који се јавља у бесконачно проласку речи ρ кроз аутомат паран.*

Игре парности могу бити примењене и у области реактивне синтезе (енг. *reactive synthesis*) и то тако што се формуле темпоралне логике преводе у Буки аутомат који се затим детерминизује конверзијом у аутомат парности. Нир Питерман [38] (енг. *Nir Piterman*) је затим доказао да Буки аутомат са n стања може да се конвертује у аутомат парности са $2 \cdot n^n \cdot n!$ стања. Нешто раније, Томас Вилке [48] (енг. *Thomas Wilke*) је доказао да проблем налажења услова прихватања аутомата парности може да се сведе на проблем проналаска победничке стратегије у игри парности, чијим решавањем добијамо решење почетног проблема.

Ди Стасио (енг. *Di Stasio*), Мурано (енг. *Murano*), Перели (енг. *Perelli*) и Варди (енг. *Vardi*) [9] повезали су приступе коришћења аутомата и параметра m и извршили су разне експерименте у којима је $m = \log(n)$. Овај приступ побољшао је временску сложеност на $O(n^{\lceil \log(m) \rceil + 6})$ што значи квазиполиномску границу извршавања јер је увек могуће изабрати m тако да је $m \leq n$ будући да број различитих приоритета не може бити већи од броја чворова, јер сваки чвор има тачно један приоритет. Штавише, ако је $m < \log(n)$, временска сложеност је $O(n^5)$ што представља фиксну и најнижу, сложеност када је игра параметризована са m.

Каснија истраживања [13, 15, 20] спустила су временску сложеност на $O(mn^{2.38})$ (ова сложеност добијена је у раду Јурџинског и Лазића) док су у другим радовима добијени слични резултати у зависности од претпоставки које су примењене у вези игара. Радови [20] [13] истичу се због чињенице да поред квазиполиномске временске сложености могу да се модификују тако да поседују квазилинеарну просторну сложеност у односу на број стања n.

4.2 Експоненцијални алгоритам

Први алгоритам за решавање игара парности који ћемо размотрити је интуитивни експоненцијални алгоритам. Алгоритам може да се користи и у другим проблемима а за игре парности први га је представио Зиелонка [50] 1998. године. Заснован је на доказу детерминистичности без памћења (енг. *memoryless determinacy*). На слици 1 дат је преудокод алгоритма.

Теорема 4.2.1. *Нека је $\mathcal{G}$ игра парности, онда је излаз алгоритма реши($\mathcal{G}$) = ($W_0(\mathcal{G})$, $W_1(\mathcal{G})$), где је $W_0(\mathcal{G})$ победнички регион првог, а $W_1(\mathcal{G})$ победнички регион другог играча. Временска сложеност алгоритма је $2^{O(n)}$, где је $n = |V(\mathcal{G})|$.*

Доказ. Доказ се конструише индукцијом по број чворова $|\mathrm{V}(\mathcal{G})|$. Базу индукције представља празан скуп $\mathrm{V}(\mathcal{G})| = \emptyset$ за који теорема очигледно важи. Ако у скупу постоји само један чвор он ће припадати победничком региону играча P_0 ако је приоритет чвора паран, иначе ће припадати победничком региону играча P_1. Иначе, почињемо од скупа $V^p = Y \subseteq V(G)$ и без губитка генералности можемо претпоставити да је p парно. Индуктивни корак представља рачунање решења (W_0, W_1) за подигру $\mathcal{G}' = \mathcal{G} \backslash F_0(Y)$. На основу индуктивне хипотезе W_0 и W_1 су победнички региони играча P_0 и P_1 у подигри $\mathcal{G}'$. □

Након почетних иницијализација, скуп A је принудни скуп играча P_i коме одговара парност највећег приоритета у $\mathcal{G}$. Дакле, играч P_i може да осигура победу у свакој бесконачној игри која посећује скуп A. Подигра $\mathcal{G}'$ настаје када се из игре $\mathcal{G}$ уклони скуп A, а њена решења представљају W_0, победнички регион за играча P_0, и W_1, победнички регион за играча P_1.

Ако је $W_i' = \mathrm{V}$, то значи да је скуп W_{1-i}' празан и да играч P_{1-i} може из скупа W_i, у коме побеђује играч P_i, да побегне само у скуп A, у коме такође побеђује други играч. То значи да у игри побеђује играч P_i.

Алгоритам 1 Експоненцијални алгоритам

```
1:  procedure реши(G)
2:      p := највећи приоритет у G
3:      if p = 0 then
4:          return W_0, W_1 := V, {}
5:      else
6:          Y := чворови у G са приоритетом p
7:          i := p mod 2
8:          A := F_i(Y)
9:          W'_0, W'_1 := реши(G\A)
10:         if W'_i = V then
11:             return W_i, W_{1-i} := V, {}
12:         B := F_{1-i}(W'_{1-i})
13:         W''_0, W''_1 := реши(G\B)
14:             return W_i, W_{1-i} := W''_i, W''_{1-i} ∪B
```

Иначе, ако скуп W_{1-i} није празан то значи да у игри може да победи само играч P_{1-i} јер играч P_i не може да побегне из скупа W_{1-i} у скуп A. Затим рачунамо принудни скуп B за играча P_{1-i} и рачунамо нову подигру $\mathcal{G}''$ која се добија избацивањем скупа B из игре $\mathcal{G}'$. Нову подигру решавамо рекурзивно и из добијених решења W_i'' и W_{1-i}'' добијамо да је W_i'' победнички регион за играча P_i, а $W_{1-i}'' \cup B$ за играча P_{1-i}.

Што се тиче сложености, у алгоритму имамо два рекурзивна позива, оба пута на мањој игри што нам даје $2 \cdot T(n-1)$, где је n број чворова у графу. Остатак алгоритма ограничен је бројем грана у графу јер је толико времена потребно да се израчуна принудни скуп, то је $O(n^2)$. Укупно време извршавања описано је формулом $T(n) \leq 2 \cdot T(n-1) + O(n^2)$, па важи $T(n) \in 2^{O(n)}$.

4.3 Мале мере напредовања

Убрзо након Зиелонкиног алгоритма, Јурџински [22] је 2000. године развио алгоритам заснован на мерама напредовања који представља један од најзначајнијих и најчешће коришћених, алгоритама за решавање игара парности. Идеја алгоритма је придруживање мере напредовања (енг. *progress measures*) сваком од чворова у графу, која би прецизније одредила да ли је чвор добар или лош за неког од играча, тј. колико је тај чвор близу задовољавања одређеног услова. Мера напредовања ажурира се након сваког потеза у игри и то тако да осигурава приближавање задатом услову.

Представљање алгоритма почећемо дефинисањем потребних термина. Нека је $\alpha \in \mathbb{N}^d$ d-торка ненегативних целих бројева за коју важи:

- $\alpha = (\alpha_0, \alpha_1, ..., \alpha_{d-1})$
- $<, \leq, =, \neq, \geq, >$ на d-торкама означавају лексикографско проширење одговарајућих релација над целим бројевима
- $(n_0, n_1, ..., n_k) \equiv_i (m_0, m_1, ..., m_l)$ акко $(n_0, n_1, ..., n_i) \equiv (m_0, m_1, ..., m_i)$, где је $\equiv \in \{<, \leq, =, \neq, \geq, >\}$
- ако је $i > k$ или $i > l$ d-торци додајемо префиксне нуле

Пример 4.3.1. *Примери коришћења d-торки:*

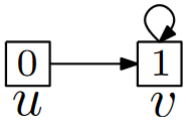
- *(0, 1, 0, 1) $=_0$ (0, 2, 0, 1) што значи да треба да важи (0) = (0) што је тачно*
- *(0, 1, 0, 1) $\geq_3$ (0, 2, 0, 1) што значи да треба да важи (0, 1, 0, 1) $\geq$ (0, 2, 0, 1) што је нетачно*

Дефиниција 4.3.1. ($\mathbb{M}_\mathcal{G}$). $\mathbb{M}_\mathcal{G} \subseteq \mathbb{N}^d$ *дефинишемо као коначан скуп d-торки које имају 0 на парним позицијама и ненегативне целе бројеве на непарним позицијама.*

Дефиниција 4.3.2. (Мера напредовања парности). *Нека је $\mathcal{G} = (V, E, p)$ игра парности. Функција $\varrho : V \to \mathbb{N}^d$ је* мера напредовања парности *на игри $\mathcal{G}$ ако за сваку грану $(v, w) \in E$ важи:*

- $\varrho(v) \geq_{p(v)} \varrho(w)$ *ако је $p(v)$ парно*
- $\varrho(v) >_{p(v)} \varrho(w)$ *ако је $p(v)$ непарно*

Проблем са дефиницијом ϱ јесте што мере могу да се додају само чворовима у парним циклусима (циклуси у којима је највећи приоритет паран) јер је релација строго веће када је $p(v)$ непарно. Пример овог недостатка можемо видети на слици 4.1 где није могуће доделити меру чворовима јер $\varrho(v) >_1 \varrho(v)$ није тачно. Због тога морамо модификовати дефиницију тако да укључује и непарне циклусе.



Слика 4.1: Граф са непраним циклусом

Дефиниција 4.3.3. ($\mathbb{M}_\mathcal{G}^\top$). $\mathbb{M}_\mathcal{G}^\top = \mathbb{M}_\mathcal{G} \cup \{\top\}$ *дефинишемо тако да:*

- $m\{<, <_i\}\top$ *за свако* $m \in \mathbb{M}_\mathcal{G}$

- $\top =_i \top$ *за свако* i

Дефиниција 4.3.4. (Prog). *Ако је* $\varrho : V \to \mathbb{M}_{\mathcal{G}}^{\top}$ *и* $(v, w) \in E$, *онда је* $Prog(\varrho, v, w)$ *најмање* $m \in \mathbb{M}_{\mathcal{G}}^{\top}$ *тако да:*

- $m \geq_{p(v)} \varrho(w)$ *ако је* $p(v)$ *парно*
- $m >_{p(v)} \varrho(w)$ *или* $m = \varrho(w) = \top$ *ако је* $p(v)$ *непарно*

Допунили смо нашу дефиницију тако да укључује $m = \varrho(w) = \top$ ако је $p(v)$ непарно. Тиме смо омогућили да постоје непарни циклуси јер ако из чвора v прелазимо у чвор w, а важи $v = w$ као на слици 4.1, онда је и $\varrho(v) = \varrho(w)$ из чега следи $m = \varrho(v) = \varrho(w)$ што значи да је $\varrho(w) = \varrho(w) = T$.

Дефиниција 4.3.5. (Мера напредовања парности - проширење). *Нека је* $\mathcal{G} = (V, E, p)$ *игра парности. Функција* $\varrho : V \to \mathbb{M}_{\mathcal{G}}^{\top}$ *је мера напредовања парности на игри* $\mathcal{G}$ *ако за сваки чвор* $v \in V$ *важи:*

- *ако је* $v \in V_0$ *онда* $\exists_{(v,w)\in E} \varrho(v) \geq_{p(v)} Prog(\varrho, v, w)$
- *ако је* $v \in V_1$ *онда* $\forall_{(v,w)\in E} \varrho(v) \geq_{p(v)} Prog(\varrho, v, w)$

Дакле, знамо како се израчунава мера напредовања у играма парности и сада је потребно да видимо како ће нам то помоћи у дефинисању стратегија за играче. Ако је $\mathcal{G} = (V, E, p)$ игра парности, а $\varrho : V \to \mathbb{M}_{\mathcal{G}}^{\top}$ мера напредовања онда дефинишемо стратегију $\overline{\varrho} : V_{Parno} \to V$ тако да је $\overline{\varrho}(v)$ чвор w који следи из v чије је $\varrho(w)$ најмање. Будући да је ϱ мера напредовања $\overline{\varrho}$ је победничка стратегија за играча P_0 из $||q|| = \{v \mid v \in V \wedge \varrho(v) \neq \top\}$. Штавише, таква мера мора да постоји.

На игри $\mathcal{G} =$ (V, E, p) и мери напредовања $\varrho : V \to \mathbb{M}_{\mathcal{G}}^{\top}$ дефинисаћемо и оператор $\sqsubseteq$:

- $\mu \sqsubseteq \varrho$ ако је $\mu(v) \leq \varrho(v)$ за свако $v \in V$
- ако је $\mu \neq \varrho$ и $\mu \sqsubseteq \varrho$ онда пишемо $\mu \sqsubset \varrho$

Оператор $\sqsubseteq$ смо увели јер нам он даје структуру пуне решетке над скупом функција $V \to \mathbb{M}_{\mathcal{G}}^{\top}$. Разлог зашто нам је потребна таква структура лежи у теорији фиксних тачака и његово објашњење превазилази домен овог рада. Укратко, на основу теореме Кнастера и Тарског [45] (енг. *Knaster–Tarski theorem*) најмања мера напредовања μ, која је и победничка за играча P_0, мора да постоји и може се израчунати итерацијом фиксних тачака.

Дефиниција 4.3.6. (Функција подизања). *Lift(*ϱ*,* v*) за* $v \in V$ *дефинишемо са:*

$$Lift(\varrho, v)(u) = \begin{cases} q(u) & \text{ако је } u \neq v \\ min\{Prog(\varrho, v, w) \mid (v, w) \in E\} & \text{ако је } u = v \in V_{Parno} \\ max\{Prog(\varrho, v, w) \mid (v, w) \in E\} & \text{ако је } u = v \in V_{Neparno} \end{cases}$$

за свако $v \in V$ $Lift(\cdot, v)$ је монотоно, а функција $\varrho : V \to \mathbb{M}_{\mathcal{G}}^{\top}$ је мера напредовања акко је $Lift(\varrho, v) \sqsubseteq \varrho$, тј. ϱ је префискна тачка $Lift(\varrho, v)$ оператора, за свако $v \in V$.

4.3.1 Алгоритам

Користећи претходно дефинисане термине можемо конструисати алгоритам за проналажење најмање мере напредовања. Алгоритам је приказан на слици 2.

Алгоритам 2 Налажење мере напредовања

1: $\mu := \lambda v \in V.(0, ..., 0)$
2: **while** $\mu \sqsubset Lift(\mu, v)$ за $v \in V$ **do**
3: $\quad \mu := Lift(\mu, v)$

Временска сложеност овог алгоритма је $O(nm \cdot (\frac{n}{d/2})^{d/2})$, где је n број чворова, m број грана, а d највећи приоритет у графу. Просторна сложеност је $O(d \cdot n)$.

4.4 Детерминистички субекспоненцијални алгоритам

Један од првих и најзначајнијих алгоритама субекспоненцијалне сложености за решавање игара парности јесте алгоритам Јурџинског, Патерсона и Цвика [23] објављен 2008. године. Пре него што је овај алгоритам развијен, најбржи алгоритми за решавање игара парности јесу били субекспоненцијални али су били засновани на случајности, конкретно на случајном субекспоненцијалном симплекс алгоритму Калаија [24] (енг. *randomized subexponential simplex algorithms*) и другим сличним алгоритмима. У овом приступу случајност игра кључну улогу у сложености алгоритма и њеном спуштању у субекспоненцијалну класу, те се наметнуло питање да ли је могуће развити стабилнији детерминистички алгоритам који припада истој класи сложености.

За разлику до случајних алгоритама, детерминистички алгоритам који ћемо да прикажемо заснован је на Зиелонкином алгоритму. Сложености алгоритма је $n^{O(\sqrt{n/\log n})}$, $n^{O(\sqrt{n})}$ ако број излазних грана чворова графа није ограничен, која је отприлике иста као и сложеност случајног алгоритма Бјорклунда [3] (енг. *Björklund*).

Као што је већ напоменуто, алгоритам представља модификацију Зиелонкиног алгоритма који је приказан у претходном поглављу. Зиелонкин алгоритам има рекурзивну структуру и састоји се од два рекурзивна позива ка мањим играма које садрже, у најгорем случају, $n-1$ чворова што нам даје експоненцијалну сложеност. Субекспоненцијални алгоритам модификује експоненцијални алгоритам тако што унапређује други рекурзивни позив уз помоћ увођења новог појма названог доминион (енг. *dominion*).

Доминион представља слуп чворова D у којима играч P_i "влада", тј. може да победи полажењем из било ког чвора у D тако да не напусти скуп D и да не дозволи другом играчу да напусти скуп D. Доминион у коме влада играч P_i означавамо са i-доминион. Пример i-домиииона јесте победнички регион играча P_i W_i, али неки подскупови тог скупа такође могу бити доминиони. Значај доминиона у нашем алгоритму огледа се у две чињенице:

- Сваки i-доминион може бити уз мали број корака уклоњен из игре
- Други рекурзивни позив у алгоритму је ка игри из које је уклоњен i-доминион.

Дакле, ако из игре прво уклонимо све мале доминионе можемо значајно смањити величину игре у коју се улази другим рекурзивним позивом. Рекурентна једначина која описује нови алгоритам јесте $T(n) \leq T(n-1) + T(n-\ell) + O(n^\ell)$ за неко $\ell = \ell(n)$. Уз адекватан одабир $\ell(n)$ постижемо баланс између времена које је потребно да се пронађу доминиони и уштеде приликом смањивања другог рекурзивног позива.

4.4.1 Проналажење доминиона

Пре описивања начина на који се проналазе доминиони у игри парности и самог алгоритма решавања увешћемо потребне термине.

Дефиниција 4.4.1. (i-затворен скуп). *Скуп $B \subseteq V$ је i-затворен, где је $i \in \{0,1\}$, V_i представља скуп чворова који припадају играчу P_0, док $V_{\neg i}$ представља скуп чворова који припадају играчу P_1, ако за сваки чвор $u \in B$:*

- *ако је* $u \in V_i$ *онда постоји грана* $(u, v) \in E$ *таква да је* $v \in B$
- *ако је* $u \in V_{\neg i}$ *онда за сваку грану* $(u, v) \in E$ *имамо чвор* $v \in B$

Дакле, скуп B је i-затворен ако играч P_i увек може да изабере да остане у B, а играч $P_{\neg i}$ не може да "побегне"(изађе) из њега.

Лема 4.4.1. *За свако* $i \in \{0, 1\}$ *скуп* $W_i(\mathcal{G})$ *је* i*-затворен.*

Доказ. Доказ следи из дефиниције игара парности и теореме 2.2.2. □

Лема 4.4.2. *За сваки скуп* $A \in V$ *и* $i \in \{0, 1\}$ *скуп* $V \backslash F_i(A)$ *је* $(\neg i)$*-затворен.*

Доказ. Нека је $u \in V \backslash F_i(A)$. Будући да сваки чвор има бад једну излазну грану, ако је $u \in V_{\neg i}$ онда мора да постоји грана $(u, v) \in E$ из чвора u у скуп $V \backslash F_i(A)$ таква да је $v \notin F_i(A)$, иначе би u било у $F_i(A)$. Слично, ако је $u \in V_i$ онда све гране из чвора u морају да иду у скуп $V \backslash F_i(A)$. Стога је скуп $V \backslash F_i(A)$ $(\neg i)$-затворен. □

Лема 4.4.3. *Нека је* $\mathcal{G}'$ *подигра игре* $\mathcal{G}$*, и нека је* $i \in \{0, 1\}$*. Ако је скуп чворова* V'*, игре* $\mathcal{G}'$*,* i*-затворен, онда је* $W_i(\mathcal{G}') \subseteq W_i(\mathcal{G})$

Доказ. Победничка стратегија играча P_i из скупа $W_i(\mathcal{G}')$ у подигри $\mathcal{G}'$, је такође победничка из истог скупа у оригиналној игри $\mathcal{G}$. Играч $\neg i$ не може да побегне у скуп $V \backslash V'$ јер је скуп V' i-затворен у $\mathcal{G}$. □

Лема 4.4.4. *Нека је* $\mathcal{G}$ *игра парности,* $i \in \{0, 1\}$ *и* $j = \neg i$*. Ако је* $U \subseteq W_j(\mathcal{G})$ *и* $U^* = F_j(U)$*, онда је* $W_j(\mathcal{G}) = U^* \cup W_j(\mathcal{G} \backslash U^*)$ *и* $W_i(\mathcal{G}) = W_i(\mathcal{G} \backslash U^*)$*.*

Доказ. Нека је $A_j = U^* \cup W_j(G \backslash U^*)$ и $A_i = W_i(G \backslash U^*)$. Будући да је (A_i, A_j) подскуп V, довољно је да докажемо да је $A_i \subseteq W_i(G)$ и $A_j \subseteq W_j(G)$. На основу леме 4.4.2 скуп чворова $G \backslash U^*$ је i-затворен, тако да смо на основу леме 4.4.3 доказали да важи $A_i \subseteq W_i(G)$.

Да би смо доказали $A_j \subseteq W_j(G)$ претпоставимо да је $U \subseteq W_j(G)$. У том случају постоји стратегија δ' за играча P_j у игри G која доноси победу из сваког чвора у скупу U, и стратегија δ'' која је победничка стратегија за играча P_j у игри $G \backslash U^*$ на скупу $W_j(G \backslash U^*)$. Стратегија δ затим се добија комбиновањем стратегија δ' и δ'' на следећи начин:

- Ако се игра одвија у скупу $W_j(G \backslash U^*)$ онда се прати стратегија δ''
- Иначе, искористити стратегију принудног скупа на скупу U и наставити игру пратећи стратегију δ'

Стратегија δ је добро дефинисана на основу леме 4.4.1 тако да играч P_i може да побегне из скупа $W_j(G \backslash U^*)$ само у скуп U^*. Будући да стратегије не зависе од претходно посећених чворова (позиционе стратегије), стратегија δ је победничка за играча P_j зато што ако икада престане да прати стратегију δ'' прелази на стратегију δ' која је у евентуалном бесконачном циклусу победничка за играча P_j. □

Лема 4.4.5. *Нека је $\mathcal{G}$ игра парности са n чворова и нека је $\ell \leq n/3$. Постоји алгоритам сложености $O(n^\ell)$ који проналази непразан доминион у $\mathcal{G}$ величине највише ℓ, или утврђује да такав доминион не постоји.*

Доказ. Ако је $\ell \leq n/3$ онда за свако $j \leq l$ имамо да је $\binom{n}{j}/\binom{n}{j-1} > 2$. Дакле, број $\sum_{j=1}^{\ell}\binom{n}{j}$ подскупова скупа чворова V величине највише ℓ је највише $2\binom{n}{j}$. За сваки такав подскуп U проверавамо у времену $O(\ell^2)$ да ли је 0-затворен или 1-затворен, ако није ни једно ни друго онда U није доминион. Ако је U i-затворен за $i \in \{0,1\}$ формирамо игру $\mathcal{G}[U]$ која је иста као игра (G) само што је ограничена на скуп U. Затим на ту игру применимо експоненцијални алгоритам Зиелонке и у времену $O(2^\ell)$ израчунавамо да ли играч P_i има победничку стратегију из сваког чвора скупа U, ако има, U је доминион, ако нема, није. Укупно време извршавања алгоритма је $O(\binom{n}{j}2^\ell) = O(n^\ell)$. □

Ако је број излазних грана сваког чвора игре $\mathcal{G}$ два, онда доминион можемо пронаћи у још бржем времену од $O(n2^\ell \ell \log \ell)$ што је доказано у раду у доказу леме 6.2.

Алгоритам из доказа леме 4.4.5 означићемо са доминион$(\mathcal{G}, \ell)$. Овај алгоритам за игру парности $\mathcal{G}$ и максимални број чворова ℓ проналази пар (D, i) доминион и играча коме припада тај доминион, или враћа ($\emptyset$, -1) ако доминион није пронађен.

4.4.1.1 Нови алгоритам

На слици (3) можемо видети нови алгоритам који у себи садржи позив ка мало модификованом старом алгоритму (4) који је приказан у раду [23]. Нови алгоритам на почетку тражи доминион величине ℓ, где је $\ell = \lceil\sqrt{2n}\rceil$ изабрано као параметар који минимизује време извршавања целог алогоритма. Ако је доминион пронађен, лако га уклањамо из игре, као и његов принудни скуп, и настављамо са мањом игром. Ако доминион није пронађен, једноставно позивамо стари експоненцијални алгоритам који је модификован тако да му рекурзивни позиви не зову њега, већ нови алгоритам.

Алгоритам 3 Субекспоненцијални алгоритам

```
1:  procedure ново-реши(𝒢)
2:      if V(G) = ∅ then
3:          return (∅, ∅)
4:      n := |V(G)|
5:      ℓ := ⌈√(2n)⌉
6:      (D, i) := доминион(G, ℓ)
7:      j := ¬i
8:      if D = ∅ then
9:          (W_0, W_1) := старо-реши(𝒢)
10:     else
11:         (W'_0, W'_1) := ново-реши(𝒢 \ F_i(D))
12:         (W_j, W_i) := (W'_j, V(𝒢) \ W'_j)
13:     return (W_0, W_1)
```

Алгоритам 4 Експоненцијални алгоритам модификација

```
1:  procedure старо-реши(𝒢)
2:      p := највећи приоритет у 𝒢
3:      Y := чворови у 𝒢 са приоритетом p
4:      i := p mod 2
5:      j := ¬i
6:      A := F_i(Y)
7:      (W'_0, W'_1) := ново-реши(𝒢 \ A)
8:      if W_{j'} = ∅ then
9:          (W_i, W_j) := (V(𝒢), ∅)
10:     else
11:         (W''_0, W''_1) := ново-реши(𝒢 \ F_j(W'_j))
12:         (W_i, W_j) := (W''_i, V(𝒢) \ W''_i)
```

Теорема 4.4.1. *Алгоритам ново-реши(𝒢) успешно проналази победничке регионе играча P_0 и P_1 игре парности $\mathcal{G}$. У игри која садржи n чворова његова сложеност је $n^{O(\sqrt{n})}$.*

Доказ. Коректност алгоритма се непосредно доказује. Алгоритам покушава да пронађе доминионе величине највише $\ell = \lceil\sqrt{2n}\rceil$. На основу леме 4.4.5 за ово је потребно $O(n^\ell)$ времена. Ако је доминион пронађен наставља се са игром која има највише $n-1$ чворова те је време извршавања највише $T(n-1)$. Иначе, ако доминион није пронађен позива се старо-реши које затим позива ново-реши$(G \setminus A)$ чије је време извршавања највише $T(n-1)$. Ако је W'_j празно, онда смо завршили, иначе је W'_j победнички регион играча P_j на скупу $\mathcal{G} \setminus F_i(A)$ који је исти као и над скупом $\mathcal{G}$ на основу леме 4.4.4. Дакле, W'_j је j-доминион у игри $\mathcal{G}$, а будући да знамо да не постоји доминион

са највише ℓ чворова, значи да је $|W'_j| > \ell$ и да други рекурзивни позив ново-реши($\mathcal{G} \setminus F_j(W'_j)$) захтева највише $T(n-\ell)$ времена. Укупна рекурентна једначина гласи:

$$T(n) \leq O(n^l) + T(n-\ell) + T(n-l)$$

Ако је $\ell = \lceil\sqrt{2n}\rceil$ онда је $T(n) = n^{O(\sqrt{n})}$, што је доказано у раду у доказу теореме 8.1. □

4.5 Напредни алгоритми

Након што смо приказали три алгоритма за решавање игара парности који су кључни за развијање те области и будуће алгоритме, сада ћемо направити кратак преглед напреднијих алгоритама. Алгоритми које ћемо приказати представљају тренутни врхунац области и развијани су у последње две године. Због сложености самих алгоритама, као и области на којима се заснивају, биће приказане само њихове кључне идеје, док ће читалац кроз референце моћи о њима детаљније да се информише.

4.5.1 Алгоритам заснован на аутоматима

Овај алгоритам представили су Ди Стасио, Мурано, Перели и Варди [9] 2016. године као први алгоритам који користи наизменичне аутомате. Алгоритам је заснован на `APT` алгоритму представљеном у раду Купфермана (енг. *Kupferman*) и Вардија [29]. `APT` алгоритам заснован је на решавању проблема празнине (енг. *emptiness problem*) на наизменичном аутомату парности (енг. *alternating parity automaton*). Будући да је `APT` алгоритам у том раду дат само у скицама и без доказа коректности, овај рад покушава да исправи те недостатке и нађе прецизан и доказано коректан алгоритам. Добијени алгоритам припада класи сложености $O(n^{\lceil\log(m)\rceil+6})$.

4.5.1.1 Унапређени `APT` алгоритам

Нека је $\alpha = F_1 \cdot \ldots \cdot F_k$ услов парности, где је $F_i = p^{-1}(i)$ скуп свих чворова приоритета i. Уводимо функцију $Win_0(\alpha, V, A)$ која проналази скуп чворова из којих играч P_0 има стратегију тако да:

- избегава чворове скупа A (енг. *avoiding*)

- форсира посећивање чворова скупа V (енг. *visiting*) или побеђује у услову парности.

слично се дефинише функција $Win_i(\alpha, V, A)$ за играча P_1.

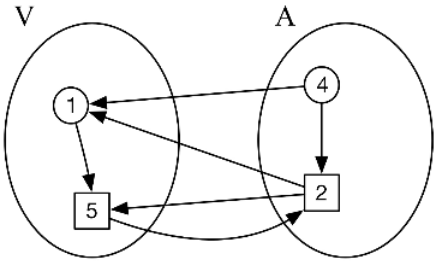
Алгоритам конструишемо уз помоћ индукције:

- База: Ако је α празно и чворови су подељени на два скупа, V и A, онда је $Win_i(\alpha, V, A)$ скуп чворова из ког играч P_i може у једном кораку да дође до скупа V. Пример можемо видети на аутомату са слике 4.2, када је $i = 1$ онда је $Win_1(\alpha, V, A) = \{1, 2\}$.
- Индуктивни корак: $\alpha = F_j \cdot \alpha'$, где је $\alpha' = F_{j+1} \cdot ... \cdot F_k$ и ако је j непарно $i = 0$, иначе $i = 1$.

Скуп V садржи чворове са приоритетима мањим од j који су добри за играча P_i, док скуп A садржи чворове са приоритетима мањим од j који су лоши за играча P_i.

Дефиниција 4.5.1. (Функција победе). *Функција $Win_i(\alpha, V, A)$ дефинише се на следећи начин:*

- *Скуп F_j са чворовима мањег приоритета уклања се из α*
- *Скуп F_j се означи са A као скуп за избегавање за играча P_i*
- *Чворови у скупу F_j из којих играч P_i може да форсира посећивање скупа V пребацују се у скуп V*



Слика 4.2: Пример аутомата

4.5.2 Квазиполиномски алгоритам

Следећи алгоритам који ћемо представити јесте квазиполиномски алгоритам који су развили Калуде (енг. *Calude*), Јаин (енг. *Jain*), Кусаинов (енг. *Khoussainov*), Ли (енг. *Li*), и Стефан (енг. *Stephan*) [5] 2017. године. Алгоритам се заснива на примени статистика које би у току игре наговештавале победу једног од играча. Ако играч има победничку стратегију онда ће то његове статистике, после одређеног броја потеза у игри, и

приказати. Дакле, да би играч P_0 победио он мора да стигне до чвора у коме његове статистике указују победу, у супротном побеђује играч P_1. На овај начин решавање игара парности сведено је на решавање игре доспутности квазиполиномске величине, која може да се реши у полиномском времену $O(|V| + |E|)$ у зависности од броја грана.

Постоје различите статистике које могу бити примењене у овом алгоритму. Приказаћемо пример једне, у овом случају, за играча P_0 који побеђује у случају непарних приоритета (статистике за играча P_1 су симетричне с тим што он тежи ка парним приоритетима).

Пример 4.5.1. *Статистику b ћемо представити као вектор позитивних целих бројева $b_0, b_1, ..., b_k$ где је $k = \lceil \log(n) + 2 \rceil$. Иницијално све вредности статистике b једнаке су нули. Играчи затим играју игру померајући жетон од чвора до чвора. Сваки пут када се дође у нови чвор узима се његов приоритет b и бира се највеће i које задовољава једно од следећих правила:*

(a) *b и $b_0, b_1, ..., b_{i-1}$ имају непарне вредности, док b_i има парну*

(b) *$0 < b_i < b$*

Ако смо нашли такво i постављамо $b_i = b$ и $b_j = 0$ за све $j < i$.

Пример израчунавања ове статистике над секвенцом чворова

1 6 7 5 1 4 5 3 2 1 3 2 3 1 3 3 1 2 1

можемо видети на слици 4.3. Играч P_0 побеђује у игри ако је $b_{\lceil \log n \rceil + 2} > 0$.

Након што смо дефинисали и израчунали статистику остао нам је још један корак у алгоритму а то је свођење игре парности на игру доступности, и решавање те игре. Игру доступности дефинисаћемо као усмерени граф (V', E') и скуп чворова T који је скуп цильних чворова играча P_0, tj. ако игри достигне неки од чворова тог скупа играч P_0 је победник. Игру парности сводимо на игру доступности тако што један чвор игре доступности представљамо преко тројке (v, p, w), где је v чвор у игри парности, p ознака ком играчу припада тај чвор, и w тренутна победничка статистика за играча P_0. Прелазак из чвора (v, p, w) у чвор (v', p', w') могуће је ако постоји грана у игри парности између v и v', чворови не припадају истом играчу, тј. $p \neq p'$, и статистика w није већ донела победу играчу P_0. Скуп T састоји се од свих чворова у којима победничка статистика играча P_0 показује победу за тог играча. Игра почиње из истог чвора као и игра парности. Играч P_0 побеђује ако се у игри дошло до чвора у скупу T, док играч P_1 побеђује ако се чворови у скупу T никад не достигну и игра траје бесконачно дуго.

Потез	b_4, b_3, b_2, b_1, b_0	Тренутна i-секвенца	Правило
1	0,0,0,0,0	0:1	1.(a)
6	0,0,0,0,6	0:1 6	1.(b)
7	0,0,0,0,7	1 6 0:7	1.(a)
5	0,0,0,5,0	1 6 1:7 1:5	1.(a)
1	0,0,0,5,1	1 6 1:7 1:5 0:1	1.(a)
4	0,0,0,5,4	1 6 1:7 1:5 0:1 4	1.(b)
5	0,0,0,5,5	1 6 1:7 1:5 1 4 0:5	1.(a)
3	0,0,3,0,0	1 6 2:7 2:5 1 4 2:5 2:3	1.(a)
2	0,0,3,0,0	1 6 2:7 2:5 1 4 2:5 2:3 2	
1	0,0,3,0,1	1 6 2:7 2:5 1 4 2:5 2:3 2 0:1	1.(a)
3	0,0,3,3,0	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3	1.(a)
2	0,0,3,3,0	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3 2	
3	0,0,3,3,3	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3 2 0:3	1.(a)
1	0,1,0,0,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1	1.(a)
3	0,3,0,0,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3	1.(b)
3	0,3,0,0,3	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 0:3	1.(a)
1	0,3,0,1,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1	1.(a)
2	0,3,0,2,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1 2	1.(b)
1	0,3,0,2,1	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1 2 0:1	1.(a)

Слика 4.3: Пример израчунавања победничке стратегије 2

Решавање добијене игре доступности обавља се по познатом алгоритму сложености $O(|V| + |E|)$ за задати граф (V, E), скуп T и почетни чвор s. Алгоритам је приказан на слици 5. Играч P_0 побеђује ако је, након извршавања алгоритма, $q_s = 0$.

Сложеност овог алгоритма је полиномска, $O(n^5)$ када број различитих вредности параметара логаритамски зависи од броја чворова $m \leq \log(n)$. Иначе, ако не постоји логаритамска зависност, алгоритам има сложеност $O(2^m \cdot n^4)$.

4.5.3 Сажете мере напредовања

Један од тренутно најефикаснијих алгоритама за решавање игара парности, како временски тако и просторно, јесте алгоритам сажетих мера напредовања Јурџинског и Лазића [20] из 2017. године. У ранијем раду Јурџинског [22], који је представљен у поглављу 4.3, описан је алгоритам решавања игара парности преко малих мера напредовања. Алгоритам сажетих мера напредовања покушава да докаже да је свака мера напредовања, на коначном графу, еквивалентна сажетије описаној мери напредовања, тј. да може сажетије да се прикаже без последица по коректност алгоритма. Такође, утврђено је да ако постоји победничка стратегија за

Алгоритам 5 Алгоритам решавања игре доступности

```
1: procedure реши игру доступности((V, E), T, s)
2:     for each v ∈ V do
3:         if p_v = 0 then
4:             q_v := 1;
5:         else
6:             for each v ∈ V − T do
7:                 q_v := број-следбеника(v)
8:     for each v ∈ T do
9:         A(v)
10:
11: procedure A(v)
12:     if q_v = 0 then
13:         return
14:     if q_v = 1 then
15:         q_v := 0
16:         for each w ∈ преци(v) do
17:             A(w)
18:         return
19:     if q_v > 1 then
20:         q_v = q_v-1
21:         return
```

играча P_0 из сваког чвора графа игре, онда мора да постоји и сажета мера напредовања.

Алгоритам је инспирисан квазиполиномским алгоритмом Калудеа [5] а главно унапређење у односу на њега јесте то што је за овај алгоритам потребан квазилинеаран простор за извршавање, а не квазиполиномски. Ово је постигнуто применом нове методе сажетог кодирања уређених стабала које се обавља тако што кретањем кроз стабло преименујемо одређене чворове (самим тим и мењамо путање рачвања из тих чворова), притом чувајући уређеност стабла, са намером да добијемо изоморфно стабло које би нам омогућило да краће (сажетије) опишемо путање до листова стабла. Путања у стаблу висине h и највише n листова, може бити кодирана са $\log h \cdot \log n$ битова коришћењем ограничених прилагодљивих бројача (енг. *bounded adaptive multi-counters*). Овако кодирање може се применити и на игре парности. Ако имамо граф са n чворова и d различитих вредности приоритета мера приоритета тог графа може да се посматра као путања у стаблу висине $d/2$ са највише n листова. Одатле следи да постоји $2^{\log d \cdot \log n} = n^{\log d}$ могућих завршетака за сваки чвор, што је осетно унапређење у односу на границу $2^{d/2 \cdot \log n} = n^{d/2}$

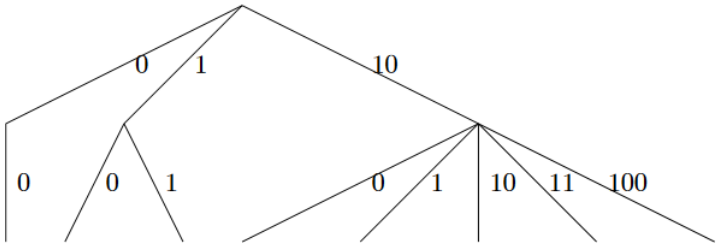
из рада Јурџинског [22]. Ипак, техника подизања (енг. lifting) коришћена у том раду може бити прилагођена тако да израчунава сажету репрезентацију мере напредовања у квазиполиномском времену $O(n^{\log d})$ и квазилинеарном простору $O(n \log n \cdot \log d)$.

Дефинишимо $B_{g,h}$ као скуп ограничених прилагодљивих бројача који се састоје од h-торки бинарних ниски дужине највише g. На пример $(0, \varepsilon, 1, 0)$ и $(\varepsilon, 1, \varepsilon, 0)$ припадају скупу $B_{3,4}$ 4-торки дужине 3, док $(0, 1, \varepsilon, \varepsilon, 1)$ и $(10, \varepsilon, 01, \varepsilon)$ не припадају. Дефинишимо затим линеарно уређење $<$ бинарних ниски, где је b бинарна цифра, а s и s' су ниске, као:

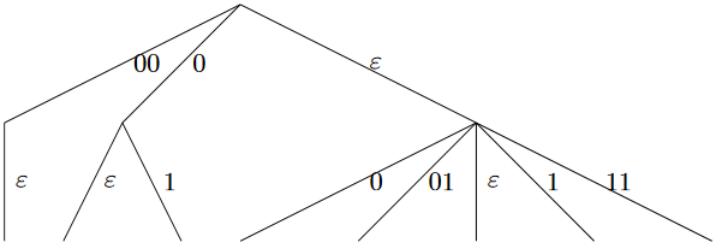
$$0s < \varepsilon, \qquad \varepsilon < 1s, \qquad bs < bs' \text{ акко } s < s'$$

Лема 4.5.1. *(Сажето кодирање стабала) За свако уређено стабло висине h и са највише l листова, постоји кодирање у коме је свака путања i-торка дужине највише $\lceil \log l \rceil$, где је $i \leq h$.*

Доказ. Доказ се може наћи у раду [20]. □



Слика 4.4: Пример стабла са бинарним путањама, путања до крајње десног чвора има 5 битова.



Слика 4.5: Пример сажето кодираног стабла са бинарним путањама, путање до било ког чвора имају највише $\lceil \log 8 \rceil = 3$ бита, у овом случају то су 2 бита.

Линеарно уређен скуп ограничених прилагодљивих бројача дефинисаћемо са:

$$S_{\eta,d} = \bigcup_{i=0}^{d/2} B_{\lceil \log \eta \rceil, i},$$

и нека је $S_{\eta,d}^{\top}$ скуп $S_{\eta,d}$ проширем елементом $\top$

Дефиниција 4.5.2. (Функција подизања). *$Lift_v(\mu)(u)$ за $v \in V$ дефинишемо са:*

$$Lift_(v)(\mu)(u) = \begin{cases} \mu(u) & \text{ако је } u \neq v \\ min_{(v,w)\in E} lift(\mu, v, w) & \text{ако је } u = v \in V_{Parno} \\ max_{(v,w)\in E} lift(\mu, v, w) & \text{ако је } u = v \in V_{Neparno} \end{cases}$$

$\bar{\imath}$гре је $lift(\mu, v, w)$ најмање $\delta \in S_{\eta,d}^{\top}$ $\bar{\imath}$тако да је $\delta \geq \mu(v)$ и (v, w) на$\bar{\imath}$редује у $\mu[v \to \delta]$.

Алгоритам унапређења дат је на слици 6. Алгоритам враћа и победничке чворове и победничку тактику за играча P_0.

Алгоритам 6 Налажење сажете мере напредовања

1: $\mu := V \to S_{\eta,d}^{\top}$ тако да пресликава свако $v \in V$ на последњи елемент у $S_{\eta,d}^{\top}$, који је празна секвенца ε
2: **while** $\mu \neq Lift_v(\mu)$ за $v \in V$ **do**
3: $\quad \mu \leftarrow Lift_v(\mu)$
4: **return** $W_{Parno} = \{v \ : \ \mu(v) \neq \top\}$ скуп победничких чворова за играча P_0, и победничку стратегију која за сваки чвор $v \in W_{Parno}$ бира излазећу грану која доводи до напретка у μ

Временска сложеност овог алгоритма је $max\{2^{O(d \log d)}, O(mn^{2.38})\}$, док је просторна $O(n \log n \cdot \log d)$.

5. Свођење на SAT

Након што смо обавили све потребне припреме, и детаљно смо се упутили у то шта представљају игре парности, на чему су засноване, као и које су могуће стратегије њиховог решавања, време је да нашу пажњу усмеримо ка другој централној теми овог рада, а то је решавање игара парности свођењем на SAT проблем, или проблем задовољивости.

5.1 Мотивација

SAT је први познати NP-комплетан проблем што су доказали независно Кук (енг. *Cook*) [1] 1971. године, и Левин (енг. *Levin*) [31] 1973. године.

Теорема 5.1.1 (Кук-Левинова теорема)**.** *SAT проблем је NP-комплетан.*

Једна од последица ове теореме важна је за овај рад а то је да сваки проблем који се налази у класи NP може, у полиномском времену, да се сведе на SAT проблем. Као што је раније наведено решавање игара парности припада класи сложености NP (прецизније NP∩coNP 2.4.1) што за собом повлачи да мора да постоји редукција, у полиномском времену, овог проблема на SAT проблем. Ова чињеница, као и убрзан развој све бољих SAT решавача и бржих рачунара, били су мотивација да се развије алгоритам за свођење проблема решавања игара парности на SAT проблем.

5.2 Постојећи алгоритми

Досадашњи радови на тему свођења проблема решавања игара парности на SAT ослањају се на идеју локалног препознавања победничких стратегија, која је основа и за алгоритам малих мера напредовања Јурџинског [22] (који је приказан у овом раду у поглављу Стратегије решавања 4.3). Разлог за ово представља природна и интуитивна редукција овог алгоритма на SAT. Уместо

да се вредности чворова (мере напредовања), које су представљене d-торкама, итеративно одређују, сваки бит те репрезентације прослеђују се SAT решавачу као појединачна променљива. Односи између тих вредности одређени су само преко оператора $<$ и $\leq$, а знамо да они означавају лексикографско уређење над d-торкама. Ово није тешко приказати у виду ограничења над појединачним битовима, који сада представљају променљиве SAT решавача.

Први рад који се дотакао ове теме објавио је Ланге [30] 2005. године и он представља полазну тачку за будућа истраживања. Идеју за овај рад дао је Емерсон својим коментаром где је описао укључивање проблема провере модела μ-рачуна у NP. Његов коментар гласи "Погодити ранг сваке μ-подформуле у сваком стању система прелаза. Показати да је лексикографско уређење над торкама система добро засновано". Пратећи идеју рангова, и локалног препознавања победничких стратегија, Ланге је дефинисао појам μ-анотација, којим је на другачији начин формулисао мере напредовања Јурџинског. Будући да се μ-анотације не ажурирају динамички, као мере напредовања, тај назив је ипак задржан, али се заслуге за теорију μ-анотација приписују Јурџинском.

Као што је наведено у Лангеовом раду, он је прототип и први покушај да се оваква врста стратегије примени на решавање игара парности. Рад који наставља његово истраживање, и унапређује његове методе, објавио је заједно са Хиљанком, Кеинаненом, и Ниемелом [16] 2012. године. У овом раду редукција на SAT врши се у два корака, прво у логику разлика [35], а затим на SAT. Логика разлика је SMT теорија која комбинује исказну логику са теоријом разлика целих бројева. Редукција на логику разлика, осим што чини међукорак у редукцији на SAT, и сама по себи представља начин решавања игара парности уз коришћење SMT решавача заснованих на приступима као што је DPLL(T) [35]. У другом кораку цели бројеви и ограничења над њима, који су елементи логике разлика, замењују се истинитосним променљивима и ограничењима над њима чиме је редукција на SAT завршена.

У овом раду биће приказан само алгоритам Лангеа, Хиљанка, Кеинанена и Ниемеле из 2012. године, будући да је он заснован на истој идеји и представља унапређење Лангеовог првобитног алгоритма.

5.3 Основе алгоритма

Централна идеја овог алгоритма поклапа се са идејом алгоритма малих мера напредовања Јурџинског [22], а то је локално препознавање победничких стратегија. У свом раду Јурџински придружује сваком чвору у графу игре парности целобројну вредност - меру напредовања. На основу ових мера играч може да процени да ли је за њега повољно да помери жетон у тај чвор или не. Мере су приказане као d-торке ненегативних вредности над којима постоје оператори за лексикографско уређење. Јурџински затим дефинише функцију напредовања парности коју итеративно унапређује од њене прве форме (4.3.2) у којој лексикографски пореди битове d-торки, до њене финалне форме (4.3.6) која је заснована на итерацији фиксних тачака.

У алгоритму који приказујемо чворовима се придружују μ-торке $\bar{a} = (a_1, ..., a_n)$ на основу изабране μ-анотације. Уколико је μ-анотација успешна, онда постоји и победничка стратегија за једног од играча (ког смо претходно изабрали). Наш задатак је да пронађемо такву μ-анотацију, тј. такав распоред μ-торки. То постижемо дефинисањем формула у логици разлика које представљају структуру графа игре (распоред чворова и грана) и односе између битова μ-торки, који морају бити задовољени да би μ-анотација била успешна. Те формуле се затим преводе у исказну логику где остаје само да се дефинише поређење појединачни битови μ-торки. Тако конструисана формула исказне логике затим може да се проследи произвољном SAT решавачу који би, уколико је формула задовољива, вратио као излаз μ-торке свих чворова као и гране које су укључене у победничу стратегију задатог играча. Уколико је формула незадовољива, то значи да у датој игри не постоји победничка стратегија за задатог играча.

Ако имамо игру парности $\mathcal{G} = (\mathrm{V}, \mathrm{E}, v_0, \Omega)$, нека је $Odd(\mathcal{G}) = \{p \mid p$ је непарно и $\Omega(v) = p$ за неко $v \in V\}$, и $Odd^{<p}(\mathcal{G}) = Odd(\mathcal{G}) \cap \{i \mid 1 \leq i < p\}$, и $mop(\mathcal{G}) = max\, Odd(\mathcal{G})$.

μ-торка за $\mathcal{G}$ представља торку $\bar{a} = (a_1, a_3, ..., a_{mop(\mathcal{G})}) \in \mathbb{N}^{mop(\mathcal{G})/2}$. Ако имамо две μ-торке $\bar{a} = (a_1, ..., a_{mop(\mathcal{G})})$ и $\bar{b} = (b_1, ..., b_{mop(\mathcal{G})})$ за $\mathcal{G}$ и приоритет p за који важи $p \leq mop(\mathcal{G})$ онда дефинишемо

$$\bar{a} \trianglelefteq_p \bar{b} \text{ акко } \begin{cases} \text{за свако } i \in Odd^{<p}(\mathcal{G}) : a_i \leq b_i & \text{ако је } p \text{ парно,} \\ a_p < b_p \text{ и за свако } i \in Odd^{<p}(\mathcal{G}) : a_i \leq b_i & \text{иначе} \end{cases}$$

Ако је $\bar{a} = (a_1, a_3..., a_{mop(\mathcal{G})})$ и $p \in Odd(\mathcal{G})$ онда $\bar{a}^{(p)}$ означава p компоненту $\bar{a}$, односно a_p.

μ-анотација за игру $\mathcal{G}$ је функција η која додељује сваком чвору $v \in V$ μ-торку. μ-анотација је успешна акко за свако $v \in V$:

- $v \in V_\forall$ онда за свако $w \in vE$: $\eta(w) \trianglelefteq_{\Omega(w)} \eta(v)$, и
- $v \in V_\exists$ онда постоји $w \in vE$: $\eta(w) \trianglelefteq_{\Omega(w)} \eta(v)$

Пример 5.3.1. *Пример коришћења μ-аноташија из нашеī референтноī рада [16] даī је на слици 5.1 īде је сваком од чворова īрафа ūридружена је μ-ūорка. Чворови означени ромбовима ūūадају иīрачу P_0, док чворови означени квадраūима ūūадају иīрачу P_1, а иīра ūочиње из īорњеī левоī чвора ūриориūеūа 2. Имамо вредносūи $Odd(\mathcal{G}) = \{1, 3\}$ и $mop(\mathcal{G}) = 3$, шūо значи да су μ-ūорке облика (a_1, a_3). Провером формуле усūешносūи μ-аноūација, која је наведена изнад, уūврђујемо да задаūа μ-аноūација η није усūешна. Грана која сūречава аноūацију да буде усūешна јесūе īрана од доњеī левоī чвора, ūриориūеūа 4, ка чвору са њеūове десне сūране, ūриориūеūа 1. Правило усūеха налаже да у ūој īрани мора да важи (1,0)$\trianglelefteq_1$(1,0), шūо даље имūлицира да важи 1$<$1 и 0$\leq$0, шūо није ūачно.*

Након шūо смо доказали да μ-аноūација која је задаūа на īрафу са слике није усūешна, ūосūавља се ūиūање како можемо наћи μ-аноūацију која јесūе усūешна. Одīовор на ово ūиūање лежи у вези између μ-аноūација за ūодиīру задаūе иīре ūарносūи и ūобедничких сūраūеīија у ūој иīри.

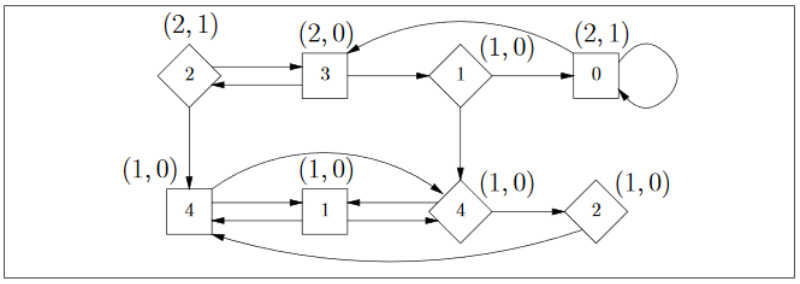
Поīледајмо ūоново наш īраф са слике 5.1 и ūокушајмо да ūронађемо ūобедничку сūраūеīију за иīрача P_0, коме ūūадају чворови са дијамаūима, и који ūобеђује ако је најмањи ūриориūеū који се ūојављује бесконачно мноīо ūуūа ūаран. Примеūимо да само иīрач P_0 одлучује да ли ће иīра икада ūрећи у доњи део īрафа (доња чеūири чвора) јер њему ūūадају оба чвора која имају излазне īране ка доњем делу. Ако се фокусирамо само на īорњи део īрафа, као ūодиīру ūочеūноī īрафа, увидећемо да у њој иīрач P_0 има ūобедничку сūраūеīију. У īорњем делу īрафа можемо идеūификоваūи ūри бесконачна циклуса:

- *2, 3, 2...*
- *3, 1, 0, 3, 1, 0...*
- *0, 0, 0...*

У ūрвом циклусу након неūарноī ūриориūеūа 3 мора да дође мањи ūарни ūриориūеū 2, шūо одīовара иīрачу P_0 јер би у ūом циклусу најмањи ūриориūеū који се ūојављује бесконачно мноīо ūуūа био ūаран. У друīом

циклусу се из чвора приоритета 3 прелази у чвор са приоритетом 1, из кога се након тога прелази у чвор приоритета 0, а затим назад у чвор приоритета 3. Најмањи приоритет који се појављује бесконачно много пута у овом случају је приоритета 0 и он је паран. У трећем циклусу, који је тривијалан, ако се заувек остане у чвору приоритета 0 долазимо до истог исхода.

Детаљнија анализа показала нам је да је ова μ-аноташија ипак успешна, али не на целом графу игре, већ на подграфу подигре. И заиста, не постоји ниједна μ-торка која задовољава $\eta(v_4) \trianglelefteq_4 \eta(v_5)$ и $\eta(v_5) \trianglelefteq_1 \eta(v_4)$ јер би то значило да је $\eta(v_4)^{(1)} \leq \eta(v_5)^{(1)}$ и да је истовремено $\eta(v_4)^{(1)} > \eta(v_5)^{(1)}$. Што значи да не постоји успешна μ-аноташија на целом графу игре.



Слика 5.1: Пример графа игре парности са μ-анотацијама.

Лема 5.3.1. *Нека је $\mathcal{G} = (V, E, v_0, \Omega)$ игра парности, η успешна μ-аноташија за $\mathcal{G}$, и $\pi = v_0 v_1 \ldots$ игра у $\mathcal{G}$. Ако за свако $i \in \mathbb{N}$ важи $\eta(v_{i+1}) \trianglelefteq_{\Omega(v_{i+1})} \eta(v_i)$ онда је најмањи приоритет који се појављује бесконачно много пута у π паран.*

Доказ. Претпоставимо да је најмањи приоритет p који је појављује бесконачно много пута у π непаран. Онда постоји бесконачно много чворова $v_{i1}, v_{i2}, \ldots$ у π таквих да $\eta(v_{i_1})^{(p)} > \eta(v_{i_2})^{(p)} > \ldots$, јер ћемо доћи до ситуације да не постоји нижи парни приоритет који задовољава $\eta(v_{i_j})^{(p)} < \eta(v_{i_{j+1}})^{(p)}$ за неко $j \in \mathbb{N}$. Али онда не можемо да задовољимо $\eta(v_{i+1}) \trianglelefteq_{\Omega(v_{i+1})} \eta(v_i)$ за свако $i \in \mathbb{N}$ зато што су природни бројеви добро засновани. □

Теорема 5.3.1. *Нека је $\mathcal{G}$ игра парности и нека је σ позициона стратегија за играча $\exists$. Онда постоји успешна μ-аноташија за $\mathcal{G}|_\sigma$ акко је σ победничка стратегија.*

Доказ. ($\Leftarrow$) Претпоставимо да постоји успешна μ-анотација η за игру парности $\mathcal{G}|_\sigma = (V, E_\sigma, v_0, \Omega)$, и имамо игру $\pi = v_0 v_1 ...$ која прати стратегију σ. Знамо да мора да важи $\eta(v_{i+1}) \trianglelefteq_{\Omega(v_{i+1})} \eta(v_i)$ за свако $i \in \mathbb{N}$. Према леми 5.3.1 најмањи приоритет који се појављује бесконачно много пута у π је паран, што значи да играч P_0 побеђује у свим таквим играма π, и да је σ заиста победничка стратегија.

($\Rightarrow$) Претпоставимо да постоји победничка стратегија σ за игру парности $\mathcal{G} = (V, E, v_0, \Omega)$, и нека је $\mathcal{G}|_\sigma = (V, E_\sigma, v_0, \Omega)$. За свако $p \in Odd(\mathcal{G})$ нека је $E_{\sigma,p} = E_\sigma \cap \{(v, w) \in E \mid \Omega(w) \geq p\}$ скуп грана у $\mathcal{G}|_\sigma$ које воде до чворова приоритета не мањих од p, који су у складу са стратегијом σ. Такође, за сваки чвор $v \in V$ нека је $W_v^p = \{w \mid (v, w) \in E^+_{\sigma,p}\} \cap V_p$ скуп чворова приоритета p који су достижни из v. $E^+_{\sigma,p}$ представља транзитивно затворење $E_{\sigma,p}$.

Сада дефинишемо μ-анотацију η за $\mathcal{G}|_\sigma$ као $\eta(v)^{(p)} = |W_v^p|$ за свако $v \in V$ и свако $p \in Odd(\mathcal{G})$. Претпоставимо затим да η није успешна μ-анотација, онда постоје $v \in V$ и $w \in vE_\sigma$ такви да је $\eta(w) \not\trianglelefteq_{\Omega(w)} \eta(v)$. Напоменимо да ако је $v \in V_{P_0}$ онда је w јединствено одређено стратегијом σ, иначе је постојање w осигурано тоталношћу (енг. *totality*).

Будући да је $w \in vE_\sigma$ тј. w је достижно из v пратећи стратегију σ, имамо $W_w^p \subseteq W_v^p$ за свако $p \geq \Omega(w)$ и $\eta(w)^{(p)} \leq \eta(v)^{(p)}$ за свако $p \leq \Omega(w)$. Према претпоставци да η није успешна μ-анотација добијамо да је $\eta(w) \not\trianglelefteq_{\Omega(w)} \eta(v)$ што даље значи да је $\Omega(w)$ непарно и $\eta(w)^{(\Omega(w))} \not< \eta(v)^{(\Omega(w))}$. Дакле $\eta(w)^{(\Omega(w))} = \eta(v)^{(\Omega(w))}$ тј. $W_w^{\Omega(w)} = W_v^{\Omega(w)}$ мора да важи. Што значи да је $(w, v) \in E^+_{\sigma,\Omega(w)}$ тј. v је достижно из w преко грана које су одређене стратегијом σ, тако да не наилазимо на чвор приоритета већег од $\Omega(w)$. Али онда постоји игра која прати стратегију σ у којој је најмањи приоритет који се појављује бесконачно много пута $\Omega(w)$ који је непаран, тј. у њој побеђује играч P_1. Ово дакле значи да σ није победничка стратегија, јер играч P_0 није победио пратећи је. Дакле, доказали смо да је претпоставка да η није успешна μ-анотација контрадикторна, што значи да мора да постоји успешна μ-анотација. □

Последица 5.3.1. *Нека је $\mathcal{G} = (V, E, v_0, \Omega)$ игра парности. Онда постоји успешна μ-анотација за $\mathcal{G}$ акко постоји успешна μ-анотација η за $\mathcal{G}$ таква да за свако $v \in V$ и свако $p \in Odd(\mathcal{G})$ важи $\eta(v)^{(p)} \leq |\{w \in V \mid (v, w) \in E_p^+\}|$.*

Вредности μ-анотације могу даље бити ограничене чврстим компонентама повезаности (енг. *strongly connected components - SCC*), у даљем тексту ЧКП, графа игре (дефиниција 5.3.1). Свака бесконачна игра

у коначној игри парности мора да буде заробљена у некој од чврстих компоненти повезаности игре парности. Такође, победник игре је одређен само на основу приоритета чворова из те ЧКП. Ово је случај јер сваки суфикс бесконачне игре чини скуп истих чворова који се појављују бесконачно много пута.

Нека је $\mathcal{G} = (V, E, v_0, \Omega)$ игра парности, $p \leq mop(\mathcal{G})$ и $scc : V \to \mathbb{N}$ функција која сваком чвору придружује јединствени индекс који означава ЧКП којој он припада. Важи $scc(v) = scc(w)$ акко је v достижно из w и обрнуто. Нека је E_p^+ транзитивно затворење релације $E \cap \{(v, w) \mid scc(v) = scc(w) \text{ и } \Omega(w) \leq p\}$. Онда последица 5.3.1 важи и у овој интерпретацији транзитивног затворења, али је μ-анотација η успешна акко за свако $v \in V$ важи:

- ако је $v \in V_{P_1}$ онда за свако $w \in vE : scc(v) = scc(w)$ имплицира $\eta(w) \trianglelefteq_{\Omega(w)} \eta(v)$, и
- ако је $v \in V_{P_0}$ онда постоји $w \in vE$ такво да је $scc(v) \neq scc(w)$ или $\eta(w) \trianglelefteq_{\Omega(w)} \eta(v)$.

Дефиниција 5.3.1. (Чврсте компоненте повезаности). *Чврсте компоненте повезаности графа представљају све његове максимално чврсто повезане подграфове. Граф је чврсто повезан ако постоји пут од сваког чвора у графу до сваког другог чвора у њему.*

5.4 Логика разлика

Нека је $P = \{P_1, P_2, ..., P_n\}$ скуп логичких променљивих и $X = \{x_1, x_2, ..., x_m\}$ скуп целобројних променљивих. Скуп атомичних формула логике разлика састоји се од исказа из P и атомичких формула облика $x_i \geq x_j$ и $x_i > x_j$ где су $x_i, x_j \in X$. Скуп $\mathcal{F}$ свих формула логике разлика је најмањи скуп атомичних формула који је затворен за негацију и конјункцију:

- ако је $\phi \in \mathcal{F}$ онда је и $\neg\phi \in \mathcal{F}$, и
- ако је $\phi \in \mathcal{F}$ и $\psi \in \mathcal{F}$, онда је и $(\phi \wedge \psi) \in \mathcal{F}$.

Бинарни оператори $\vee, \rightarrow, \longleftrightarrow$ дефинишу се на уобичајан начин преко негације и конјункције.

Валуација (P, X) састоји се из две функције. $\beta : P \to \{\top, \bot\}$ и $\beta : X \to \mathbb{Z}$, где је $\mathbb{Z}$ скуп целих бројева. Валуација се проширује на све формуле у $\mathcal{F}$ дефинисањем ограничења $\beta(x_i \geq x_j) = \top$ акко $\beta(x_i) \geq \beta(x_j)$, и $\beta(x_i > x_j) = \top$, акко $\beta(x_i) > \beta(x_j)$. Валуација задовољава формулу $\phi \in \mathcal{F}$ ако је

$\beta(\phi) = \top$. Формула је задовољива ако постоји валуација која је задовољава. За задату формулу $\phi \in \mathcal{F}$ проблем задовољивости представља проверу да ли је формула ϕ задовољива.

Теорема 5.4.1. *Проблем задовољивости за логику разлика је NP-комплетан.*

Доказ. Тврдња да је проблем задовољивости логике разлика NP-тежак следи из чињенице да се он може свести на задовољивост у исказној логици, док сама припадност класи NP следи из чињенице да постоји алгоритам који одређује задовољивост конјункције ограничења разлика у полиномском (кубном) времену за произвољно одабирану валуацију [32] [8]. □

5.5 Кодирање победничких стратегија у логици разлика

Након што смо увидели да нам је потребна победничка стратегија на подигри главне игре парности и дефинисали логику разлика, можемо да почнемо са првим кораком нашег алгоритма а то је кодирање победничких стратегија у логици разлика. У овом кораку за дату игру парности $\mathcal{G} = (V, E, v_0, \Omega)$ градимо формулу $\Phi_\mathcal{G}$ која је задовољива ако играч P_0 има победничку стратегију у $\mathcal{G}$. Формула се састоји из исказних променљивих S_v за сваки чвор $v \in V$, и $T_{v,w}$ за сваку грану $(v, w) \in E$. Оне се користе да пронађу подигру игре $\mathcal{G}$ на којој постоји победничка стратегија σ, која би затим била и победничка за играча P_0 на целој игри $\mathcal{G}$. Додатно $\Phi_\mathcal{G}$ садржи и целобројне променљиве x_p^v за сваки чвор $v \in V$ и сваки приоритет $p \in Odd(\mathcal{G})$ ради имплементирања μ-анотација.
$\Phi_\mathcal{G}$ дефинишемо као:

$$\Phi_\mathcal{G} = S_{v_0} \wedge \Phi_{P_0} \wedge \Phi_{P_1} \wedge \Phi_V \wedge \Phi_A$$

Док потформуле дефинишемо са:

$$\Phi_{P_0} = \bigwedge_{v \in V_{P_0}} (S_v \rightarrow \bigvee_{(v,w) \in E} T_{v,w}),$$

$$\Phi_{P_1} = \bigwedge_{v \in V_{P_1}} (S_v \rightarrow \bigwedge_{(v,w) \in E} T_{v,w}),$$

$$\Phi_V = \bigwedge_{v \in V, v \neq v_0} ((\bigvee_{(w,v) \in E} T_{w,v}) \to S_v), \text{ и}$$

$$\Phi_A = \bigwedge_{(v,w) \in E} (T_{v,w} \to \Psi_{v,w})$$

Где је $\Psi_{v,w}$:

$$\Psi_{v,w} = \begin{cases} \bigwedge_{p \in Neparno^{<\Omega(w)}(G)} (x_p^v \geq x_p^w) & \text{ако је } \Omega(w) \text{ парно,} \\ (x_{\Omega(w)}^v > x_{\Omega(w)}^w) \wedge \bigwedge_{p \in Neparno^{<\Omega(w)}(G)} (x_p^v \geq x_p^w) & \text{иначе} \end{cases}$$

S_{v_0} представља почетни чвор, он мора бити у подигри. Формуле Φ_{P_0}, Φ_{P_1} и Φ_V користе се да пронађу подигру у којој играч P_0 има победничку стратегију. Φ_{P_0} представља све гране које излазе из чворова који припадају играчу P_0. Будући да правимо стратегију за играча P_0, овде се може изабрати било која грана. Φ_{P_1} представља све гране које излазе из чворова који припадају играчу P_1. Будући да правимо стратегију за играча P_0, овде морамо размотрити сваку грану, јер је избор следеће гране на играчу P_1. Φ_V означава да сваки чвор у графу мора да има бар једну улазну грану, тј. да је граф повезан. Формула Φ_A представља имплементацију μ-анотација у логици разлика, и служи да пронађе победничку стратегију за играча P_0, ако она постоји.

Теорема 5.5.1. *Играч P_0 има победничку стратегију у игри парности $\mathcal{G}$ акко је формула логике разлика $\Phi_{\mathcal{G}}$ задовољива.*

Доказ. ($\Rightarrow$) Претпоставимо да играч P_0 има победничку стратегију σ у игри парности $\mathcal{G} = (V, E, v_0, \Omega)$. То значи да постоји функција β која додељује вредности променљивама S_v и $T_{v,w}$, за свако $v \in V$ и $(v, w) \in E$: $\beta(S_v) = \top$, $\beta(T_{v,w}) = \top$, ако постоји игра која прати стратегију σ која пролази кроз чвор v, и преко гране (v, w). Није тешко видети да су потформуле $S_{v_0}, \Phi_{P_0}, \Phi_{P_1}, \Phi_V$ тачне у валуацији β.

Према теореми 5.3.1 постоји успешна μ-анотација η за $\mathcal{G}_\sigma$. То значи да постоји функција β која додељује вредности променљивама x_p^v за све чворове игре која прати стратегију σ, дефинисана са $\beta(x_p^v) = \eta(v)^{(p)}$. Будући да је η успешна, имамо $\eta(w) \trianglelefteq_{\Omega(w)} \eta(v)$ за сваку грану $(v, w) \in E_\sigma$, што значи да је задовољена и последња потформула Φ_A формуле $\Phi_{\mathcal{G}}$.

($\Leftarrow$) Претпоставимо да постоји функција β која додељује вредности променљивава у формули $\Phi_{\mathcal{G}}$. Из формуле $\Phi_{\mathcal{G}}$ затим можемо конструисати игру $\mathcal{G}_\sigma = (V, E_\sigma, v_0, \Omega)$ следећим корацима:

- за сваки чвор $v \in V_{P_0}$ такав да је $\beta(S_v) = \top$ додати произвољну грану (v, w) у E_σ такву да је $\beta(T_{v,w}) = \top$, и
- за сваки чвор $v \in V_{P_1}$ такав да је $\beta(S_v) = \top$ додати све гране $(v, w) \in E$ у E_σ

Потформуле $S_{v_0}, \Phi_{P_0}, \Phi_{P_1}, \Phi_V$ осигуравају да одговарајуће гране потребне за конструкцију игре постоје и да је игра $\mathcal{G}|_\sigma$ направљена на овај начин заиста подигра игре $\mathcal{G}$.

Даље, можемо извести μ-анотацију η за $\mathcal{G}|_\sigma$ дефинисану са $\eta(v)^{(p)} = \beta(x_v^p)$ за свако $v \in V$ које прати стратегију σ и свако $p \in Odd(\mathcal{G})$. (Ако неко $\beta(x_v^p)$ буде негативно, то можемо лако поправити тако што ћемо прво увећати све $\beta(x_v'^{p'})$ за довољно велику корективну вредност o, и на тај начин учинити све целобројне вредности у нашем моделу позитивним. Формула Φ_A остаје задовољива и после ове корекције.). Потформула Φ_A осигурава да је η успешна анотација за $\mathcal{G}_\sigma$ и, према теореми 5.3.1, σ је победничка стратегија за играча P_0 у игри $\mathcal{G}$. □

Тврђење 5.5.1. *Ако имамо игру парности $\mathcal{G}_\sigma = (V, E_\sigma, v_0, \Omega)$, са највећим непарним приоритетом p_{max} величина формуле логике разлика $\Phi_{\mathcal{G}}$ је $O(|E| \cdot \lceil \frac{p_{max}}{2} \rceil)$.*

Анализом формуле $\Phi_{\mathcal{G}}$ увиђамо да наш алгоритам уме да је преведе у конјуктивну нормалну форму логике разлика без великог раста њене величине.

5.6 Кодирање у исказној логици

Након што смо приказали како се победничке стратегије дефинишу у логици разлика, преостаје нам да добијену формулу $\Phi_{\mathcal{G}}$ преведемо у формулу исказне логике. Потформуле $S_{v_0}, \Phi_{P_0}, \Phi_{P_1}, \Phi_V$ су већ у облику који нам је потребан, остаје нам само да преведемо потформулу Φ, тј. њену потформулу $\Psi_{v,w}$ која садржи поређење целобројних променљивих облика $(x_p^v \geq x_p^w)$ и $(x_{\Omega(w)}^v > x_{\Omega(w)}^w)$.

Нека је $\mathcal{G} = (V, E, v_0, \Omega)$ игра парности. Према последици 5.3.1 домен вредности променљивих x_p^v, за коначне вредности p и v, може бити ограничен

са $|V_p|+1$. Ако знамо да су вредности променљивих x_p^v ограничене то значи да тај број можемо приказати у битовској репрезентацији такође ограничене дужине. Нека је $m_p = \lceil \log(|V_p|+1) \rceil$ број битова потребан да се прикаже број вредности од 0 до $|V_p|$. Даље, сваки бит битовских репрезентација променљивих x_p^v можемо представити исказном променљивом $x_{p,i}^v$, за $i \in \{0, ..., m_p - 1\}$.

За свако $v, w \in V$, и било које $p \in Odd(\mathcal{G})$ и $m \geq 1$ дефинишемо рекурзивне формуле *ВећеИлиЈеднако* и *СтрогоВеће* са параметрима v, w, p, m:

$$
\begin{aligned}
&\mathit{ВећеИлиЈеднако}(v,w,p,0) &&= x_{p,0}^w \rightarrow x_{p,0}^v \\
&\mathit{ВећеИлиЈеднако}(v,w,p,m) &&= (x_{p,m}^w \rightarrow x_{p,m}^v) \wedge ((x_{p,m}^w \vee \neg x_{p,m}^v) \rightarrow \\
&&&\quad \mathit{ВећеИлиЈеднако}(v,w,p,m-1)) \\
&\mathit{СтрогоВеће}(v,w,p,0) &&= x_{p,0}^v \wedge \neg x_{p,0}^w \\
&\mathit{СтрогоВеће}(v,w,p,m) &&= (x_{p,m}^w \rightarrow x_{p,m}^v) \wedge ((x_{p,m}^w \vee \neg x_{p,m}^v) \rightarrow \\
&&&\quad \mathit{СтрогоВеће}(v,w,p,m-1))
\end{aligned}
$$

Из формула видимо да су поређења променљивих $(x_p^v \geq x_p^w)$ и $(x_{\Omega(w)}^v > x_{\Omega(w)}^w)$ у исказној логици представљена као поређења појединачних битова њихових битовски репрезентација, идући са лева на десно, од највиших битова до најнижих. Обе формуле су практично исте, са изузетком да формула *СтрогоВеће* у базном случају осигурава да је најнижи бит првог број $x_{p,0}^v$ већи од најнижег бита другог $x_{p,0}^w$.

Коначна формула $\Phi_\mathcal{G}$ се добија тако што се сваки бит битовских репрезентација целобројних променљивих замени исказним променљивама, а сва поређења целобројних променљивих својим адекватним функцијама:

- $(x_p^v \geq x_p^w)$ са $\mathit{ВећеИлиЈеднако}(v, w, p, m_p)$, и
- $(x_{\Omega(w)}^v > x_{\Omega(w)}^w)$ са $\mathit{СтрогоВеће}(v, w, \Omega w, m_{\Omega w})$

које се затим рекурзивно своде на формуле исказне логике као што је приказано у њиховој дефиницији.

Теорема 5.6.1. *Играч P_0 има победничку стратегију у игри $\mathcal{G}$ акко је формула исказне логике $\Phi_\mathcal{G}$ задовољива.*

Доказ. Следи из замене целобројних променљивих и ограничења њиховим одговарајућим истиносним променљивима, теореме 5.5.1 и последице 5.3.1. □

5.7 Детаљи имплементације

Алгоритам је имплементиран у програмском језику C++ и подељен је на неколико целина[1]. Сама логика алгоритма свођења игара парности на CAT, и главни део имплементације, налази се у датотекама `parity_games_cnf.h` и `parity_games_cnf.cpp`. У њима се налазе функције за прављење и поређење μ-анотација, као и део кода који конструише потребне формуле за кодирање победничке стратегије у логици разлика. Датотеке `jurdzinski.h` и `jurdzinski.cpp` садрже функцију која прави граф Јурџинског задате дубине и ширине. Датотеке `graph.h` и `graph.cpp` садрже класе које имплементирају чворове и гране графа игре, као и пратеће функције потребне за рад са њима. Датотеке `formula.h` и `formula.cpp` садрже имплементацију формула исказне логике (атома, везника и основних операција над њима) као и алгоритме за свођење формуле на КНФ, основни и Цејтинову трансформацију[2], и функцију за испис добијене формуле у dimacs-cnf формату.

Програм има два режима покретања. У првом корисник може да унесе име текстуалног фајла, у коме се налази опис графа који жели да тестира, као аргумент командне линије. Фајл треба да буде у следећем формату:

1, 3, 1 - чвор ознаке 1, приоритета 3, који припада играчу P_1

2, 2, 0 - чвор ознаке 2, приоритета 2, који припада играчу P_0

3, 3, 0

4, 2, 1

5, 1, 1

0, 1 - грана која повезује чворове ознака 0 и 1

0, 2 - грана која повезује чворове ознака 0 и 2

1, 4

2, 0

2, 3

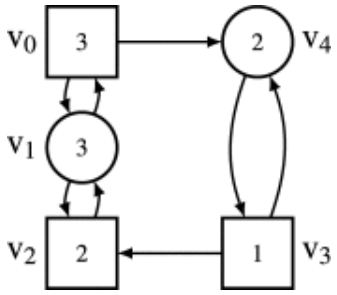
3, 2

4, 3

4, 1

[1]Имплементација је доступна на следећој адреси: https://github.com/IgorRodic/MasterRad-IgreParnosti

[2]Имплементација алгоритма Цејтинове трансформације ослања се на семинарски рад колегиница Александре Ђурић и Ане Ђорђевић "Цајтинова трансформација"на курсу Аутоматско резоновање, из 2016. године, из ког је преузет део програмског кода



Слика 5.2: Пример незадовољивог графа

Граф чији је опис приказан у примеру за улазни формат графа преузет је из рада [18] и пример је незадовољивости, јер у њему не постоји победничка стратегија за играча P_0 из v_0. Граф се може видети на слици 5.2. Пример покретања програма:

`./parity_games_cnf graf.txt` - за граф у фајлу graf.txt

Други режим користи се за тестирање програма над великим графовима. У ту сврху користе се графови Јурџинског $J_{d,w}$ који имају параметре d и w, дубину и ширину графа. Структура графа Јурџинског приказана је на слици 5.3. Параметри се уносе у програм као аргументи командне линије, прво d па w. Пример покретања:

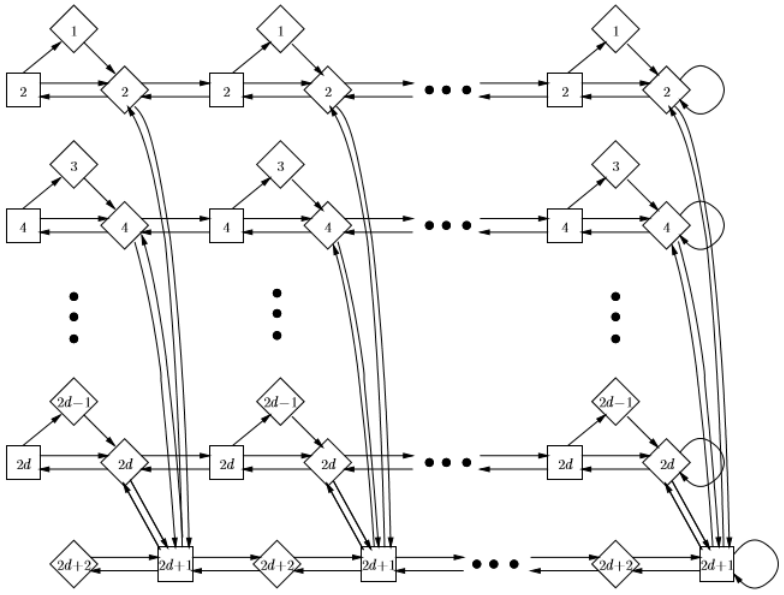
`./parity_games_cnf 3 4` - за граф Јурџинског дубине 3 и ширине 4.

Након покретања у једном од режима програм генерише фајл *formula.cnf* који представља формулу у конјуктивној нормалној форми у $DIMACS$ формату. Генерисани фајл затим можемо проследити произвољном SAT решавачу ради провере задовољивости. Један од SAT решавача који је коришћен ради тестирања алгоритма из нашег рада јесте minisat. Пример његовог покретања:

```
minisat formula.cnf solution.txt
```

У случају да је формула задовољива, на основу валуације која је пронађена као задовољавајућа, можемо одредити победничку стратегију за играча P_0. Чворови који су у валуацији означени са 1 су чворови који су укључени у подграф у ком играч P_0 има победничку стратегију, и све гране ће водити игру кроз те чворове. Гране које су у валуацији означене

са 1 су гране којима игра треба да се креће да би дошло до победе играча P_0. Дакле, када је играч P_0 на потезу треба да изабере за следећу излазну грану ону која је означена са 1 (у случају да их има више од једне свеједно је коју изабере). Избор играча P_1 није релевантан и све његове излазне гране су укључене.



Слика 5.3: Граф Јурџинског $J_{d,w}$

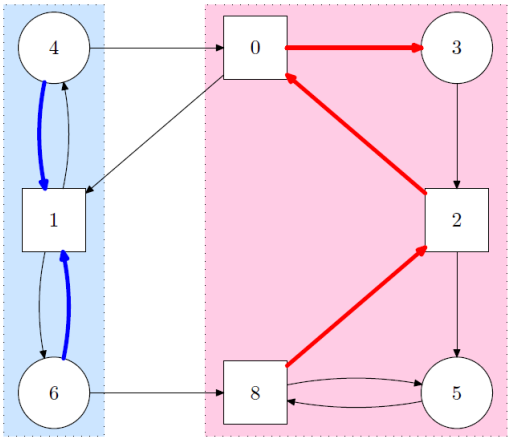
5.8 Валидација алгоритма

Валидација исправности алгоритма извршена је на 5 графова који су пронађени у литератури и за које је познато да ли постоји победничка стратегија за играча P_0, из чвора v_0, у њима. Чвор v_0 у сваком графу представља горњи леви чвор графа.

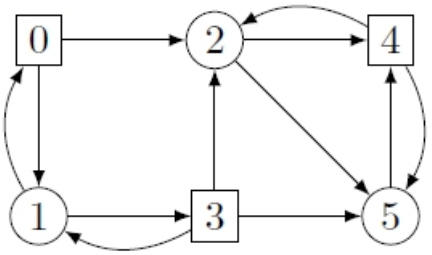
- Први граф преузет је из референтног рада овог поглавља [16] и приказан је на слици 5.1. У овом графу чвор v_0 припада играчу P_0 и овај граф је задовољив. Детаљно објашњење за то дато је у примеру 5.3.1
- Други граф већ је приказан на почетку ове тезе 3.4 и у нашем случају, пошто тражимо најмањи приоритет који се појављује бесконачно много пута, представља пример незадовољивости јер играч P_0 креће из чвора v_0 приоритета 4 и једини избор му је да даље пређе у чвор v_1 приоритета 3. Одатле се играч P_1 може поново вратити у чвор v_0 тако направивши

бесконачни циклус v_0, v_1, v_0... у коме је најмањи приоритет који се бесконачно пута појављује непаран, те у игри побеђује играч P_1

- Трећи граф приказан је на слици 5.4 и такође је пример незадовољивости. У овом графу чвор v_0 такође припада играчу P_0
- Четврти граф приказан је у секцији изнад као пример улазног формата графа и незадовољивости. Граф се може видети на слици 5.2 а чвор v_0 припада играчу P_1
- Пети граф приказан је на слици 5.5 и задовољив је. У овом графу чвор v_0 такође припада играчу P_1



Слика 5.4: Граф 3 - Пример незадовољивости



Слика 5.5: Граф 5 - Пример задовољивости

5.9 Резултати извршавања

Поредићемо два аспекта извршавања нашег алгоритма, време извршавања самог алгоритма, тј. генерисање формуле које је описано у овом поглављу и њено свођење на КНФ, и само време потребно SAT решавачу да одреди задовољивост тако добијене формуле. Времена извршавања тестирана су на графовима пронађеним у литератури који су наведени у претходној

секцији и графовима Јурџинског различитих величина, од $J_{5,5}$ до $J_{40,20}$, где игра почиње из чвора v_0.

Време извршавања алгоритма поредићемо на два имплементирана алгоритма свођења формула које су добијене на КНФ, основном алгоритму и Цејтиновој трансформацији. Основни алгоритам користи свођење формула на логички еквивалентне формуле елиминишући еквиваленције, импликације, дупле негације, користећи Де Морганове законе и закон дистрибутивности. Таква трансформација потенцијално производи формулу експоненцијалне величине, у односу на величину почетне формуле (2^n конјункта). Ефикаснији начин свођења јесте Цејтинова трансформација, она производи формулу линеарне величине у односу на величину почетне формуле [46].

Како је брзина модерних SAT решавача један од кључних разлога за саму идеју свођења решавања игара парности на SAT, упоредићемо брзине извршавања два SAT решавача, minisat-a, широко распрострањеног SAT решавача отвореног кода, и zChaff-a [34], развијеног на универзитету Принстон (енг. *Princeton University*), који се показао као ефикасан у пракси. Алгоритам је тестиран на још неколико SAT решавача, међу којима је и један од тренутно најбржих на свету MapleLCMDistChronoBT, али се њихове перформансе нису показале као задовољавајуће за наш проблем.

Тестирање је рађено на машини са процесором AMD Athlon x2 2GHz и 4GB RAM меморије. Поређење резултата извршавања алгоритма за графове из литературе приказано је у табели 5.1 док су резултати за графове Јурџинског приказани у табели 5.3. Основне карактеристике, односно број променљивих и број клауза, формула генерисаних Цејтиновом трансформацијом на графовима Јурџинског приказане су у табели 5.4. SAT решавачи поређени су за графове из литературе у табели 5.2, док су за графове Јурџинског поређени у табели 5.5. Времена су приказана у секундама. MEM OUT означава да рачунар није имао довољно RAM меморије за израчунавање дате формуле.

Времена која су потребна да би се генерисала формула која је описана у алгоритму из овог поглавља показују нам две ствари. Прво, Цејтинова трансформација се као метода свођења формуле на КНФ показала као вишеструко ефикаснија од основне интуитивне идеје, што је било очекивано и опште познато. И друго, чак и таква оптимизована трансформација нам не даје значајније резултате што се тиче брзине јер већ када се разматрају графови Јурџинског дубине веће од 30, и ширине веће од 10, извршавање

	Основни алгоритам	Цејтинова трансформација
Граф 1	0.0201	0.0089
Граф 2	0.0244	0.0098
Граф 3	0.0443	0.0144
Граф 4	0.0264	0.0038
Граф 5	0.0127	0.0095

Tabela 5.1: Времена генерисања формула на графовима из литературе

Граф 1		Граф 2		Граф 3		Граф 4		Граф 5	
minisat	zChaff	minisat	zChaff	minisat	zChaff	minisat	zChaff	minisat	zChaff
0.003	0.0001	0.0038	0.0001	0.0057	0.0002	0.0029	0.0000	0.0068	0.0002

Tabela 5.2: Времена извршавања SAT решавача над добијеним формулама на графовима из литературе

d	Јурџински $J_{d,5}$		Јурџински $J_{d,10}$		Јурџински $J_{d,15}$		Јурџински $J_{d,20}$	
	Основни	Цејтин	Основни	Цејтин	Основни	Цејтин	Основни	Цејтин
5	0.759	0.689	1.711	0.776	4.992	1.167	6.5473	5.712
10	6.668	1.692	14.384	3.356	39.365	5.087	53.4356	24.476
15	45.787	4.237	107.283	8.848	200.589	15.068	267.166	34.883
20	91.332	7.703	237.82	15.538	358.431	23.673	464.333	137.585
25	150.554	12.365	367.873	30.458	550.333	71.983	739.045	168.926
30	240.492	17.93	469.492	39.952	782.29	105.357	1050.74	125.027
35	847.485	41.324	1664.14	69.932	2504.82	169.518	3478.77	226.666
40	1082.95	48.788	2180.84	128.933	3285.61	222.703	4408.45	247.635

Tabela 5.3: Времена генерисања формула на графовима Јурџинског

траје дуже од 30 секунди, док некада прелази и неколико минута.

Из времена извршавања SAT решавача над добијеним формулама увидели смо следеће. Решавач zChaff генерално се показао као вишеструко бољи од решавача minisat, и често је решења налазио неупоредиво брже. Међутим, у неколико наврата показало се да и zChaff има своје слабости и да му неке формуле стварају проблеме, конкретни примери су $J_{15,15}$, $J_{15,20}$ и $J_{25,20}$ док за формуле $J_{35,20}$ и $J_{40,20}$ уопште није нашао решење јер није имао довољно RAM меморије на располагању. Из ових открића можемо закључити да сваки SAT решавач има своје предности и мане и да их треба бирати сходно потреби и хардверу који је на располагању. Друга ствар коју можемо да приметимо јесте да се времена извршавања на графовима Јурџинског већих ширина, преко 15, повећавају великом брзином и ова метода можда није погодна за примену на њиховом решавању.

d	Јурџински $J_{d,5}$		Јурџински $J_{d,10}$		Јурџински $J_{d,15}$		Јурџински $J_{d,20}$	
	Пром.	Клаузе	Пром.	Клаузе	Пром.	Клаузе	Пром.	Клаузе
5	13449	33733	27124	68133	40799	102533	54474	136933
10	57289	144508	115599	292058	173909	439608	232219	587158
15	146749	371143	296184	750313	445619	1129438	595054	1508653
20	253529	641518	511739	1297038	769949	1952558	1028159	2608078
25	388709	983593	784644	1988813	1180579	2994033	1576514	3999253
30	552589	1398268	1115499	2827438	1678409	4256608	2241319	5685778
35	877529	2224843	1771544	4499073	2665559	6773303	3559574	9047533
40	1138399	2886238	2298239	5836718	3458079	8787198	4617919	11737678

Tabela 5.4: Карактеристике формула генерисаних Цејтиновом трансформацијом на графовима Јурџинског

d	Јурџински $J_{d,5}$		Јурџински $J_{d,10}$		Јурџински $J_{d,15}$		Јурџински $J_{d,20}$	
	minisat	zChaff	minisat	zChaff	minisat	zChaff	minisat	zChaff
5	0.164	0.032	0.375	0.078	0.556	0.119	2.917	0.369
10	0.833	0.102	1.834	0.302	3.3	2.591	16.212	13.412
15	2.355	0.217	7.727	7.252	9.517	42.319	62.838	667.391
20	5.826	0.569	16.06	1.187	40.822	5.255	196.212	47.971
25	7.204	0.591	46.06	1.796	106.995	5.346	360.142	248.553
30	10.235	0.991	84.361	4.619	220.764	6.038	768.354	136.359
35	26.718	2.023	298.129	6.603	715.287	12.481	1365.44	MEM OUT
40	68.134	2.435	77.762	8.989	3285.61	16.242	2317.05	MEM OUT

Tabela 5.5: Времена извршавања SAT решавача над добијеним формулама на графовима Јурџинског

6. Закључци и даљи рад

Решавање игара парности представља важан проблем у теоријском рачунарству и налази своју примену у многим пољима као што су провера модела, верификација софтвера, реактивни системи и процедуре одлучивања. Такође, овај проблем припада у класи NP∩coNP и један је од ретких проблема у овој класи за који није пронађен алгоритам полиномске сложености. Сходно томе, проблем решавања игара парности интересантан је са алгоритамског становишта и током година научници су покушавали да га реше на што ефикаснији начин.

У овом раду приказане су игре парности и алгоритам за њихово решавање свођењем на SAT проблем. Основна идеја алгоритма заснива се на алгоритму малих мера напредовања Јурџинског који је он представио 2000. године. Мотивација за испитивање овакве идеје поред развоја све бржих SAT решавача, поготово у области провере модела, јесте и чињеница да оваква редукција није била истражена у прошлости и да она може да пружи нови увид у решавање игара парности.

Из постигнутих времена извршавања SAT решавача над формулама добијеним алгоритмом који је приказан можемо закључити неколико ствари. Чак и без оптимизација алгоритма он може да нам пружи резултате за дати проблем (и на слабијим рачунарима). Сваки SAT решавач има своје предности и мане, тј. формуле над којима ради брже или спорије, што значи да не можемо користити један SAT решавач и задовољити се његовим резултатима, морамо експериментисати са више њих и закључити који нам одговара највише у датој ситуацији. Графови Јурџинског већих ширина представљају временски комплексан задатак за решавање и ту се описана метода није показала као ефикасна.

Овај рад је показао да метода решавања игара парности свођењем на SAT има потенцијал за даљи развој. Он представља добру основу за будуће радове и оставља простора за напредак и даље развијање ове идеје. Могућа

унапређења могу бити: оптимизовање тренутног алгоритма, употреба система URSA (енг. *Uniform Reduction to SAT*) за кодирање SAT формуле [19], и анализа других сродних алгоритама и њиховог потенцијала за свођења на SAT. Највећи подстрек за даљи рад у овом смеру је развој све ефикаснијих SAT решавача, док би рад могао да се одведе и у другом смеру коришћењем SMT решавача над формулом логике разлика која представља међукорак нашег алгоритма.

Литература

[1] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proc 3rd FOCS*, pages 151–158, 01 1971. doi: 10.1145/800157.805047.

[2] Adam Antonik, Nathaniel Charlton, and Michael Huth. Polynomial-time under-approximation of winning regions in parity games. *Electr. Notes Theor. Comput. Sci.*, 225:115–139, 01 2009. doi: 10.1016/j.entcs.2008.12.070.

[3] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. A discrete subexponential algorithm for parity games. In Helmut Alt and Michel Habib, editors, *STACS 2003*, pages 663–674, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-36494-8.

[4] Patrick Blackburn, Johan F. A. K. van Benthem, and Frank Wolter. *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA, 2006. ISBN 0444516905.

[5] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 252–263, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4528-6. doi: 10.1145/3055399.3055409. URL http://doi.acm.org/10.1145/3055399.3055409.

[6] S. G. Cole. Conflict and cooperation in potentially intense conflict situations. *Journal of Personality and Social Psychology*, 22(1):31 – 50, 1972. doi: https://doi.org/10.1037/h0032390.

[7] A. Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The

MIT Press, 2nd edition, 2001. ISBN 0262032937. URL http://www.amazon.com/Introduction-Algorithms-Thomas-H-Cormen/dp/0262032937%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0262032937.

[9] Antonio Di Stasio, Aniello Murano, Giuseppe Perelli, and Moshe Vardi. Solving parity games using an automata-based algorithm. 9705, 07 2016.

[10] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.

[11] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32Nd Annual Symposium on Foundations of Computer Science*, SFCS '91, pages 368–377, Washington, DC, USA, 1991. IEEE Computer Society. ISBN 0-8186-2445-0. doi: 10.1109/SFCS.1991.185392. URL https://doi.org/10.1109/SFCS.1991.185392.

[12] E. Allen EMERSON. Chapter 16 - temporal and modal logic. In JAN VAN LEEUWEN, editor, *Formal Models and Semantics*, Handbook of Theoretical Computer Science, pages 995 – 1072. Elsevier, Amsterdam, 1990. ISBN 978-0-444-88074-1. doi: https://doi.org/10.1016/B978-0-444-88074-1.50021-4. URL http://www.sciencedirect.com/science/article/pii/B9780444880741500214.

[13] John Fearnley, Sanjay Jain, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. In *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*, SPIN 2017, pages 112–121, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5077-8. doi: 10.1145/3092282.3092286. URL http://doi.acm.org/10.1145/3092282.3092286.

[14] Oliver Friedmann and Martin Lange. Solving parity games in practice. In Zhiming Liu and Anders P. Ravn, editors, *Automated Technology for Verification and Analysis*, pages 182–196, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04761-9.

[15] Hugo Gimbert and Rasmus Ibsen-Jensen. A short proof of correctness of the quasi-polynomial time algorithm for parity games. 02 2017.

[16] Keijo Heljanko, Misa Keinänen, Martin Lange, and Ilkka Niemelä. Solving parity games by a reduction to sat. *Journal of Computer and System Sciences*, 78(2):430 – 440, 2012. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2011.05.004. URL http://www.sciencedirect.com/science/article/pii/S0022000011000535. Games in Verification.

[17] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, January 1985. ISSN 0004-5411. doi: 10.1145/2455.2460. URL http://doi.acm.org/10.1145/2455.2460.

[18] Michael Huth, Jim Huan-Pu Kuo, and Nir Piterman. The rabin index of parity games: Its complexity and approximation. *Information and Computation*, 245(Supplement C):36 – 53, 2015. ISSN 0890-5401. doi: https://doi.org/10.1016/j.ic.2015.06.005. URL http://www.sciencedirect.com/science/article/pii/S0890540115000723.

[19] Predrag Janicic. Ursa: A system for uniform reduction to sat. *Logical Methods in Computer Science*, 8, 12 2010. doi: 10.2168/LMCS-8(3:30)2012.

[20] Marcin Jurdzinski and Ranko Lazic. Succinct progress measures for solving parity games. *CoRR*, abs/1702.05051, 2017. URL http://arxiv.org/abs/1702.05051.

[21] Marcin Jurdziński. Deciding the winner in parity games is in up ∩ co-up. *Information Processing Letters*, 68(3):119 – 124, 1998. ISSN 0020-0190. doi: https://doi.org/10.1016/S0020-0190(98)00150-1. URL http://www.sciencedirect.com/science/article/pii/S0020019098001501.

[22] Marcin Jurdziński. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS 2000*, pages 290–301, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-46541-6.

[23] Marcin Jurdziński, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM Journal on Computing*, 38(4):1519–1532, 2008. doi: 10.1137/070686652. URL https://doi.org/10.1137/070686652.

[24] Gil Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 475–482, New York, NY, USA,

1992. ACM. ISBN 0-89791-511-9. doi: 10.1145/129712.129759. URL http://doi.acm.org/10.1145/129712.129759.

[25] Jeroen JA Keiren. Modal μ-calculus (version 1.1). 2013.

[26] Robert M. Keller. Formal verification of parallel programs. *Commun. ACM*, 19(7):371–384, July 1976. ISSN 0001-0782. doi: 10.1145/360248.360251. URL https://doi.org/10.1145/360248.360251.

[27] William Kirk and Brailey Sims. *Handbook of Metric Fixed Point Theory*. Springer Netherlands, 2001. ISBN 978-94-017-1748-9.

[28] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333 – 354, 1983. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(82)90125-6. URL http://www.sciencedirect.com/science/article/pii/0304397582901256. Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer 1982.

[29] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 224–233, New York, NY, USA, 1998. ACM. ISBN 0-89791-962-9. doi: 10.1145/276698.276748. URL http://doi.acm.org/10.1145/276698.276748.

[30] Martin Lange. Solving parity games by a reduction to sat. In *In Proc. of the Workshop on Games in Design and Verification, GDV'05*, 2005.

[31] L. A. Levin. Universal sequential search problems. In *Problems of Information Transmission, Probl. Peredachi Inf.*, volume 9, pages 115–116, 1973. URL http://mi.mathnet.ru/ppi914.

[32] Moez Mahfoudh, Peter Niebert, Eugene Asarin, and Oded Maler. A satisfiability checker for difference logic. In *FIFTH INTERNATIONAL SYMPOSIUM ON THE THEORY AND APPLICATIONS OF SATISFIABILITY TESTING*, 2002.

[33] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149 – 184, 1993. ISSN 0168-0072. doi: https://doi.org/10.1016/0168-0072(93)90036-D. URL http://www.sciencedirect.com/science/article/pii/016800729390036D.

[34] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pages 530–535, New York, NY, USA, 2001. ACM. ISBN 1-58113-297-2. doi: 10.1145/378239.379017. URL http://doi.acm.org/10.1145/378239.379017.

[35] Robert Nieuwenhuis and Albert Oliveras. Dpll(t) with exhaustive theory propagation and its application to difference logic. In Kousha Etessami and Sriram K. Rajamani, editors, *Computer Aided Verification*, pages 321–334, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31686-2.

[36] J. Obdržálek. *Algorithmic Analysis of Parity Games*. PhD thesis, University of Edinburgh, 2006. Submitted: January 31, 2006. Examined: May 29, 2006.

[37] Viktor Petersson and Sergei Vorobyov. A randomized subexponential algorithm for parity games, 2001.

[38] N. Piterman. From nondeterministic buchi and streett automata to deterministic parity automata. In *21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)*, pages 255–264, 2006. doi: 10.1109/LICS.2006.28.

[39] Jaco Pol and Michael Weber. A multi-core solver for parity games. *Electronic Notes in Theoretical Computer Science*, 220:19–34, 12 2008. doi: 10.1016/j.entcs.2008.11.011.

[40] Anuj Puri. *Theory of Hybrid Systems and Discrete Event Systems*. PhD thesis, University of California at Berkeley, 1995.

[41] Sven Schewe. Solving parity games in big steps. *Journal of Computer and System Sciences*, 84:243 – 262, 2017. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2016.10.002. URL http://www.sciencedirect.com/science/article/pii/S0022000016300952.

[42] Helmut Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59(6):303 – 308, 1996. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(96)00130-5. URL http://www.sciencedirect.com/science/article/pii/0020019096001305.

[43] L. S. Shapley. Stochastic games. *Proc. Nat. Acad Sci. U.S.A.*, 39:1095–1100, 1953.

[44] C. Stirling. Local model checking games. *In CONCUR '95*, 962(LNCS):1–11, 1995.

[45] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 06 1955.

[46] G. S. Tseitin. *On the Complexity of Derivation in Propositional Calculus*, pages 466–483. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983. ISBN 978-3-642-81955-1. doi: 10.1007/978-3-642-81955-1_28. URL https://doi.org/10.1007/978-3-642-81955-1_28.

[47] László Varga. The parity games. Master's thesis, Eötvös Loránd University. Faculty of Science, Budapest, 2014.

[48] Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bull. Belg. Math. Soc. Simon Stevin*, 8(2):359–391, 2001. doi: 10.36045/bbms/1102714178. URL https://doi.org/10.36045/bbms/1102714178.

[49] Ernst Zermelo and E Borel. On an application of set theory to the theory of the game of chess. In *Congress of Mathematicians*, pages 501–504. CUP, 1913.

[50] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1):135 – 183, 1998. ISSN 0304-3975. doi: https://doi.org/10.1016/S0304-3975(98)00009-7. URL http://www.sciencedirect.com/science/article/pii/S0304397598000097.

[51] U. Zwick and M. S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.