Latvijas Universitāte
Fizikas un Matemātikas fakultāte
Datorikas nodaļa

Valodu klašu apraksti ar loģikas līdzekļiem

Bakalaura darbs

Autors
Mārtiņš Kalvāns
Vadītājs
Rūsiņš Mārtiņš Freivalds
Habilitēts matemātikas doktors

Rīgā, 2006

Anotācija

Bakalaura darbā ir sniegts pārskats par galīgiem kvantu automātiem un to saistībai ar dažādiem matemātiskās loģikas veidiem. Darbā pastiprināta uzmanība tiek veltīta measure-many galīgiem kvantu automātiem. Tāpat arī tiek veikti secinājumi un minētas iespējamās tendences, uz ko varētu tiekties nākotnē.

Tiek arī aplūkota klasisko galīgo automātu un otrās kārtas loģikas saistības teorijas pētījumu rezultāti praktiskajā dzīvē, un to pielietojumi izstrādājot dažādus datoru programmatūras un aparatūras modeļus.

Abstract

Bachelors work contains an overview of quantum finite automata and their connection to several types of mathematical logic. We focus on measure-many quantum finite automata. Conclusions are made and future tendencies are noted.

Also the results of scientific work focusing on the connections between classical finite automata and second order logic and their practical uses in creating different software and hardware models is examined.

Περίληψη

Η εργασία αγάμων περιέχει μια επισκόπηση των κβαντικών πεπερασμένων αυτομάτων και της σύνδεσής τους σε διάφορους τύπους μαθηματικών λογικών. Εστιάζουμε σεπολλά κβαντικά πεπερασμένα αυτόματα. Τα συμπεράσματα συνάγονται και οι μελλοντικές τάσεις σημειώνονται.

Επίσης τα αποτελέσματα της επιστημονικής εργασίας που εστιάζουν στις συνδέσεις μεταξύ των κλασσικών πεπερασμένων αυτομάτων και της δεύτερης λογικής διαταγής και των πρακτικών χρήσεών τους των προτύπων στη δημιουργία του διαφορετικών λογισμικού και υλικού εξετάζονται.

Autoreferāts

Šajā darbā esmu aplūkojis saistības starp dažādām valodu klasēm un matemātiskās loģikas tipiem, kā arī teorētisko pētījumu rezultātu pielietojumus un rezultātus reālajā dzīvē.

Darbā esmu sniedzis ieskatu matemātiskajā loģikā un klasiskajos galīgajos kvantu automātos, kā galvenajiem rezultātiem klasisko galīgo automātu pētījumos ar loģikas līdzekļiem. Padziļinātu uzmanību esmu veltījis saiknes starp measure-many galīgu kvantu automātu un pirmās pakāpes loģikas pētījumu rezultātu analīzei un secinājumu veikšanai. Tāpat darbā padziļināti esmu aplūkojis klasisko galīgo automātu un otrās kārtas loģikas teorētisko rezultātu praktiskajiem pielietojumiem reālajā dzīvē.

Esmu sniedzis priekšlikumus tālākajiem virzieniem measure-many galīgu kvantu automātu izpētei ar loģikas paņēmieniem.

Saturs

Ievads

Algoritmu sarežģītības izteikšana sākās ar dabīgo lielumu laika un telpas aplūkošanu, t.i. laika un telpas patēriņš aprēķinu veikšanai. Par laimi, ar to algoritmu sarežģītības pētījumi neaprobežojās, un matemātiķi uzdeva jautājumu: „Kāda ir sarežģītība īpašības S *izteikšanai*?". Intuitīvi rodas mazas šaubas, ka šie raksturlielumi – aprēķinu sarežģītība un izteikšanas sarežģītība – ir savā starpā saistīti. Tomēr pārsteidzoša izrādījās saistība, kad īpašības izteikšana tiek aplūkota ar pirmās kārtas loģikas līdzekļiem. Daudzām sarežģītības klasēm, kuras ir definētas izmantojot laika un telpas aprakstus, ir precīzi aprakstāmas atbilstošas definīcijas izmantojot pirmās vai otrās kārtas loģiku.

Pirmie pētījumu rezultāti šajā nozarē tiek datēti ar 60. gadu sākumu, kad Büchi [6.], Elgot [11.] un Trakhtenbrot [12.] darbiem 60. gadu sākumā. Viņi parādīja, kā loģikas formulas var tikt pārveidotas galīgā automātā, kurš akceptē valodu, kuru apraksta attiecīgā formula, un otrādi. Vēlāk ekvivalenci starp galīgu automātu un otrās kārtas loģiku bezgalīgiem vārdiem un kokiem uzrādīja Büchi [7.], McNaughton [23.] un Rabin [27.] savos darbos. Nākošais sasniegums, saistības starp automātu teoriju un loģiku stiprināšanā, bija Pnueli darbs [24.], kurā tiek ierosināts lietot laika loģiku (Temporal Logic) spriedumos par nepārtrauktām paralēlām programmām. Tas noveda pie tā, ka 80. gados, „pagaidu" loģika un fiksētā-punkta loģika ieņēma galveno lomu valodu aprakstos, kā arī tika atrasti efektīgāki paņēmieni loģikas formulu transformēšanai uz automātiem. Līdz ar to var teikt, ka automātu teorijas un loģikas formālisma ekvivalences pētījumi ir ietekmējuši arī pašu automātu teoriju. Piemēram, automātu klases tika aprakstītas loģikas terminos. Pētījumu rezultāti atspoguļojās arī praktiskos pielietojumos – efektīgu algoritmu izveidē un pārbaužu datorprogrammu izstrādē.

Nesenā pagātnē tehnoloģiskie sasniegumi ir ļāvuši būvēt daudzprocesoru datorus un sistēmas, kuras sastāda ļoti liels daudzums paralēli strādājoši datori, ar vienu mērķi – risināt uzdevumus izmantojot iespējami mazus laika resursus. Tas noveda pie paralēlās skaitļošanas un paralēlā laika sarežģītības pastiprinātiem pētījumiem.

Skaitļošanas modeļu īpašību loģikas apraksti ietekmēja arī algoritmu sarežģītības teoriju. 1974. gadā Fagin [12.] deva aprakstu nedeterminētam

polinomiālam laikam (NP), kā kopai ar īpašībām, izsakāmību ar otrās kārtas eksistences loģikas paņēmieniem. Vēlāk Immerman [13.] un Vardi [30.] aprakstīja polinomiālu laiku, kā īpašību kopu, izteiktu ar pirmās kārtas indukcijas definīcijām, kuras ir definētas ar pirmās loģiku plus fiksētā punkta operatoru. Līdzīgā veidā tika aprakstīta arī polinomiāla telpa.

Darba pirmajā nodaļā tiek dots neliels ieskats pirmās un otrās kārtas loģikā. Otrajā nodaļa vispārīgi tiek aplūkoti klasiskie galīgi automāti, kā arī tiek pieminēti galvenie sasniegumi to izpētē ar matemātiskās loģikas līdzekļiem. Trešā nodaļa sniedz ieskatu galīgos kvantu automātos un sniedz measure-many galīgu kvantu automātu izpētes rezultātu aprakstu. Ceturtā nodaļa satur izklāstu par teorijas reālajiem pielietojumiem.

Ieskats loģikā

Matemātiskā loģika ir matemātikas nozare, kas nodarbojas ar kopēju struktūru veidošanu, lai ar formālām sistēmām aprakstītu intuitīvus pierādījumu un skaitļošanas konceptus. Matemātiskā loģika tiek uzskatīta par vienu no matemātikas pamatnozarēm.

Turpmāk nodaļā tiek aplūkotas pirmās kārtas loģikas un otrās kārtas loģikas definīcijas, kā arī valodu definēšana ar loģikas formulām.

Pirmās kārtas loģika

Pirmās kārtas loģika ir matemātiskās loģikas sistēma, kas paplašina izteikumu loģiku.

Definīcija 1. Par vārdnīcu jeb signatūru σ sauc galīgu relāciju un konstantu simbolu virkni:

$$\sigma = (R_1, \ldots, R_m, c_1, \ldots, c_n),$$

kur katru relācijas simbolu R_i asociē ar tā argumentu skaitu a_i.

Definīcija 2. Par struktūru *A* vārdnīcai σ sauc kortežu:

$$A = (A, R_1^A, \ldots, R_m^A, c_1^A, \ldots c_n^A),$$

kur

- A ir kopa, struktūras *A* universs,
- R_i^A ir šīs kopas relācija ar argumentu skaitu a_i, t.i., $R_i^A \subseteq A^{ai}$,
- c_i^A ir kopas A elements.

Definīcija 3. Mainīgo un konstanšu formulas ģenerē termus. Tas nozīmē:

- mainīgie un konstantes ir termi,
- ja f^n ir n-argumentu funkcija un $t_1, t_2, \ldots, t_n$ ir termi, tad $f^n = (t_1, t_2, \ldots, t_n)$ arī ir terms,
- citu termu nav.

Definīcija 4. Par pirmās kārtas loģikas formulu ar vārdnīcu σ sauc formulu, kas sastāv no vārdnīcā σ simboliem, bezgalīgas mainīgo kopas $\{v_1, v_2, ...\}$ un no loģikas simboliem, kas apmierina šādus nosacījumus:

- katrs mainīgais *x* un konstante ir terms,
- ja R ir relācijas simbols ar argumentu skaitu a un $t_1, ..., t_a$ ir termi, tad $R(t_1, ..., t_a)$ ir formula,
- ja φ un ψ ir formulas, tad arī $\varphi \wedge \psi$ un $\neg\varphi$ ir formulas,
- ja φ ir formula un *x* ir mainīgais, tad $\exists x\varphi$ arī ir formula.

Otrās kārtas loģika

Otrās kārtas loģika ir papildinājums pirmās kārtas loģikai. Otrās kārtas loģika satur mainīgos, kas apzīmē arī predikātus, un šo mainīgo eksistences un universālo kvantorus, atšķirībā no pirmās kārtas loģikas, kur mainīgie apzīmē tikai termus.

Definīcija 5. Par otrās kārtas loģikas formulu ar vārdnīcu σ sauc formulu, kas sastāv no vārdnīcas σ simboliem, bezgalīgas mainīgo kopas $\{v_1, v_2, ...\}$, bezgalīgas relāciju mainīgo kopas $\{V_1, V_2, ... \}$ un no dažādiem loģikas simboliem, kas apmierina šādus nosacījumus:

- katrs mainīgais *x* un konstante ir terms,
- ja R ir relācijas simbols ar argumentu skaitu a un t1, ..., ta ir termi, tad R(t1, ..., ta) ir formula,
- ja φ un ψ ir formulas, tad arī $\varphi \wedge \psi$ un $\neg\varphi$ ir formulas,
- ja φ ir formula un *x* ir mainīgais, tad $\exists x\varphi$ arī ir formula,
- ja X ir relācijas mainīgais ar argumentu skaitu a un $t_1, ..., t_a$ ir termi, tad $X(t_1, ..., t_a)$ ir formula,
- ja X ir relācijas mainīgais un φ ir formula, tad $\exists X\varphi$ arī ir formula.

Dažkārt tiek arī aplūkota pavājinātā otrās kārtas loģika, kuras formulu relāciju mainīgie ir unāri (viena argumenta relācijas).

Piemērs.

$$\forall F \; F(\text{persona}) \vee \neg F(\text{persona})$$

Formula apraksta sekojošo: katrai īpašībai vai nu tā piemīt personai vai arī ne.

Valodu definēšana izmantojot loģikas formulas

Aprakstot valodas ar alfabētu A vārdus, tos aplūko kā relācijas struktūras atbilstošajā vārdnīcā. Alfabētam $A=\{a_1, ..., a_s\}$ definēsim vārdnīcu $\sigma_A(<, P_{a1}, ..., P_{as})$, kur $P_{a1}, ..., P_{as}$ ir unitāri predikāti.

Definīcija 6. Par vārda $w \in A^*$ struktūru S_w sauc

$$s_w = (\{1, \ldots, n\}, <, P_{a1}, \ldots, P_{as}),$$

kur

- $<$ ir secības relācijas simbols kopā $\{1, ..., n\}$,
- $i \in P_a$ tad un tikai tad, ja vārda w i-tais simbols ir a.

Definīcija 7. Saka, ka formula φ vārdnīcā σ_A definē valodu $\{w \mid S_w \models \varphi\}$. Ja $p_1, ..., p_n$ ir pozīcijas vārdā w un $\varphi(x_1, ..., x_n)$ ir formula, tad $S_w, p_1, ..., p_n \models \varphi(x_1, ..., x_n)$ nozīmē, ka φ izpildās S_w, kad katru x_i aizstāj ar p_i.

Darbā tiek aplūkoti vārdu modeļus galīgā alfabētā A. Atbilstošajai pirmās kārtas valodai pieder mainīgie x, y, ... variējot pār pozīcijām vārdu modeļos, un ir veidoti no pamata formulām formā

$$x = y, \; S(x, y), \; Q_a(x), \; x < y$$

izmantojot saikļus $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ un kvantorus $\exists$ un $\forall$. Pieraksts $\varphi(x_1, x_2, ..., x_n)$ nozīmē, ka formulā φ nav citu brīvo mainīgo, kā tikai $x_1, ... x_n$, t.i. viņi nav sastopami kvantoros. Teikums ir formula bez brīvajiem mainīgajiem. Ja $p_1, p_2, ..., p_n$ ir pozīcijas burtu pozīcijas vārdā w, tad $(\underline{w}, p_1, p_2, ..., p_n) \models \varphi(x_1, x_2, ..., x_n)$ nozīmē, ka φ apmierina vārdu modeli $\underline{w}$ tad, kad $p_1, p_2, ..., p_n$ ir kā $x_1, x_2, ..., x_n$ interpretatori. Valoda, kas definēta ar teikumu φ ir $L(\phi) = \{w \in A^* \mid \underline{w} \models \varphi\}$. Ar šādu teikumu definētās valodas ir pirmās kārtas valodas. Piemēram, teikums $\forall x(Q_a(x))$ alfabētā $A = \{ a, b \}$ definē valodu, kura satur visus vārdus, kuri sastāv no burtiem a. Šī valoda ir pirmās kārtas valoda. Klasiskās ekvivalences rezultātus pirmās kārtas loģikā, ir aprakstījis Schuzenberg [28.].

Teorēma 1. Valodai $L \in A^*$ sekojošais ir ekvivalents:

- L ir „bez zvaigznes" (**Definīcija 16**),
- L var atpazīt galīgs neciklisks monoīds (galīgs monoīds M, kuram eksistē $n >= 1$ tā, ka $m^{n+1} = m^n$ ir patiess visiem $m \in M$),
- L ir definēta ar pirmās kārtas formulu.

Galīgi automāti (klasiskie)

Šajā nodaļā tiks aplūkoti klasiskie galīgie automāti – dotas to definīcijas un pieminēti nozīmīgākie sasniegumi klasisko galīgo automātu un loģikas sakarā.

Klasisko galīgo automātu definīcijas

Definīcija 8. Par galīgu determinētu automātu sauc elementu piecinieku

$$A = \{Q, \Sigma, q_0, [gamma], F\},$$

kur

- Q – galīga stāvokļu kopa,
- Σ – galīgs alfabēts,
- q_0 – sākuma stāvoklis ($q_0 \in Q$),
- F – akceptējošo stāvokļu kopa ($F \subseteq Q$),
- [gamma] – pārejas funkcija:

$$[gamma] : Q \times \Sigma \to Q.$$

Definīcija 9. Par determinēta automāta konfigurāciju sauc pārīti (q *w*), kur

- q – pašreizējais stāvoklis,
- *w* – vēl nenolasītā vārda daļa.

Definīcija 10. $\vdash_M : Q \times \Sigma^*$.

Definīcija 11. Saka, ka galīgs determinēts automāts A atpazīst vārdu *w*, ja $(s, w) \vdash_M (q\ \varepsilon)$, kur $q \in F$.

Definīcija 12. Saka, ka automāts A atpazīst valodu L, ja L = (*w* : A atpazīst *w*).

Definīcija 13. Valodas, ko atpazīst galīgs determinēts automāts, sauc par regulārām valodām.

Definīcija 14. Par atgriezenisku determinētu automātu A sauc galīgu determinētu automātu $A = (Q\ \Sigma\ q_0\ \delta\ F)$, kuram katram $q' \in Q$ un $\sigma \in \Sigma$ ir ne vairāk kā viens tāds stāvoklis q, ka $\delta(q'\ \sigma) = q$.

Definīcija 15. Par minimālo automātu valodai L sauc galīgu determinētu automātu, kas atpazīst valodu L, ar vismazāko stāvokļu skaitu.

Nozīmīgākie rezultāti

Aprakstot galīgus determinētus automātus ar loģikas līdzekļiem tiek iegūti divi nozīmīgi rezultāti, kas tieši attiecas uz atbilstošo valodu aprakstiem.

Definīcija 16. Par „bez zvaigznes" valodu klasi SF_Σ galīgā alfabētā Σ sauc mazāko klasi, kas apmierina nosacījumus:

- visas valodas, kas satur alfabēta Σ burtu, pieder SF_Σ,
- ja valodas L_1, L_2 ir „bez zvaigznes", tad tādas ir arī
  - $L1*L2$,
  - $L1 \cup L2$,
  - $L1 \cap L2$,
  - $L1 = A^* \setminus L$.

Piemērs. $\Sigma = \{a, b\}$, tad $a(ba)^* \in SF_\Sigma$, jo

$$a(ba)^* = a\ (\Sigma^* \setminus (a\Sigma^* \cup \Sigma^* b \cup \Sigma^* aa\Sigma^* \cup \Sigma^* bb\Sigma^*)).$$

Piemērs rāda, ka valoda, kas satur zvaigznītes operatoru „*", arī var būt „bez zvaigznes" valoda.

Teorēma 2. [31.] Valoda ir „bez zvaigznes" valoda tad un tikai tad, ja to ir iespējams definēt ar pirmās kārtas loģikas palīdzību.

Teorēma 3. [31.] Valoda ir „bez zvaigznes" valoda tad un tikai tad, ja tai atbilstošais galīgais monoīds ir neciklisks.

Teorēma 4. [7.] Galīgu vārdu valodu var atpazīt galīgs determinēts automāts tad un tikai tad, ja valodu var aprakstīt ar otrās kārtas vājās loģikas formulu.

Galīgi kvantu automāti un loģika

Klasisko aprēķinu ar galīgu atmiņu dabīgs modelis ir galīgs automāts, līdzīgi galīgs kvantu automāts ir dabīgs modelis kvantu aprēķiniem. Tiek lietoti dažādi galīgu kvantu automātu modeļi. Divi populārākie galīgu kvantu automātu modeļi ir measure-once galīgs kvantu automāts (Moore un Crutchfield [24.]) un measure-many galīgs kvantu automāts (Kondacs un Watrous [16.]). Šiem automātiem ir šķietami maza atšķirība. Pirmā automāta definīcija norāda, ka automāts veic mērījumu tikai aprēķinu beigās, savukārt, otra galīgā kvantu automāta definīcija norāda, ka automāts veic mērījumus katrā aprēķinu solī. Measure-once galīgs kvantu automāts un measure-many galīgs kvantu automāts ar izolētu griezuma punktu atpazīst tikai apakškopu no visām regulārajām valodām. Bez šiem galīgo kvantu automātu modeļiem, eksistē arī tādi galīgu kvantu automātu modeļi kā „paplašinātais" („enhanced") galīgs kvantu automāts [25.], Latviešu (Latvian) galīgs kvantu automāts [1.], vienvirziena (1-way) galīgs kvantu automāts ar kontroles valodu [1.], galīgs kvantu automāts ar jauktiem stāvokļiem [1.] un galīgs kvantu automāts gan ar kvantu stāvokļiem, gan ar klasiskajiem stāvokļiem [4.]. Salīdzinot ar klasiskajiem galīgiem automātiem, galīgiem kvantu automātiem ir gan savas priekšrocības, gan savi trūkumi. Galīgo kvantu automātu priekšrocība ir faktā, ka tas var būt ar eksponenciālu kārtu efektīgāks, bet galvenais trūkums rodas dēļ nepieciešamības kvantu procesam būt atgriezeniskam, tas arī padara lielāko daļu galīgo kvantu automātu spējīgus atpazīt tikai daļu no visām regulārajām valodām. Pie tam, daudziem galīgo kvantu automātu modeļiem joprojām nav pilnībā aprakstītas valodu klases, ko attiecīgie galīgie kvantu automāti ir spējīgi atpazīt. Šķiet pašsaprotami, ka arī šo problēmu ir lietderīgi mēģināt risināt aplūkot galīgo kvantu automātu atpazīstamās valodas no loģikas skatu punkta.

Saistība starp measure-once galīgiem kvantu automātiem un loģiku tiek pētīta Ilzes Dzelmes darbā „Quantum Finite Automata and Logics" [10.], kur tika parādīts, ka measure-once galīgi kvantu automāti neakceptē valodas, kuras definē pirmās kārtas loģika, izņemot triviālās valodas. Tas nozīmē, ka pirmās kārtas valodas ir daļa no regulārajām valodām, kas nevar tikt atpazītas ar measure-once galīgiem kvantu automātiem.

Daudz interesantāki un arī jaunāki rezultāti saistībā ar measure-many galīgiem kvantu automātiem un loģiku ir aprakstīti Ilzes Dzelmes publikācijā

[9.]. Ir parādīta pirmās kārtas loģikas formulu forma, kuru aprakstītajām valodām ir iespējams konstruēt atbilstošus measure-many galīgu kvantu automāts, spējīgus atpazīt attiecīgās valodas, un ir uzrādīta pirmās kārtas formulas konstrukcija, kurai nav atbilstoša measure-many galīga kvantu automāta, kurš šo valodu atpazītu. Turpmākajā darbā arī šiem rezultātiem tiks pievērta lielākā uzmanība.

Measure-once galīgs kvantu automāts

Definīcija 17. Measure-once galīgs kvantu automāts tiek definēts ar šādu elementu septiņnieku:

$$(Q,\ \Sigma,\ q_0,\ \delta,\ Q_{acc},\ Q_{rej},\ Q_{non}),$$

kur

- Q – galīga stāvokļu kopa,
- Σ – ieejas alfabēts,
- q_0 – sākuma stāvoklis ($q_0 \in Q$),
- δ – pārejas funkcija, $\delta : Q \times \Gamma \times Q \to C_{[0,1]}$, kur $\Gamma = \Sigma \cup \{ \#, \$ \}$ – darba alfabēts, kur # un $\$(\notin\Sigma)$ ir labais un kreisais ievadvārda beigu marķieris,
- $Q_{acc} \subseteq Q$ un $Q_{rej} \subseteq Q$ ir kopas ar attiecīgi akceptējošiem un noraidošajiem stāvokļiem ($Q_{acc} \cap Q_{rej} = \varnothing$),
- Q_{non} – darba stāvokļu kopa ($Q_{non} = Q / (Q_{acc} \cup Q_{rej})$).

Visiem stāvokļiem $q_1, q_2, q \in Q$ un simboliem $\sigma \in \Gamma$, funkcijai δ ir jābūt unitārai, līdz ar to jāapmierina nosacījums:

$$\Sigma_{q'}\delta^{t}(q_1,\ \sigma,\ q')\delta(q_2,\ \sigma,\ q') = \{1\ \ (q_1=q_2)\ \ |\ \ 0\ \ (q_1 \neq q_2)\}.$$

Pārejas funkciju δ attēlo ar unitāru matricu kopu $\{V_\sigma \vee \sigma \in \Gamma\}$, kur V_σ atbilst automāta A unitārai transformācijai pēc simbola σ nolasīšanas, un tiek definēta kā:

$$V_\sigma(|q\rangle) = \Sigma_{q' \notin Q}\delta(q,\ \sigma,\ q')|q'\rangle.$$

Measure-once galīgs kvantu automāts darbību sāk stāvoklī q_0 un veic unitāru transformāciju $V_{\#}$, kas atbilst sākuma marķierim #. Tālāk, pēc katra simbola σ nolasīšanas, tiek veikta unitāra transformācija V_σ, kas atbilst ieejas simbolam. Transformācijas rezultātā tiek iegūts jauns amplitūdas sadalījums $\psi=V_\sigma(\psi')$, kur ψ' ir iepriekšējā amplitūda.

Tiek pieņemts, ka vārds beidzas ar beigu marķiera simbolu, kuram atbilst sava unitārā transformācija. Pēc tā nolasīšanas veic mērījumu telpā $E_{acc} \oplus E_{rej} \oplus E_{non}$, kur $E_{acc} = span\{ |q>:q\in Q_{acc}\}$, $E_{rej} = span\{ |q>:q\in Q_{rej}\}$ un $E_{non} = span\{ |q>:q\in Q_{non}\}$. Mērījuma rezultātā iegūst vērtību $x\in E_i$, kas ir vienāda ar ψ amplitūdas projekciju telpā E_i jeb $||\psi||^2$ normas kvadrātu telpā E_i.

Measure-once galīgs kvantu automāts un pirmās kārtas loģika

Jau C. Moore un J. Churtchfield [24.] parādīja, ka measure-once galīgs kvantu automāts atpazīst regulāro valodu apakšklasi. I. Dzelme [9.] apraksta veidu, kā no measure-once galīga kvantu automāta var izveidot atbilstošu vājo otrās kārtas matemātiskās loģikas formulu.

Teorēma 5. [9.] Valodu kopas, ko atpazīst galīgs kvantu automāts ar mērījumu beigās, un valodu kopas, ko apraksta pirmās kārtas formulas, šėēlums satur tukšo kopu un Σ^* , un apvienojums nesatur visas regulārās valodas.

Measure-once galīgs kvantu automāts un otrās kārtas vājā loģika

Ir pieradīts, ka galīgs kvantu automāts ar mērījumu beigās atpazīst regulāro valodu apakškopu, līdz ar to valodas, ko atpazīst galīgais kvantu automāts ar mērījumu beigās var aprakstīt ar otrās kārtas vājo loģiku.

I. Dzelme [10.][9.] apraksta veidu, kā no measure-once galīga kvantu automāta var iegūt otrās kārtas vājās loģikas formulu, kas apraksta attiecīgo valodu.

Measure-many kvantu automāts

Definīcija 18. Measure-many galīgs kvantu automāts tiek definēts ar šādu elementu sešinieku [16.]:

$$A = (Q;\ \Sigma;\ d;\ q_0;\ Q_{acc};\ Q_{rej}),$$

kur

- Q ir galīga stāvokļu kopa,
- Σ ir ievada alfabēts un Γ = Σ ∪ { #; $ } ir automāta A darba alfabēts, kur # un $ ($\notin\Sigma$) ir labais un kreisais ievadvārda beigu marķieris,
- δ ir pārejas funkcija $\delta : Q \times \Gamma \times Q \rightarrow C_{[0,1]}$, kura attēlo amplitūdas, kas seko no stāvokļiem q uz stāvokli q' pēc simbola σ nolasīšanas,
- $q_0 \in Q$ ir sākuma stāvoklis,
- $Q_{acc} \subseteq Q$ un $Q_{rej} \subseteq Q$ ir kopas ar attiecīgi akceptējošiem un noraidošajiem stāvokļiem ($Q_{acc} \cap Q_{rej} = \varnothing$).

Stāvokļi Q_{acc} un Q_{rej} ir darbu pārtraucoši rezultāta stāvokļi, un stāvokļi $Q_{non} = Q \setminus (Q_{acc} \cup Q_{rej})$ ir darba stāvokļi.

Visiem stāvokļiem q_1, q_2, $q' \in Q$ un simboliem $\sigma \in \Gamma$, funkcijai δ jābūt unitārai, tas nozīmē, ka funkcija apmierina nosacījumu

$$\Sigma_{q'} \delta(q_1, \sigma, q') \delta(q_2, \sigma, q') = \{1 (q_1 = q_2) \mid 0 (q_1 \neq q_2)\}.$$

Ir pieņemts, ka ievada vārds sākas ar kreisās puses marķieri un beidzas ar labās puses marķieri.

Automāta A stāvokļu lineāra superpozīcija tiek attēlota ar n-dimensionālu kompleksu skaitļu vektoru, kur n = |Q|. Vektors tiek definēts kā $|\varphi\rangle = \Sigma_{i=1}^{n} \alpha_i |q_i\rangle$, kur $\{|q_i\rangle\}$ ir ortonormālo bāžu vektoru kopa, atbilstoši automāta A stāvokļiem.

Pārejas funkcija δ tiek attēlota ar kopu, kura sastāv no unitārām matricām $\{V_\sigma\}_{\sigma \in \Gamma}$, kur V_σ ir automāta A unitāra transformācija pēc simbola σ nolasīšanas un ir definēta kā $V_\sigma(|q\rangle) = \Sigma_{q' \in Q} \delta(q, \sigma, q') |q'\rangle$.

Automāta A darbība ieejas vārdam $\#\sigma_1, \sigma_2, \ldots \sigma_n\$$ notiek šādi. Darbība tiek sākta superpozīcija $|q_0\rangle$. Tad notiek transformācija atbilstoši ieejas burtam. Pēc katras transformācijā automāts A mēra tā stāvokli ņemot vērā $E_{acc} \oplus E_{rej} \oplus E_{non}$, kur $E_{acc} = span\{ |q\rangle : q \in Q_{acc}\}$, $E_{rej} = span\{ |q\rangle : q \in Q_{rej}\}$ un $E_{non} = span\{$

$|q\rangle : q \in Q_{non}\}$. Ja automāta A iegūtais stāvoklis ir E_{acc}, tad tas akceptē ievada vārdu; ja iegūtais stāvoklis ir E_{rej}, tad tas noraida. Pārējos gadījumos skaitļošana tiek turpināta. Pēc katra mērījuma veikšanas superpozīcija kolapsē līdz mērījuma apakštelpai. Mērījums tiek attēlots ar diagonāles nuļļu un vieninieku projekcijas matricām P_{acc}, P_{rej} un P_{non}, kuras projicē vektoru uz E_{acc}, E_{rej} un E_{non}.

Tā kā automāts A var saturēt ne-nulles darba beigšanas varbūtību, ir lietderīgi sekot kopīgajām darba akceptējošajām un noraidošajām varbūtībām. Tādēļ automāta A stāvokļi tiek attēloti ar elementu trijnieku (φ, p_{acc}, p_{rej}), kur p_{acc} un p_{rej} ir automāta kopīgās akceptējošās un noraidošās varbūtības. Automāta A pāreja pēc burta σ nolasīšanas tiek apzīmēta ar

$$(P_{non}|\varphi'\rangle, p_{acc}+||P_{acc}\varphi'||^2, p_{rej}+P_{rej}\varphi'||^2),$$

kur $\varphi' = V_\sigma\varphi$.

Measure-many galīgi kvantu automāti un pirmās kārtas loģika

Šajā nodaļā tiek aplūkota saistība starp pirmās kārtas loģiku un measure-many galīgiem kvantu automātiem. Ir parādīts [9.], ka valodas, kuras apraksta pirmās kārtas loģikas formulas, nevar atpazīt measure-once galīgi kvantu automāti un otrādāk, izņemot triviālās valodas. Tā kā measure-many galīgs kvantu automāts atpazīst visas valodas, kuras atpazīst measure-once galīgs kvantu automāts, tad ir skaidrs, ka eksistē tādas valodas, kuras tiek atpazītas ar measure-many galīgiem kvantu automātiem, bet nav aprakstāmas ar pirmās kārtas loģiku. Piemēram, valoda, kura satur visus vārdus, kuru garums ir skaitļa „3" daudzkārtnis, ir atpazīstams ar measure-many galīgu kvantu automātu, bet nav aprakstāms ar pirmās kārtas loģikas formulām.

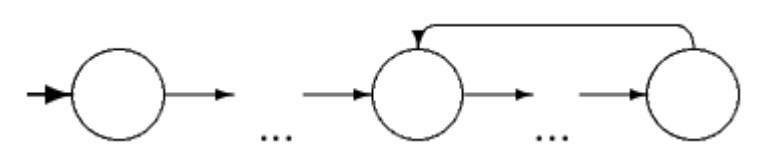
Nākamais loģiskais jautājums ir, vai eksistē valoda, kas ir definēta ar pirmās kārtas loģikas formulām, bet, kuru nevar atpazīt ar measure-many galīgu kvantu automātu? Jā, šādas valodas eksistē. Ir zināms [16.], ka measure-many galīgs kvantu automāts nevar atpazīt regulāro valodu {{a, b}*b}, kaut arī tā ir aprakstāma ar šādu pirmās loģikas formulu:

$$\forall x(last(x) \supset Q_b(x)).$$

Eksistē arī valodas, kuras var aprakstīt ar pirmās kārtas formulām un kuras var atpazīt ar measure-many galīgu kvantu automātu, piemēram, valoda, kura satur vārdus, kuri sākas ar burtu *a*. Ņemot vērā šos faktus, tiek izvirzīts mērķis: aprakstīt pirmās kārtas loģikas formulas, kurām eksistē atbilstošs measure-many galīgs kvantu automāts, kurš atpazīst atbilstošo valodu, un aprakstīt pirmās kārtas loģikas formulas, kurām šādu automātu konstruēt nav iespējams.

6. teorēma. Viena burta alfabēta valodas, kuras ir definētas ar pirmās kārtas formulām, var atpazīt ar measure-many galīgiem kvantu automātiem, bet ne otrādāk.

Pierādījums. Galīgs determinēts automāts viena burta alfabēta valodai ir parādīts zīmējumā:



Figūra 1. DFA valodai viena burta alfabētā

Ir iespējams konstruēt measure-many galīgu kvantu automātu šādai valodai. Regulāra valoda viena burta alfabētā var tikt uzdota kā $a^k a^{nm}$, kur k ir konstantās daļas garums un m ir cikla garums. Measure-many galīgam kvantu automātam ir k+m+1 darba stāvokļi (k+1 stāvokļi ir pirmajiem k burtiem un m ir ciklam) un k+m+3 ir darbu beidzošie stāvokļi. Pārejas funkcija tiek definēta šādi:

$$V_{\#}(|q_0\rangle)=2/3|q_0\rangle+1/3|q_{k+1+m-k(\mathrm{mod}\ m)}\rangle \quad (k \neq m),$$

$$V_{\#}(|q_0\rangle)=2/3|q_0\rangle+1/3|q_{k+1}\rangle \quad (k = m),$$

$$V_a(|q_i\rangle)=|q_{i+1}\rangle, \text{ kur } i\in\{0, 1, \ldots, k-1, k+1, k+m-1\},$$

$$V_a(|q_k\rangle)=1/2|q_{2k+2m+2}\rangle+1/2|q_{2k+2m+3}\rangle,$$

$$V_{\$}(|q_{k+m}\rangle)=|q_{k+1}\rangle,$$

$$V_{\$}(|q_i\rangle)=|q_{i+k+m+1}\rangle \quad (i\in\{0, 1, \ldots, k+m\}).$$

$|q_{2k+2m+2}>$ ir akceptējošais stāvoklis, $|q_{2k+2m+3}>$ ir noraidošais stāvoklis, ja $|q_i>$ ($i<k+1$) ir akceptējošais stāvoklis determinētajā galīgajā automātā, tad $|q_{i+k+m+1}>$ ir akceptējošais stāvoklis measure-many galīgajā kvantu automātā un otrādāk. Ja $|q_i>$ ($i > k$) ir akceptējošais stāvoklis determinētajā galīgajā automātā, tad $|q_{i+k+m+2}>$ ir akceptējošais stāvoklis measure-many galīgajā kvantu automātā un otrādāk.

Aplūkojam vārdu $w \in L$, kurš tiek akceptēts ar measure-many galīgu kvantu automātu. Var izšķirt divus gadījumus –

- vārda w garums nav garāks par k, tad w tiks akceptēts ar varbūtību vismaz 2/3. Tas nozīmē, ka pēc vārda w nolasīšanas measure-many galīgs kvantu automāts ar varbūtību 2/3 būs stāvoklī $q_{|w|-1}$, un pēc beigu simbola nolasīšanas automāts ies no stāvokļa $q_{|w|-1}$ uz akceptējošo stāvokli ar varbūtību 2/3.

- ja vārda w garums ir lielāks par k, tad w tāpat tiks akceptēts ar varbūtību, ne mazāku par 2/3. Pēc pirmo k burtu nolasīšanas measure-many galīgs kvantu automāts ir stāvoklī q_{k+1} ar varbūtību 1/3, pēc nākošā burta nolasīšanas automāts nonāk stāvoklī $q_{k+1+(|w|-k)(\mathrm{mod}\ m)}$ ar varbūtību 1/3 un tas ir akceptējošais stāvoklis $|q_{2k+2m+2}>$ ar varbūtību 1/3. Kad automāts nolasa beigu simbolu, vārds tiek akceptēts ar varbūtību vismaz 2/3.

Ja vārds $w \notin L$, tad, viegli redzēt, ka measure-many galīgs kvantu automāts to noraidīs ar varbūtību vismaz 2/3.

Tā kā valodas, kuras var definēt ar pirmās kārtas loģiku, ir apakškopa no visām regulārajām valodām, tad measure-many galīgs kvantu automāts var atpazīt valodas viena burta alfabētā, kuras ir definētas ar pirmās kārtas loģikas palīdzību. Visas regulārās valodas viena burta alfabētā nav definējamas ar pirmās kārtas loģikas paņēmieniem, tas nozīmē, ka visas valodas viena burta alfabētā, kuras atpazīst measure-many galīgs kvantu automāts, nevar definēt ar pirmās kārtas loģikas paņēmieniem. To pierāda sekojošā lemma.

1. lemma. Viena burta alfabēta valodas ir definējamas ar pirmās kārtas loģikas formulām, tad un tikai tad, ja tām atbilstošais mazākais determinētais galīgais automāts satur ciklu garumā 1.

Pierādījums. Ir viegli konstruēt pirmās kārtas loģikas formulu, kas definē valodu, kura tiek atpazīta ar determinētu galīgu automātu, kurš satur ciklu garumā 1. Sanumurēsim galīgā determinētā automāta stāvokļus tā, ka q_0 ir sākuma stāvoklis, no stāvokļa q_i automāts nonāk stāvoklī q_{i+1}. Katram akceptējošajam stāvoklim formula tiek veidota šādā veidā –

- ja stāvoklis q_i nav pēdējais ($i \neq n$, $n+1$ – stāvokļu skaits), tad vārdi, kuri tiek akceptēti stāvoklī q_i var tikt definēti ar pirmās kārtas loģikas formulu

$$\forall x_1 x_2 \ldots x_i \, (first(x_1) \supset S(x_1, x_2) \supset \ldots \supset S(x_{i-1}, x_i) \supset last(x_i))$$

- ja stāvoklis q_n ir pēdējais stāvoklis, tad vārdi, kuri tiek akceptēti stāvoklī q_n tiek definēti ar pirmās kārtas loģikas formulu

$$\forall x_1 x_2 \ldots x_n \, (first(x_1) \supset S(x_1, x_2) \supset \ldots \supset S(x_{n-1}, x_n))$$

Sanumurējot visas akceptējošo stāvokļu formulas φ no 1 līdz j, iegūst, ka pirmās kārtas loģikas formula

$$\varphi_1 \vee \varphi_1 \vee \ldots \vee \varphi_j$$

definē valodu, kura tiek akceptēta ar šo galīgo determinēto automātu.

Pieņemsim no pretējā, ka ir iespējams konstruēt pirmās kārtas formulu arī priekš galīga determinēta automāta viena burta alfabētā, kurš satur ciklu ar garumu lielāku par 1. Lietosim tos pašus apzīmējumus, ko iepriekšējā pierādījumā – regulāra valoda viena burta alfabētā tiek uzdota kā $a^k a^{nm}$, kur k ir konstantās daļas garums un m ir cikla garums. Monoīds M, kurš atpazīst valodu ir $M = \{1_M, \delta_a, \delta_{a2}, \ldots, \delta_{ak}, \delta_{ak+1}, \ldots, \delta_{ak+m-1}\}$, un šim monoīdam ir apakšgrupa $\{1_M, \delta_{ak}, \delta_{ak+1}, \ldots, \delta_{ak+m-1}\}$. Tātad, monoīds nav neciklisks un pieņēmums ir aplams. Nav iespējams konstruēt pirmās kārtas formulu galīgam determinētam automātam viena burta alfabētā, kurš satur ciklu ar garumu lielāku par 1.

2. lemma. Valodas, kuras definētas ar pirmās kārtas loģikas formulām formā

$$\forall x \, (Q_\sigma(x)) \; (\sigma \in \Sigma)$$

var atpazīt ar measure-many galīgu kvantu automātu.

Pierādījums. Measure-many galīgs kvantu automāts, kurš atpazīst valodu, kura ir definēta ar formulu formā $\forall x(Q_\sigma(x))$ $(\sigma\in\Sigma)$, sastāv no trīs stāvokļiem – diviem beigu stāvokļiem $\{q_1, q_2\}$, kur q_1 ir akceptējošais stāvoklis un q_2 noraidošais stāvoklis un viens neterminālais stāvoklis q_0, kurš ir arī sākuma stāvoklis. Pārejas funkcija sākuma marķierim un σ ir definēta ar vienības matricu, transformācijas funkcija burtam $\gamma(\gamma\in\Sigma$ un $\gamma\neq\sigma)$ tiek definēta ar matricu

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

un transformācijas funkcija beigu simbola marķierim tiek definēta ar matricu

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

3. lemma. Valodas, kuras ir definētas ar pirmās kārtas loģikas formulu formā

$$\forall x_1x_2...x_n(first(x_1)\supset\varphi_1\supset S(x_1,x_2)\supset\varphi_2\supset S(x_2,\ x_3)\supset...$$

$$...\supset S(x_{n-1},x_n)\supset\varphi_n(\sigma\in\Sigma)$$

kur σ_i ir formā

$$Q_{\sigma1}(x_i)\vee Q_{\sigma2}(x_i)\vee...\vee Q_{\sigma s}(x_i)$$

$(\sigma_j\in\Sigma,\ i\in\{1, 2, ..., n\text{-}1\})$ un φ_n ir formā

$$Q_{\sigma1}(x_n)\vee Q_{\sigma2}(x_n)\vee...\vee\ Q_{\sigma s}(n_i)\,[\wedge last(x_n)\,](\sigma_j\in\Sigma),$$

var atpazīt ar measure-many galīgu kvantu automātu.

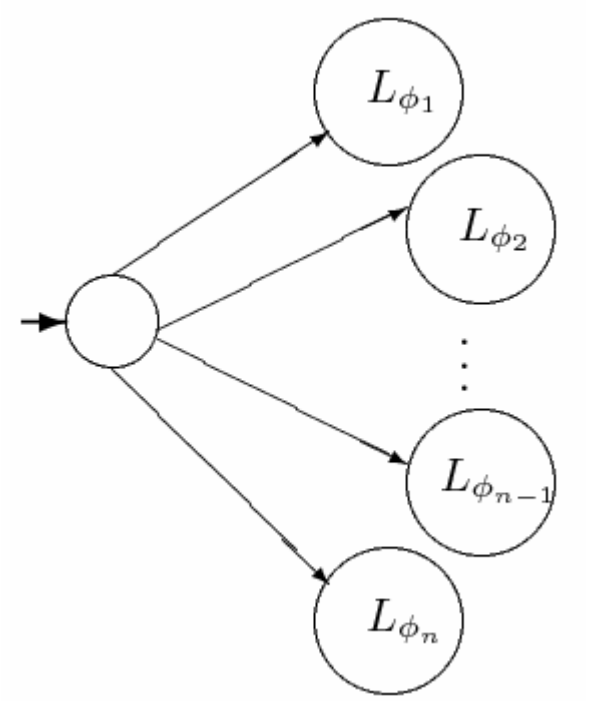
Pierādījums. Measure-many galīgs kvantu automāts, kurš akceptē valoda, kura ir definēta ar doto pirmās kārtas loģikas formulu, tiek definēts šādi:

$$A = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej}),$$

kur $Q = \{q_0, q_1, \ldots, q_{2n+1}, q_{2n+2}\}$, ja formula φ_n satur *last*(x_n), tad $Q_{acc} = \{q_{2n+2}\}$, pretējā gadījumā $Q_{acc} = \{q_n, q_{2n+2}\}$, $Q_{rej} = \{q_{n+1}, \ldots, q_{2n+1}\}$, transformācija sākuma marķierim ir definēta ar vienības matricu. Burtam σ_i, ja formula φ_j satur $Q_{\sigma i}(x_i)$, tad $V_{\sigma i}(|q_{j-1}\rangle) = |q_j\rangle$, pretējā gadījumā $j<n+1$, $V_{\sigma i}(|q_{j-1}\rangle) = |q_{n+j}\rangle$. Ja φ_n satur *last*(x_n), tad $V_\sigma(|q_n\rangle) = |q_{2n+2}\rangle$.

7. teorēma. Valodām, kuras tiek definētas ar pirmās kārtas formulu formā $\varphi_1 \vee \varphi_2 \vee \ldots \vee \varphi_n$, kur φ ir pirmās kārtas loģikas formula no 3. lemmas, var konstruēt measure-many galīgu kvantu automātu, kurš spēj atpazīt šīs valodas.

Pierādījums. Formulas definētā valoda L ir apvienojums no formulu φ_i definētajām valodām L_i. Galīgs nedeterminēts automāts, kurš atpazīst valodu L, ir parādīts zīmējumā, kur $L_{\varphi i}$ ir valoda, kas ir definēta ar pirmās kārtas loģikas formulu φ_i un zīmējumā tiek lietota, lai apzīmētu galīgu determinētu automātu, kurš atpazīst valodu $L_{\varphi i}$.



Figūra 2 Galīgs nedeterminēts automāts

Viegli redzēt, ka determinētais automāts, kurš atpazīst valodu ir reversējams automāts, un izmantojot 1. teorēmu [1.] tiek iegūts pierādījums.

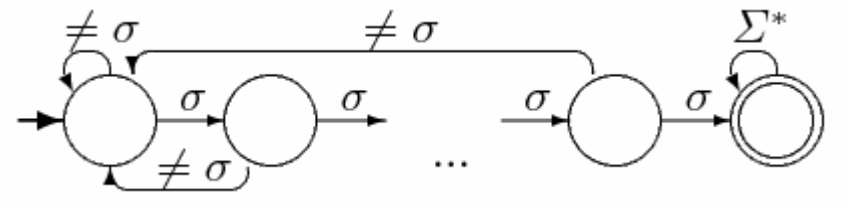
8. teorēma. Valoda, kuru definē ar pirmās kārtas loģikas formulu

$$\exists x_1 x_2 \ldots x_n (Q_{\sigma 1} \supset S(x_1, x_2) \supset Q_{\sigma 2} \supset S(x_2, x_3) \supset \ldots$$

$$\ldots \supset S(x_{n-1}, x_n) \supset Q_{\sigma n}),$$

kur $\sigma_j \in \Sigma$ un $n > 2$, nevar atpazīt ar measure-many galīgu kvantu automātu.

Pierādījums. Vispirms pieņemot, ka $\sigma_i = \sigma_j$ visiem i un j, iegūst atbilstošo galīgo determinēto automātu, kurš ir attēlots zīmējumā.



Figūra 3 Galīgs determinēts automāts

Tā kā galīgais determinētais automāts satur „aizliegto" konstrukciju [1.], tad valoda, kuru definē formula

$$\exists x_1 x_2 \ldots x_n (Q_{\sigma 1} \supset S(x_1, x_2) \supset Q_{\sigma 2} \supset S(x_2, x_3) \supset \ldots$$

$$\ldots \supset S(x_{n-1}, x_n) \supset Q_{\sigma n})$$

nav atpazīstama ar measure-many galīgu kvantu automātu.

Aplūkosim gadījumu, kad eksistē σ_i tāds, ka $\sigma_1 \neq \sigma_i$. Sanumurējam minimālā galīgā determinētā automāta stāvokļus tā, ka automāts no stāvokļa q_i nonāk stāvoklī q_j pēc simbola σj nolasīšanas. Pieņemsim, ka pirmais simbols formulā, kurš atšķiras no σ_1, ir σ_i. Tas nozīmē, ka pēc simbola σ_1 nolasīšanas automāts nonāk no stāvokļa q_{i-1} uz stāvokli q_{i-1} un no stāvokļa q_{i-2} uz stāvokli q_{i-1}. Ja $\sigma_i = \sigma_j$, tad automāts pēc burta σ_1 nolasīšanas nonāk no stāvokļa q_i stāvoklī q_{i-1}. Līdz ar to mēs iegūstam automātu ar „aizliegto" konstrukciju. Ja σ_i != σ_j, tad automāts pēc simbola σ_i nolasīšanas nonāk no stāvokļa q_i stāvoklī q_0 un mēs atkal iegūstam „aizliegto" konstrukciju.

Measure-many galīgs kvantu automāts un vājā otrās kārtas loģika

Tā kā measure-many galīgs kvantu automāts, līdzīgi kā measure-once galīgs kvantu automāts, spēj atpazīt tikai visu regulāro valodu apakškopu, tad seko, ka arī šo valodu klasi var aprakstīt ar vājo otrās kārtas loģiku. Un, lai konstruētu otrās kārtas loģikas formulu, kas apraksta attiecīgo valodu, var izmantot tieši to pašu metodi, kas ir aprakstīta measure-once galīga kvantu automāta gadījumā.

Rezultātu analīze

Pašreizējie rezultāti, aprakstot measure-many kvantu automātus ar pirmās kārtas loģikas līdzekļiem, ļauj secināt vairākas būtiskas lietas. Ir noskaidrots, ka measure-many kvantu automātu atpazīstamo valodu kopa šķeļas ar valodu kopu, ko var aprakstīt ar pirmās kārtas loģikas formulām. Tas nozīmē, ka eksistē gan valodas, kuras var aprakstīt ar measure-many galīgu kvantu automātu, bet nav aprakstāmas ar pirmās kārtas loģikas formulām, gan valodas, kuras ir aprakstāmas ar pirmās kārtas loģikas formulām, bet nav aprakstāmas ar measure-many galīgu kvantu automātu, gan arī tādas valodas, kuras ir aprakstāmas gan ar measure-many galīgu kvantu automātu, gan ar pirmās kārtas loģikas formulām. Analogi rezultāti ir aprakstīti [9.] saistībā starp measure-many galīgu kvantu automātu un modulāro loģiku un modulāro loģiku + pirmās kārtas loģiku. Viegli redzēt, ka joprojām atliek daudz neatbildētu jautājumu. Vai eksistē loģikas veids, ar kuru var aprakstīt valodas, kuras atpazīst measure-many galīgs kvantu automāts. Vai arī pirmajā tuvinājumā, vai eksistē loģikas veids, ar kuru var aprakstīt visas regulārās valodas, kuras atpazīst measure-many galīgs kvantu automāts un vai eksistē loģikas veids, ar kuru var aprakstīt valodu kopu, kura nesatur regulārās valodas, bet ir atpazīstamas ar measure-many galīgu kvantu automātu.

Kamēr nebūs rastas atbildes uz šiem jautājumiem, būs arī apgrūtināta measure-many galīgu kvantu automātu sarežģītības mēru pētīšana, izmantojot loģikas paņēmienus.

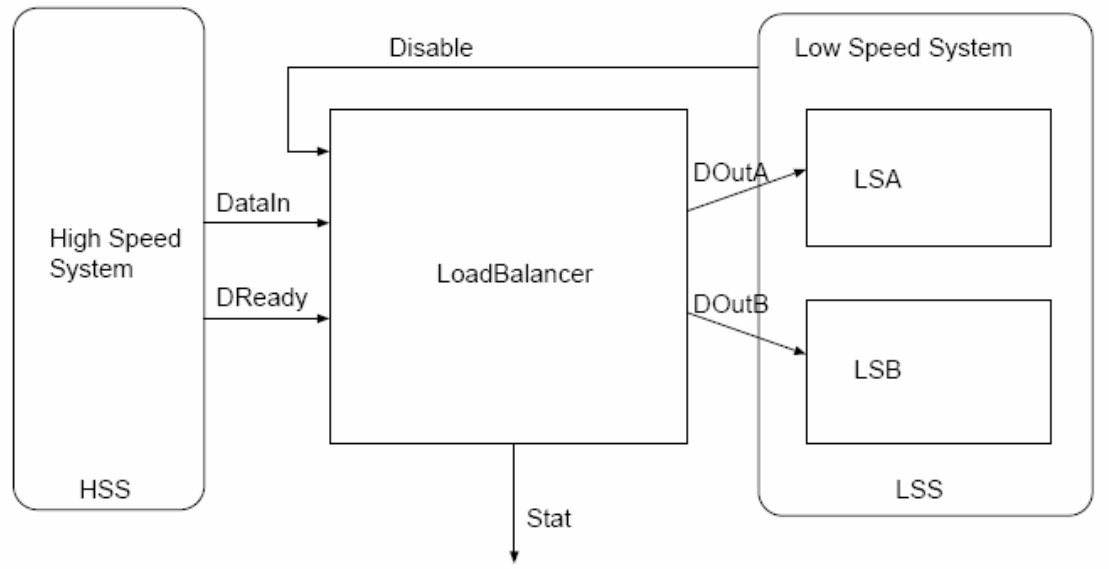
Praktiski rezultāti uz modeļiem bāzētu automātu sintēzē un analīzē izmantojot pavājinātu otrās kārtas loģiku

Šajā nodaļā ir aprakstīts, kā pavājināto otrās kārtas loģiku var lietot, kā aprakstošo līdzekli uz modeļiem bāzētu datorprogrammu un aparatūras sistēmu analīzē [22.]. Šajā gadījumā loģika, gan kā valodas specifikācija, gan kā programmēšanas valoda, veiksmīgi apvieno divas lietas vienā formālā rezultātā. Katra šāda specifikācija valodu var aprakstīt ar tai ekvivalentu galīgu automātu, līdz ar to tā iegūst uzreiz realizējamu veidolu.

Praktiskais pielietojums

Tehniskā attīstība ir ļāvusi izveidot gigantiskas multiprocesoru sistēmas, kas spēj risināt daudzas problēmas cilvēkam pieņemamā laika periodā, bet kuras šobrīd vēl nespēj risināt viena procesora sistēmas saprātīgā laika periodā. Multiprocesoru sistēmu var sadalīt divās daļās, no kurām viena problēmas pārraudzības daļa (High-Speed system – HSS), kas ģenerē kopējās problēmas ievada datus, kuri tālākai apstrādei tiek sadalīti un nodoti individuālajiem procesoriem (Lower-Speed system – LSS), kas savā nodabā risina tiem uzdoto problēmu daļu. Kopējā sistēma vienmēr veidojas ar lielām nepilnībām tās darbībā, jo HSS darba sadales process LSS procesoriem nav nedz konstants, nedz paredzams, nedz kontrolējams. Līdz ar to, kopējās sistēmas efektīgai darbībai, ir nepieciešams attīstīts sinhronizācijas un balansēšanas mehānisms, kurš nodrošina vienmērīgu noslodzes sadalījumu starp LSS procesoriem. Šo funkciju uzņemas noslodzes balansētājs, kurš jau tālāk veic darbības atkarībā no LSS procesora padotajiem datiem, kā arī no LSS procesoru padotajiem signāliem.

Sistēmas kopaina ir attēlota zemāk redzamajā zīmējumā. Kā redzams, tad noslodzes sadalītājs pilnībā atdala HSS no LSS uzņemoties pilnīgu vidutāja funkciju.



Figūra 4 Globālā arhitektūra

Datu plūsma: Visi ievada dati tiek pa DataIn kanālu tiek nodoti noslodzes balansētājam, kurš tos tālāk sadala „uz galviņām" un pa DOutA un DOutB kanāliem nodod attiecīgi LSS procesoriem LSA un LSB.

Plūsmas kontrole: Jaunu ievada datu ienākšana no HSS procesora, tiek paziņota ar DReady paziņojuma nosūtīšanu. Ja noslodzes balansētājs ir saņēmis Disable ziņojumu no LSS, tad datu saņemšana no HSS tiek atlikta, līdz tiek saņemts ziņojums, kurš norāda, ka LSS ir spējīgs turpināt darbu ar jauniem ievada datiem. Procesa gaitā no Stat kanāla tiek vākta statistika performances uzmanīšanai.

Rezultātā datu plūsmas fāzes var aprakstīt šādi: sākumā kontrolieris akceptē ienākošos datus no HSS procesora. Tie pamīšus tiek dalīti starp LSA un LSB procesoriem. Tiklīdz tiek saņemts Disable paziņojums, pārslēgšanās starp LSS procesoriem beidzas, un, pie pirmā DReady ziņojuma saņemšanas Disable stāvoklī, uz Stat kanālu tiek padota informācija par LSS procesoru. Pirmajā tuvinājumā noslodzes balansētāja kontroliera darbība varētu tikt aprakstīta ar šādu pseidokoda gabalu:

```
module LoadBalancerControl (In: DReady, Disable: Bool;
                            Out: Stat: Bool)
Var: Proc, NextProc, HangProc, NextHangProc: Bool
Proc := ff
HangProc := ff
while not endOfWorld
  when DReady do
    Proc := NextProc
```

```
      HangProc := NextHangProc
      NextProc = if Disable then Proc
                             else next(Proc)
      NextHangProc = if not Disable then HangProc
                                    else Proc
      Stat = (prev(HangProc)) and Disable
    enddo
  done
  end LoadBalancerControl
```

Šajā pseidokoda gabalā mēs aprakstam nestabilu sistēmu, kas vienmēr akceptē HSS datus bezgalīgajā ciklā. Tiklīdz tiek saņemts ziņojums par ievaddatu esamību, izpildās *when* saturs, kur tiek piešķirtas jaunas vērtības stāvokļu mainīgajiem Proc, HangProc, NextProc un NextHangProc, kā arī tiek ziņots par procesoru un Disable statusu.

Turpmākajās nodaļās tiks aplūkoti konkrēti piemēri:

- šādas implementācijas modeļa aprakstiem izmantojot pavājināto otrās kārtas loģiku,
- kā rīkoties ar modeļu orientētu pieeju, lai pārbaudītu specifikācijas, pārbaudot aprakstu (piemēram, konsekvences un ekvivalences) īpašības,
- kā lietot uz modeļiem bāzētu procedūru šai loģikai, lai iegūtu minimālo modeli, izrēķinot tā semantikas laika sarežģītību galīga automāta formā.

Galīgs automāts ir pieļaujams rezultāts, ko padot uz automātisku optimālu implementācijas shēmu, kuras implementē galīgā automāta darbību, veidotāju. Kā sekas tiek iegūta iespēja izmantot šo pašu procedūru arī specifiskām aparatūras analīzēm:

- dizaina pārbaudei, t.i., pierādīšanai, ka implementācija ir ekvivalenta specifikācijā aprakstītajai,
- tipisko aparatūras kļūdu detektēšanai.

Pavājinātās otrās kārtas loģikas pielāgošana

Jau pirms vairāk nekā 30 gadiem A. Church ierosināja lietot pavājināto otrās kārtas loģiku, kā līdzekli, lai aprakstītu bitu vektorus [8.]. Tieši šī loģika viena no tām, kas visprecīzāk apraksta galīgas stāvokļu sistēmas. Šeit jāpiemin nepatīkams fakts. Šī loģika ir arī ar nepieļaujami lielu sarežģītību – sliktākajā

gadījumā eksponenču kaudze ar augstumu, kurš ir proporcionāls formulas garumam. Tomēr tas nav šķērslis, jo aplūkojamās problēmas parasti ir ar daudz vienkāršāku sarežģītību, kas ļauj tās risināt saprātīgā laika periodā.

Acīmredzama pavājinātās otrās kārtas loģikas priekšrocība ir tas, ka tā kalpo kā precīzs valodas apraksts, kas apvieno modeļu orientētu metožu pilnīgu automatizāciju ar izsakāmību ar loģikas līdzekļiem. Būtībā, ir iespējams pilnīgi automatizēti uzrādīt, pārbaudīt un sintezēt saistītas parametru sistēmas klases, izmantojot Mona [14.] (uz ko balstās šeit aprakstītie rezultāti) vai jaunāko Mosel [17.].

Lielo elastīgumu pavājinātajai otrās kārtas loģikai dod predikātu loģikas pamats, kas arī nodrošina iespēju aprakstīt būtiskas datorprogrammu un aparatūras problēmas. Tas ļauj aplūkot svarīgus drošības parametrus programmas stāvokļos, kur konkrētā stāvokļa interpretācija var tikt aprakstīta dažādos veidos, atkarībā no konkrētās programmas un konkrētā skatu modelēšanas punkta. Eksistē vairākas publikācijas, kurās šī daudzpusība. Piemēram, tā ir arī pielietota aparatūras kontrolieros [21.], secīgām shēmām ar parametriskiem datu celiņiem [20.]. Datorprogrammu nozarē tā ir pielietota, piemēram, RPC-atmiņas specifikācijas problēmas gadījumu pētīšanā [18.], un dažādu „pusdienojošo filozofu" problēmu pētīšanā [14.].

Sintakse un semantika

Aplūkojot pavājināto otrās kārtas loģiku kā abstraktu specifikācijas valodu sistēmām, kas ar laiku attīstās, piešķiram šādu interpretāciju:

- pirmās kārtas loģikas termi t apraksta atsevišķus novērošanas punktus. 0 ir pirmais novērojums un $ ir pēdējais. Operators + apzīmē saskaitīšanu pēc ceļa garuma un parametri var būt naturālo skaitļu konstantes. *p* ir pirmās kārtas loģikas mainīgais,

- otrās kārtas terms T apzīmē boolean signāla vērtību aplūkojamā ceļā. Tiek attēlots ar aplūkoto punktu kopu, kuros signāla vērtība ir 1. *all* ir konstants 1 signāls, *inter* ir punktveida AND operators diviem signāliem, *compl* T apzīmē punktveida papildinājumu T un T + *i* ir operators, kurš pārbīda T pa labi par i pozīcijām.

- Formulas F apzīmē sistēmu visos galīgos ceļos. Nedalāmas formulas apzīmē identitāti $t_1=t_2$ un secību $t_1<t_2$ aplūkojamajiem punktiem un signālu $T_1=T_2$ vienādību. Nedalāma konstrukcija ir arī T@t, kas ir true, ja T=1 punktā t.

Katra formula F var tikt pārveidota par automātu A, kurš tālāk var tikt pārveidots ar automātu teorijas līdzekļiem (minimizāciju, determinētības izteikšanu). Ja F atspoguļo sistēmas specifikāciju, tad A var tikt uzskatīts par šīs sistēmas implementāciju, un F $\rightarrow$ A apzīmē sintēzes procesu. Ja F atspoguļo atbilstošo pārbaudes formulējumu, tad automāts A var tikt lietots, lai noteiktu vai implementācija apmierina doto specifikāciju, un, ja ne, tad iegūt minimālu pretpiemēru.

Specifikāciju tipi

Uzvedības specifikācija.

Uzrādām piemērus formulai loadBalanceControl ar diviem brīvajiem mainīgajiem Disable un Stat, kas apzīmē ievada un izvada signālus. Formula apraksta kopu visām galīgām kalkulācijām, kur mēs apzīmējam vienu aplūkošanas punktu katram DReady notikumam:

```
loadBalanceControl(Disable, Stat) =
  Ex NextHangProc, NextProc, HangProc, Proc:
    (All t: (Stat@t <=> ( Disable@t & ~HangProc@t)) &
            (NextProc@t <=> (( Disable@t & Proc@t) |
                             (~Disable@t & ~Proc@t))) &
            (NextHangProc@t <=> ((~Disable@t & HangProc@t) |
                                 ( Disable@t & Proc@t))) &
    (All t: t<$ => ((Proc@t+1 <=> NextProc@t) &
                    (HangProc@t+1 <=> NextHangProc@t))) &
~Proc@0 & ~HangProc@0)
```

Šī formula ir ļoti līdzīga pseidokodā dotajai specifikācijai. Signālu mainīgie var tikt lietoti kā laika funkcijas, tajā pašā laikā, viņi reprezentē galīgus notikumus.

Signālveida specifikācija.

Ir iespējams definēt otrās kārtas operatorus tieši signāliem, kuri slēpj skaidru saiti uz aplūkojamo stāvokli, saglabājot korektu semantiku. Piemēram, mēs varam aprakstīt stāvokļa signālu Proc ar vienādību

```
Proc = (compl xor(Disable, Proc)) + 1
```

Šajā gadījumā varam tikt vaļā no visiem laika kvantoriem nosliecoties par labu algebriskai signālu salīdzināšanai. Tā pati specifikācija izmantojot šo modelēšanas tipu:

```
loadBalanceControl1(Disable, Stat) =
  Ex NextHangProc, NextProc, HangProc, Proc:
    ((Stat = (Disable inter (compl HangProc))) &
    (NextProc = ((Disable inter Proc) union ((compl Disable) inter
                                             (compl Proc)))) &
    (NextHangProc = (((compl Disable) inter HangProc) union
                     (Disable inter Proc))) &
    (HangProc = NextHangProc + 1) & (Proc = NextProc + 1) &
~HangProc@0 & ~Proc@0)
```

Šī specifikācija ir ne tikai abstraktāka, bet tai arī ir lielākas priekšrocības – to ir vieglāk pārveidot pavājinātā otrās kārtas loģikas pamata sintaksē.

Pārbaudes

Ar piedāvāto metodi mēs varam pārbaudīt korektības relācijas starp specifikāciju un implementāciju vai arī starp viņu alternatīvajiem formulējumiem. Turklāt, ir iespējams formulēt un pārbaudīt nosacījumus, kurus specifikācija vai implementācija vēlas apmierināt.

Nosacījumu pārbaude

Lai pārbaudītu specifikāciju, mēs vēlamies pārbaudīt vai tā ir konsekventa, ka tā definē funkcionālu saistību starp ievadu un izvadu, vai arī, ka tā uzrāda svarīgas īpašības. Visas šīs darbības ir veicamas ar datorprogrammu Mona. Tika parādīts, ka:

- specifikācija ir konsekveneta, jo formula

```
Ex Disable, Stat: loadBalanceControl(Disable, Stat)
```

ir tautoloģija,

- loadBalanceControl definē ievada un izvada relāciju, pierādot

```
All Disable: Ex Stat: loadBalanceControl1(Disable,
                                          Stat)
```

- tāda pati ievada un izvada relācija var tikt definēta:

```
All Disable, Stat1, Stat2:
  (loadBalanceControl1(Disable, Stat1) &
   loadBalanceControl1(Disable, Stat2) => Stat1 = Stat2)
```

Specifikācijas pārbaude

Mērķis ir pārbaudīt pašu specifikāciju to salīdzinot ar kādu uzrādītu alternatīvu. Piemēram , ir iespējams pierādīt, ka divas alternatīva iepriekšējās nodaļās aprakstītās specifikācijas, ir semantiski identiskas.

```
All Disable, Stat: loadBalanceControl(Disable, Stat) <=>
  loadBalanceControl1(Disable, Stat)
```

Modeļu sintēze

Pavājināto otrās kārtas loģiku var izmantot, lai formulētu un atrisinātu sintēzes problēmu. Katra formula F var tikt pārveidota par automātu A, kurš attēlo formulas semantiku. Precīzāk, ja F satur brīvos otrās kārtas loģikas mainīgos, tad A ir automāts, kurš implementē ievada un izvada īpašības, kuras apraksta F brīvie mainīgie.

Pavājinātās otrās kārtas loģikas formulas pārveidošanu par tai atbilstošu automātu var uztvert kā kompilāciju no augsta līmeņa aprakstošas valodas uz zema līmeņa kodu. Tā kā rezultātā vienmēr var iegūt galīgu automātu, tad nav nozīmes tam, cik sarežģīta ir formula vai cik slēpti ir aprakstīta uzvedība. Tiklīdz ir iegūts automāts, tā tas var tikt simulēts, vai nodots aparatūras ražošanas automātiem. Tomēr, šajā brīdī ir jāpiemin sarežģītības sliktākais gadījums. Bezrūpīga sintēzes rīku lietošana var būt lietderīga tikai vienkāršu formulu klasēm.

Secīgu shēmu modelēšana izmantojot pavājināto otrās kārtas loģiku

Aplūkojot pavājināto otrās kārtas loģiku kā shēmu specifikāciju valodu, tiek pieņemta šāda interpretācija:

- pirmās kārtas loģikas terms t apzīmē konkrētu laika stāvokli vai takti. Pie kam, 0 ir sākuma un $ ir beigu stāvoklis aplūkotajā laika intervālā,
- otrās kārtas loģikas terms T, apzīmējot boolean signālus aplūkotājā laika intervālā, apzīmē viļņveida formas cikla laika kopu termos, kuros signāls ir 1. T + i ir i-ciklu aiztures operators signālam,
- formulas F apraksta shēmas darbību.

Šī nav vienīgā loģikas interpretācija. Dažādās aplikācijās otrās kārtas vispārināšana var tikt izteikta telpā, nevis laikā.

Specifikāciju interpretācija

Uzvedības specifikācija. Tā pati formula loadBalanceControl(Disable, Stat) apraksta visu galīgu darbību kopu.

Signālveida specifikācija. Aplūkojot aparatūras kontekstā, šo specifikāciju ir vieglāk lasīt, jo laika apraksti ir daudz tuvāki tipiskiem aparatūras aprakstu valodām.

Dotais galīgais algoritms ar ievadu Disable un izvadu Stat, un sākuma stāvokļa mainīgajiem HangProc, Proc atbilst šādai boolean ekvivalences sistēmai, pārejas funkcijai δ:

$$\delta = (\underline{\text{Disable}} \cdot \text{HangProc} + \text{Disable} \cdot \text{Proc},$$

$$\underline{\text{Disable} \oplus \text{Proc}}) \text{ [lambada]} = \text{Disable} \cdot \underline{\text{HangProc}}$$

Vārtu līmeņa implementācija

Vārtu līmeņa shēmas parasti tiek aprakstītas kā tīklu listes. Vārtu līmeņa primitīvās definīcijas:

```
and(In1, In2, Out) = All t: Out@t <=> (In1@t & In2@t);
or(In1, In2, Out) = All t: Out@t <=> (In1@t | In2@t);
not(In1, Out) = All t: Out@t <=> ~In1@t;
dff(D, Q, Qn) = (All t: t < $ => (Q@t+1 <=> D@t)) &
(All t: Q@t <=> ~Qn@t) &
~Q@0;
```

Tīkla liste daļēja implementācija tiek aprakstīta kā konjunkcija:

```
circuit(Disable, Stat) =
  Ex A, A0, G1, G2, NextHangProc, HangProc, HangProcN, G3, G4,
    NextProc, Proc, ProcN:
    A0 = A & and(Disable, HangProcN, Stat) &
             and(Disable, Proc, G1) &
             and(ProcN, A, G2) &
             or(G1, G2, NextProc) &
             and(A0, HangProc, G3) &
             and(Disable, Proc, G4) &
             or(G3, G4, NextHangProc) &
             dff(NextHangProc, HangProc, HangProcN) &
             dff(NextProc, Proc, ProcN) &
             not(Disable, A);
```

Kļūdu detektēšana

Kā iepriekš tika pieminēts, tad pavājinātā otrās kārtas loģikas formulas semantika atbilst automātam, kurš apraksta visas formulas interpretācijas. Tas ļauj aplūkot tādas problēmas, kā kļūdu atrašana un testa piemēru vispārināšanu, kuri uzrāda kļūdas. Ir zināms, ka kļūda nav detektējama, ja atbilstošā formula ir tautoloģija: šajā gadījumā kļūda neietekmē izvada vērtību. Pretējā gadījumā automāts akceptē ievada vārdu, kurš nepieder valodai, un valodas papildinājums precīzi definē vārdus, kas var kalpot kā testa piemēri. Šī kļūdu meklēšanas pieeja ir pateicīga un ļauj detektēt daudzas kļūdu klases: bez tradicionālajām iestrēgšanas kļūdām, ir iespējams detektēt kļūdas inicializācijas procesa, tāpat arī augsta līmeņa funkcionālas kļūdas – komponenšu trūkums, nepareizi slēgumi.

Aplūkojam gadījumu, kad vēlamies pārbaudīt implementācijas darbību uz inicializācijas kļūdu līnijā A. Ieviešam inicializācijas kļūdu, nomainot A0 = A shēmas deklarācijā ar A0 = all. Nosaucam šo shēmu par circuit_faulty. Tādā gadījumā shēma ar kļūdaino daļu nevar vienmēr izrēķināt identiskus rezultātus. Datorprogramma Mona izdod šādu rezultātu:

```
circuit(Disable, Stat) & circuit_faulty(Disable,
            Stat_f) => Stat = Stat_f
```

Formula ir atspēkota uzrādot modeli, kurš kļūdās.

Noslēgums

Pašreizējie sasniegumi klasisko automātu un matemātiskās loģikas saiknes pētījumos ir devuši labus rezultātus ne tikai teorētiskos pētījumos, bet arī praktiskos dzīves pielietojumos. Ir nodibināta skaidra saikne starp pirmās kārtas loģiku, otrās kārtas loģiku un galīgiem klasiskajiem automātiem. Ir veikti pētījumi, kas noveduši pie tādiem algoritmu sarežģītības teorijas rezultātiem [16.], kā – pirmās kārtas loģika + transitīvais slēgums ir ekvivalents NSPACE[log n]. Kā arī rezultāts pirmās kārtas loģika + pos. transitīvais slēgums ir noslēgts pret papildinājumiem.

Raugoties uz galīgiem kvantu automātiem no loģikas formulu skatu punkta, ir skaidrs, ka vēl ir daudz kur augt un ko pētīt. Rodas tikai bažas par to, vai vispār kvantu mehānikas daba pieļaus tiešu formālu valodu klašu un kādas loģikas bijekcijas nodibināšanu.

Rezultāti praktiskajā sfērā ir ļāvuši izstrādāt pārbaudes freimvorkus ar plašu abstrakcijas līmeni, ietverot sevī no programmu arhitektūras un protokolu līmeņa līdz aparatūras reģistru izmaiņām. Pie tam, abi aprakstu tipi – uzvedības un strukturālais – ļauj veikt modeļu bāzētas analīzes, pārbaudes un kļūdu atklāšanas. Iespēja pilnībā apslēpt programmatūras darbību un ievada valodu, tiklīdz tā tiek integrēta projektā, ir lielākā priekšrocība izstrādes procesā.

Izmantotā literatūra

[1.] D. Aharonov, A. Kitaev, N. Nisan. Quantum Circuits with Mixed States. STOC, 20-30 lpp, 1998.

[2.] A. Ambainis, M. Beaudry, M. Golovkins, A. Kikusts, M. Mercer, D. Thrien. Algebraic Results on Quantum Automata. In: STACS 2004:, 93-104 lpp, 2004.

[3.] A. Ambainis, R. Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In: Proc. FOCS, 332-341 lpp, 1998.

[4.] A. Ambainis, J. Watrous. Two-way finite automata with quantum and classical states. Theoretical Computer Science 287, No.1, 299-311 lpp, 2002.

[5.] A. Bertoni, C. Mereghetti, B. Palano. Quantum Computing: 1-Way Quantum Automata. Developments in Language Theory 2003: 1-20 lpp, 2003.

[6.] J. R. Büchi. On a decision method in restricted second order arithmetic. Proc. Int. Congress Logic, Methodology and Philosophy of Sciences 1960, Stanford University Press, 1-11 lpp, 1962.

[7.] J. R. Büchi. Weak second-order arithmetic and finite automata. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 6:66-92 lpp, 1960.

[8.] A. Church: Logic, arithmetic and automata, Proc. Int. Congr. Math., Almqvist and Wiksells, Uppsala 1963, 23-35 lpp.

[9.] I. Dzelme. First Order Logic and Quantum Finite State Automata. 2006.

[10.] I. Dzelme. Quantum Finite Automata and Logics. SOFSEM 2006, 246-253, 2006.

[11.] C. C. Elgot. Decision problems of finite automata design and related arithmetic's. Transactions of AMS, 98:21-52, 1961.

[12.] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In: Complexity of Computation, SIAM-AMS Proceedings 7, 43-73, 1974.

[13.] R. Freivalds. Languages Recognizable by Quantum Finite Automata, 2006.

[14.] J. Henriksen, J. Jensen, M. Jorgensen N. Klarlund, R. Paige, T. Rauhe, A. Sandholm. Mona: Monadic second-order logic in practice, Proc. of TACAS'95, Aarhus (DK), LNCS 1019, Springer Verlag, 89-110 lpp, 1995.

[15.] N. Immerman. Languages that capture complexity classes. SIAM J. Comput. 16:4, 760-778, 1987.

[16.] N. Immerman. Nondeterministic Space is Closed Under Complementation, Third Structure in Complexity Theory Conf. 1988. kā arī SIAM J. Comput.

[17.] P. Kelb, T. Margaria, M. Mendler, C. Gsottberger: MOSEL: A flexible toolset for monadic second-order logic, To appear in Proc. of TACAS'97, Twente (NL), LNCS, Springer Verlag. 1997.
[18.] N. Klarlund, M. Nielsen, K. Sunesen: A case study in verification based on trace abstraction, In M. Broy and S. Merz and K. Spies, eds., Formal Systems Specification, The RPC-Memory Specification Case Study, LNCS 1069, Springer Verlag, 341-373 lpp.
[19.] A. Kondacs, J. Watrous. On the power of quantum finite state automata. In: Proc. FOCS'97, 66-75.
[20.] T. Margaria: Fully Automatic Verification and Error Detection for Parameterized Iterative Sequential Circuits, Proc. TACAS'96, Passau (D), LNCS 1055, Springer Verlag, 1996.
[21.] T. Margaria, M. Mendler. Automatic Treatment of Sequential Circuits in Second-Order Monadic Logic. 4th GI/ITG/GME Worksh. on Methoden des Entwurfs und der Verifikation digitaler Systeme, Kreischa (D), Shaker Verlag, 1996.
[22.] T. Margaria, M. Mendler. Model-based Automatic Synthesis and Analysis in Second-Order Monadic Logic. 2000. Pieejams internetā: http://ls5-www.cs.uni-dortmund.de/imperia/md/content/publikationen/mame97-c.pdf
[23.] R. McNaughton. Testing and generating infinite sequences by finite automaton. Inform. Contr. 9, 521-530, 1966.
[24.] C. Moore, J. Crutchfield. Quantum automata and quantum grammars. Theoretical Computer Science, 237:275-306, 2000.
[25.] A. Nayak. Optimal Lower Bounds for Quantum Automata and Random Access Codes. Proc. 40th FOCS, 1999, pp. 369-377.
[26.] A. Pnueli. The temporal logic of programs. Proc. FOCS'77, 1-14, 1977.
[27.] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. Trans. Amer. Math. Soc. 141:1-35, 1969.
[28.] M. P. Schüzenberg. On finite monoids having only trivial subgroups. Information and Control 8:283-305, 1965.
[29.] Boris A. Tranhtenbrot. Finite automata and the logic of one-place predicates. Siberian Mathematical Journal, 3:103-131, 1962 (in Russia), English translation: American Mathematical Society Translations, Series 2, 59:23-55, 1966.
[30.] M. Vardi. Complexity of relational query languages. Proc. STOC'82, 137-146, 1982.
[31.] I. Walukiewicz. Automata and Logic. [tiešsaite] Warsaw University Pieejams internetā: http://citeseer.ist.psu.edu/520826.html.
[32.] T. Wolfgang. Languages, Automata and Logic. [tiešsaite] Technical Report 9607, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, Germany, May 1996. Pieejams internetā: http://www.math.bgu.ac.il/~abraham/courses/Automata/thomas.ps.

Apliecinājums

Ar šo es apliecinu, ka šodien iesniegto bakalaura darbu esmu veicis pašrocīgi un esmu izmantojis tikai tajā norādītos palīglīdzekļus.

Rīgā, Paraksts:

Reģistrācijas lapa

Bakalaura darbs izstrādāts

LU Datorikas nodaļā

Autors:

Fizikas un matemātikas

fakultātes students Mārtiņš Kalvāns .....

St. apl. Nr. DatZ 020031, ...

Darba vadītājs

Dr. habil. math. Rūsiņš Mārtiņš Freivalds .....

Recenzents

Dr. sc. comp Ēvalds Ikaunieks.....

Darbs iesniegts Datorikas nodaļā 2006. g. . . . . .

Pieņēma sekretāre .....

Aizstāvēts datorzinātņu bakalaura pārbaudījumu komisijas sēdē

2006.g. . . . . .ar atzīmi. . . . . .

Kursa pārbaudījumu komisija . . . . . .