Mathematical Problems of Computer Science 28, 2007, 34–44.

OTAS - տեքստերի համեմատման խնդիրների լուծման ծրագրային գործիք

Արմեն Վ. Քոչարյան

ՀՀ ԳԱԱ Ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտ
e-mail: armkoch@ipia.sci.am

Ամփոփում

Ստեղծված է OTAS (Online Tessallation Automata Syntesator) ծրագրային գործիքը, որը տեքստերի համեմատման խնդիրների ՕԽԱ ճանաչելի նկարագրի համար սինթեզում է տեքստերը համեմատող ծրագրային մոդուլ:

1 Ներածություն

Հաշվողական տեխնիկայի վերջին զարգացումներն առաջ բերեցին հետաքրքրություն դեպի զուգահեռ հաշվարկների մեթոդները և բազմաթիվ բնագավառներում նրանց կիրառությունների: Բջջային ավտոմատները պատկանում են ապակենտրոնացված հաշվողական մոդելների շարքին, որոնք հանդիսանում են կոմպլեքս հաշվարկների հիմք: Դրանց կառուցվածքը կարելի է դիտարկել որպես զուգահեռ հաշվողական գործիք: Բջջային ավտոմատներն օգտագործվում են բազմաթիվ բնագավառներում տարբեր պրոցեսների մոդելավորման ժամանակ: Հաշվի առնելով տրամաբանության և ՕԽԱ ավտոմատների էկվիվալենտությունը՝ վերջիններս կարող են բերվել մեկտեղանի երկրորդ կարգի տրամաբանական բանաձևերի [3,4]:

Վերջին տարիներում մեծ ջանքեր են գործադրվում բջջային ավտոմատների ավտոմատ սինթեզման մեթոդների զարգացման համար [8]: Այդ մեթոդները կառուցում են բջջային ավտոմատներ, որոնք կատարում են կարգընթաց ֆունկցիաների համակարգով նկարագրվող հաշվարկներ:

Մեր մոտեցումը տարբերվում է դրանից: Մենք դիտարկում ենք խնդիրների դաս, որի համար ապացուցում ենք բջջային ավտոմատի գոյությունը, և որը կարող է նկարագրվել հատուկ տիպի տրամաբանական բանաձևերով: Այդ դասի համար առաջարկվում է բջջային ավտոմատների սինթեզման ալգորիթմ:

Այս աշխատանքի նպատակն է լուսաբանել OTAS (Online Tessallation Automata Syntesator) ծրագրային գործիքի հնարավորությունները և օգտվելու կանոնները Տեքստերի Համեմատման Խնդիրների(ՏՀԽ) դասի լուծման համար:

ՕԽԱ-ն բջջային ավտոմատ է, որը ներմուծվել է K.Inoue-ի և A.Nakamura-ի կողմից և նախատեսված է մատրիցային լեզուների ճանաչման համար[1,2]: Այս ավտոմատը միանման վերջավոր ավտոմատների զանգված է, որտեղ անցումների ալիքը մեկ անգամ անկյունագծով անցնում է զանգվածի(մատրիցայի) վրայով: Հաշվի առնելով տրամաբանության և ՕԽԱ-ների էկվիվալենտությունը՝ վերջիններս կարող են բերվել մեկտեղանի երկրորդ կարգի բանաձևերի [3]:

Աշխատանքը նվիրված է ՏՀԽ լուծող ՕԽԱ ավտոմատների սինթեզմանը: Տեքստերի համեմատման խնդիրները հանդիսանում են հայտնի Բառերի Համեմատման Խնդիրների (ԲՀԽ) ընդհանրացում: OTAS ծրագրային գործիքով սինթեզված ՕԽԱ-ն հանդիսանում է միաչափ բջջային ավտոմատ, որը յուրաքանչյուր քայլում որպես մուտք կարդում է U և V տեքստերի միայն երկու հաջորդ տառերը: Վերջում ներկայացված են փորձարկումների արդյունքները ArmCluster անճնական պզտագործման համակարգիչների կլաստերի վրա:

Այժմ բացատրենք տեքստերի վերաբերյալ ներմուծված հասկացողությունները:

Սովորաբար տեքստերը հասկացվում են որպես բառերի՝ վերջավոր A այբուբենում շղթաների հաջորդականություն: Գոյություն ունեն բառերի համեմատման բազմաթիվ խնդիրներ(ալգորիթմներ)՝ հիմնված տառերի համեմատման վրա:

Նույն ձևով ինչպես բառերի դեպքում(այսինքն տառերի շղթաների), մենք դիտարկում ենք տեքստերը որպես բառերի հաջորդականություն: Մենք ավելացրել ենք մեկ նոր սիմվոլ, որը կոչվում է »բաժանիչ« և չի պատկանում A այբուբենին՝ »բաժանիչ« $\notin A$, որը ծառայում է որպես անջատիչ սիմվոլ և ներառված է $A\cup\{$»բաժանիչ«$\}$ այբուբենում շղթաների կազմում: $A\cup\{$»բաժանիչ«$\}$այբուբենում շղթաները վերածվում են A այբուբենում բառերի հաջորդականությունների, որոնք բաժանված են »բաժանիչ« լրացուցիչ սիմվոլով:

Դիտարկված է տեքստերի համեմատման խնդիրների (ՏՀԽ) ՕԽԱ ճանաչելի ենթադասը: Այս դասի յուրաքանչյուր խնդիր տրվում է տեքստերի վրա որոշված որոշակի բիլար $P_{problem}$ պրեդիկատով, և կայանում է $P_{problem}(U,V)$ պրեդիկատի գնահատման մեջ կամայական տրված $U=u_1,\ldots,u_m$ և $V=v_1,\ldots,v_n$ տեքստերի համար, որտեղ u_i, v_j բառեր են A այբուբենում:

Դրված են որոշակի սահմանափակումներ $P_{problem}$ պրեդիկատի վրա՝ դրանով սահմանափակելով դիտարկվող խնդիրների դասը: Ենթադրվում է, որ գոյություն ունեն բառերի վրա որոշված $P_1,\ldots,P_r$ բիլար պրեդիկատներ (լոկալ պրեդիկատներ) և $\{0,1\}^r$ այբուբենում մատրիցաների վրա որոշված $\mathcal{P}$ ունար պրեդիկատ (գլոբալ պրեդիկատ): $P_{problem}(U,V)$ պրեդիկատի գնահատումը կարող է կատարվել երկու քայլով, որոնք սահմանվում են ներքևում: Այս երկու քայլերի արդյունքում ստացվում է $P_{problem}(U,V)$ պրեդիկատի արդյունքը:

Առաջին քայլը կայանում է U և V տեքստերի համեմատման մեջ: Յուրաքանչյուր u_i բառ համեմատվում է յուրաքանչյուր v_j բառի հետ, այսինքն u_i, v_j յուրաքանչյուր զույգի համար $P_1,\ldots,P_r$ բիլար պրեդիկատները պետք է հաշվարկվեն: Այս լոկալ համեմատումների արդյունքում ստացվում է U և V տեքստերի $IM_{U,V}$-ինֆորմացիոն մատրիցան, որտեղ

$$IM_{U,V}(i,j)=(P_1(u_i,v_j),\ldots,P_r(u_i,v_j))\quad i=1,\ldots,m,\ \ j=1,\ldots,n.$$

Երկրորդ քայլը կայանում է $\mathcal{P}(IM_{U,V})$ գլոբալ պրեդիկատի արժեքը հաշվելու մեջ, որը հանդիսանում է $P_{problem}(U,V)=\mathcal{P}(IM_{U,V})$ պրեդիկատի արժեքը:

Այս խնդիրը կնշանակենք ՏՀԽ$(P_1,\ldots,P_r,\mathcal{P})$:

Հեշտ է նկատել, որ շատ խնդիրներ կարող են ձևակերպվել որպես ՏՀԽ: Շատ ԲՀԽ-ներ, որոնք հայտնի են որպես նմուշի փնտրման, պարզ և զուգահեռ նմուշի փնտրման, պատուհանում նմուշի փնտրման և այլ խնդիրներ, կարող են ընդհանրացվել որպես ՏՀԽ-ներ:

Այժմ բերենք երկու օրինակ:

Օրինակ 1. Նմուշի փնտրման խնդիրը որպես բառերի փնտրման խնդիր կայանում է նրանում, թե արդյո՞ք u բառը հանդիսանում է v բառի ֆակտոր: Նմուշի փնտրման խնդիրը որպես տեքստերի փնտրման խնդիր կայանում է նրանում, թե արդյո՞ք U տեքստը

հանդիսանում է V տեքստի ֆակտոր, որտեղ U-ի և V-ի մեջ մտնող բառերը համեմատվում են ամբողջությամբ, որպես միավորներ:

Այստեղ որպես լոկալ պրեդիկատ մենք կարող ենք օգտագործել $P_=$ նույնության պրեդիկատը: $\mathcal{P}_1(M)$ գլոբալ պրեդիկատն ընդունում է »1« արժեք այն և միայն այն դեպքում, երբ M մատրիցայում գոյություն ունեն այնպիսի հաջորդական սյուներ, որոնք կազմում են միավոր մատրիցա:

Երբ $V =$ „*non − empty word*" և $U =$ „*empty*" կամ $U =$ „*empty word*", որտեղ պրոբլեմը հանդիսանում է բաժանիչ, ապա տեքստերի համար ճմուշի փնտրման խնդիրն ունի բացասական լուծում:

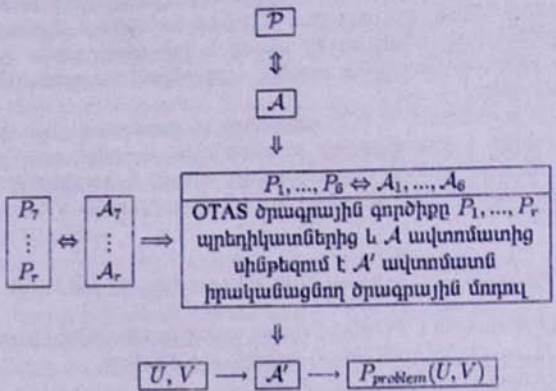
Երբ մենք դիտարկենք V-ն և U-ն որպես բառեր այնպիսի այբուբենում, որը պարունակում է բացակը որպես տառ, ապա բառերի համար ճմուշի փնտրման խնդիրն ունի դրական լուծում: ◇

Օրինակ 2. Նմուշի զուգահեռ փնտրում:

Դիցուք տրված են $U = u_1, ..., u_m$ և $V = v_1, ..., v_n$ երկու տեքստերը: Խնդիրը կայանում է նրանում, թե գոյություն ունի՞ արդյոք V-ում այնպիսի $v_{j_1}, ..., v_{j_m}$ բառերի ենթահաջորդականություն, որ $u_i = v_{j_i}$ բոլոր $i = 1, ..., m$ համար: Այստեղ նույնպես մենք կարող ենք օգտագործել որպես լոկալ պրեդիկատ $P_=$ նույնության պրեդիկատը: $\mathcal{P}_2(M)$ գլոբալ պրեդիկատն ընդունում է 1 արժեք այն և միայն այն դեպքում, երբ M մատրիցայի յուրաքանչյուր տող պարունակում է ճիշտ մեկ հատ 1: ◇

Ստեղծված է OTAS (Online Tessallation Automata Syntesator) ծրագրային գործիք, որը տեքստերի համեմատման խնդիրների ՕԽԱ [7] ճանաչելի ենթադասի համար սինթեզում է տեքստերը համեմատող ծրագրային մոդուլ:

Նկար 1-ում ցույց է տրված OTAS ծրագրային գործիքի կառուցվածքային պատկերը:



Նկար 1. OTAS ծրագրային գործիքի աշխատանքի կառուցվածքային պատկեր:

OTAS ծրագրային գործիքը բավարարում է նախապես դրված հետևյալ պահանջներին.

1. այն շատ պարզ է օգտագործման համար,
2. OTAS-ն ունի ներդրված բառերի համեմատման լոկալ պրեդիկատներ,
3. ապահովում է բառերի համեմատման լոկալ պրեդիկատներ ավելացնելու հնարավորություն,

4. որպես գլոբալ պրեդիկատ այն ընդունում է ինֆորմացիոն մատրիցայի վրա աշխատող խճանկարային ավտոմատի անցումների ֆունկցիան,

5. որպես աշխատանքի արդյունք վերադարձնում է ավտոմատ կերպով սինթեզված ծրագրային մոդուլ, որը մոդելավորում է մուտքագրված խճանկարային ավտոմատի աշխատանքը համեմատվող տեքստերի վրա,

6. ավտոմատ կերպով սինթեզված ծրագրային մոդուլն աշխատում է գծային ժամանակում,

7. իրականացված են OTAS ծրագրային գործիքի հաջորդական և զուգահեռ տարբերակներ: Հաջորդական տարբերակում գեներացվում է այնպիսի ծրագրային մոդուլ, որն աշխատում է մեկ պրոցեսորի վրա որպես հաջորդական ալգորիթմ: Զուգահեռ տարբերակում սինթեզվում է զուգահեռ մոդուլ, որն աշխատում է բազմապրոցեսորային կլաստերի վրա որպես զուգահեռ ալգորիթմ:

2 Սահմանումներ

Ենթադրենք A -ն վերջավոր ոչ դատարկ այբուբեն է: A այբուբենում $w = w_1w_2...w_k$, $w_i \in A$, $i = 1...k$ տառերի հաջորդականությունը կանվանենք *բառ*: $w \in A^+$, որտեղ A^+ -ով կնշանակենք A այբուբենում բոլոր բառերի բազմությունը՝ բացի դատարկ բառից:

$A^+ \cup \{*\}$ այբուբենում $t_1, ..., t_n$ բառերից կազմված $T = *t_1 * ... * t_n*$ տիպի հաջորդականությունը կանվանենք *տեքստ*, որտեղ $t_i \in A^+, i = 1, ..., n$ և $* \notin A$:

T տեքստի *երկարություն* կանվանենք $|T| = n + 1 + \sum|t_i|$, որտեղ $|t_i|$ -ն t_i բառի երկարությունն է:

Ենթադրենք U-ն և V-ն տեքստեր են A այբուբենում:

$U = *u_1 * \cdots * u_m*$, որտեղ $u_1 = u_{11}u_{12}...u_{1k_1}, \cdots, u_m = u_{m1}u_{m2}...u_{mk_m}$ $u_{ij} \in A$:

$V = *v_1 * \cdots * v_n*$, որտեղ $v_1 = v_{11}v_{12}...v_{1l_1}, \cdots, v_n = v_{n1}v_{n2}...v_{nl_n}$ $v_{ij} \in A$:

Ենթադրենք ֆիքսված են $P_1, ..., P_r$ քիչար պրեդիկատները՝ որոշված A այբուբենում բառերի զույգերի վրա: Կամայական (U, V) տեքստերի զույգին կհամապատասխանեցնենք (m, n) մատրիցա և կնշանակենք $IM_{U,V}$, որի էլեմենտները r քիտանոց վեկտորներ են՝ $IM_{U,V}(i, j) = (P_1(u_i, v_j), ..., P_r(u_i, v_j))$, $1 \le i \le m, 1 \le j \le n$: $IM_{U,V}$ մատրիցան կանվանենք (U, V) զույգի *ինֆորմացիոն մատրիցա*:

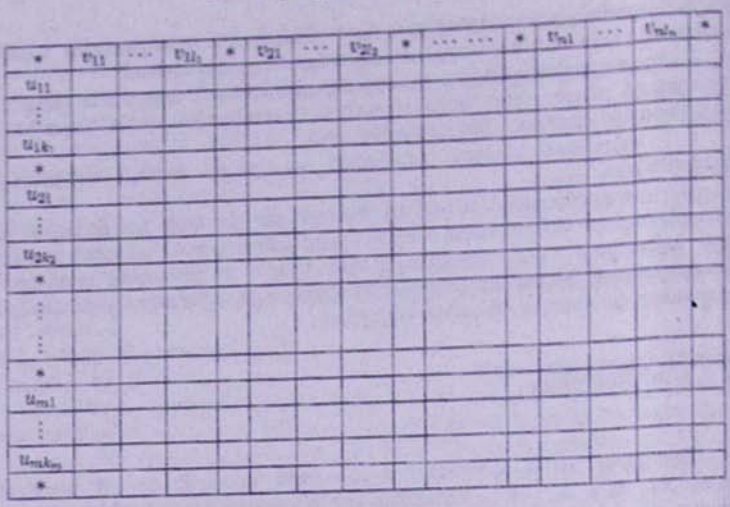
$IM_{U,V} =$

$(P_1(u_1, v_1), ..., P_r(u_1, v_1))$	$\cdots \cdots$	$(P_1(u_1, v_n), ..., P_r(u_1, v_n))$
$(P_1(u_2, v_1), ..., P_r(u_2, v_1))$	$\cdots \cdots$	$(P_1(u_2, v_n), ..., P_r(u_2, v_n))$
$\vdots$		$\vdots$
$\vdots$		$\vdots$
$(P_1(u_m, v_1), ..., P_r(u_m, v_1))$	$\cdots \cdots$	$(P_1(u_m, v_n), ..., P_r(u_m, v_n))$

Նկար 2. (U, V) տեքստերի ինֆորմացիոն մատրիցա

(U, V) տեքստերի զույգի *մատրիցային կոդ* կանվանենք (կնշանակենք $MC_{U,V}$) $A \cup \{*\}$ այբուբենում այնպիսի $(|U|, |V|)$ մատրիցան, որի առաջին սյունում գրված է U տեքստը, առաջին տողում՝ V տեքստը, իսկ մնացած էլեմենտները կամայական սիմվոլներ են վերցված A այբուբենից:

$MC_{U,V} =$



Նկար 3. (U, V) տեքստերի մատրիցային կող

Օրինակ 3. Ենթադրենք $U = *I * match * texts*$ և $V = *Now * I * match * two * texts*$: Դիցուք $P_=$-ը բառերի հավասարության պրեդիկատն է՝

$$P_=(u, v) = \begin{cases} 1, & \text{եթե } u = v \\ 0, & \text{եթե } u \neq v \end{cases}$$

U և V տեքստերի մատրիցային կողը հետևյալն է՝

$MC_{U,V} =$

*	N	o	w	*	I	*	m	a	t	c	h	*	t	w	o	*	t	e	x	t	s	*
I	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
*	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
m	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
a	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
t	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
c	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
h	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
*	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
t	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
e	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
x	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
t	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
s	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
*	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o

Նկար 4. Օրինակ 3-ում նշված տեքստերի մատրիցային կող

Բառերի հավասարության պրեդիկատով U և V տեքստերի ինֆորմացիոն մատրիցան

հետևյալն է՝

$$IM_{U,V} = \begin{array}{|c|c|c|c|c|}\hline 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Նկար 5. Օրինակ 3-ում նշված տեքստերի ինֆորմացիոն մատրիցան $P_=$ պրեդիկատով

3 Ծրագրի աշխատանքի նկարագրությունը

Դիցուք օգտվողի խնդիրը կայանում է հետևյալում՝ անհրաժեշտ է ծրագիր, որը U, V տեքստերի զույգի համար հաշվի $\mathcal{P}(U,V)$ պրեդիկատի արժեքը

$$\mathcal{P}(U,V) = \begin{cases} 1\,, & \text{եթե տեքստերը համեմատելի են} \\ 0\,, & \text{հակառակ դեպքում:} \end{cases}$$

Դիցուք այդ պրեդիկատն արտահայտվում է $(S_1, S_2, P_1, ..., P_r)$ սիգնատուրայում EMSO տրամաբանության մեջ, որտեղ $P_1, ..., P_r$ բառերի համեմատման պրեդիկատները ՕԽԱ ճանաչելի են:

Հայտնի է, որ մատրցային լեզուների վրա EMSO տրամաբանության բանաձևի իրագործելիությունը համարժեք է ՕԽԱ ավտոմատով ճանաչելիությանը: Ուստի համեմատման $\mathcal{P}$ պրեդիկատը կարելի է բերել մատրիցաների վրա համարժեք ՕԽԱ ավտեմատի [3]:

Եթե բավարարված է վերը նշված պահանջը, ապա կարելի է խնդիրը լուծել OTAS ծրագրային գործիքի միջոցով:

Օրինակ 4. Դիտարկենք հետևյալ խնդիրը. հանդիսանում է՞ արդյոք U տեքստը V տեքստի ենթատեքստ: Օգտագործելով OTAS-ը՝ կարող ենք ստանալ ծրագրային մոդուլ, որը կլուծի խնդիրը տեքստերի կամայական U, V զույգի համար:

Դիտարկենք կոնկրետ օրինակ: Դիցուք $U = *I * match * texts*$ և $V = *Now * I * match * two * texts * using * OTAS*$:

Բառերի հավասարության պրեդիկատով U և V տեքստերի ինֆորմացիոն մատրիցան հետևյալն է՝

$$U \quad V \quad \begin{array}{|c|c|c|c|c|c|c|}\hline 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline \end{array}$$

Նկար 6. Օրինակ 4-ում նշված տեքստերի ինֆորմացիոն մատրիցան $P_=$ պրեդիկատով

Որպեսզի U տեքստը լինի V -ի ենթատեքստ, անհրաժեշտ է և բավարար, որպեսզի ինֆորմացիոն մատրիցայի յուրաքանչյուր տողում հնարավոր լինի ընտրել առնվազն մեկ հատ 1 պարունակող վանդակ այնպիսին, որ յուրաքանչյուր տողում ընտրված վանդակը

գտնվի նախորդ տողում ընտրվածից աջ:

$\mathcal{A}$ -ով նշանակենք $\{0,1\}$ այբուբենում մատրիցաների վրա աշխատող այն ՕԽԱ ավտոմատը, որը լուծում է ենթատեքստ գտնելու խնդիրը:

$\mathcal{A}$ ՕԽԱ ավտոմատն ունի հետևյալ տեսքը՝

$\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, որտեղ

$\Sigma = \{0,1\}$ և $a \in \Sigma$;

$Q = \{q_0, q_1, q_2, q_3, q_4\}$;

q_0-ն սկզբնական վիճակն է;

$F = q_2, q_3$:

Ավտոմատի անցումների δ ֆունկցիան ունի հետևյալ տեսքը՝

$\delta(q_0, q_0, 0) = q_1$, $\delta(q_1, q_0, 0) = q_1$, $\delta(q_2, q_0, 0) = q_3$, $\delta(q_3, q_0, 0) = q_3$,
$\delta(q_0, q_0, 1) = q_2$, $\delta(q_1, q_0, 1) = q_2$, $\delta(q_2, q_0, 1) = q_3$, $\delta(q_3, q_0, 1) = q_3$,
$\delta(q_0, q_1, 0) = q_1$, $\delta(q_1, q_1, 0) = q_1$, $\delta(q_1, q_2, 0) = q_1$, $\delta(q_1, q_3, 0) = q_1$,
$\delta(q_2, q_1, 1) = q_1$, $\delta(q_1, q_1, 1) = q_1$, $\delta(q_1, q_2, 1) = q_1$, $\delta(q_1, q_3, 1) = q_2$:

OTAS-ը ստանալով $\mathcal{A}$ ավտոմատի անցումների ֆունկցիան իրականացնող ծրագրային կողը որպես մուտք՝ սինթեզում է ենթահաջորդականություն գտնելու խնդիրի լուծման ծրագրային մոդուլ: Ստացված ծրագրի միջոցով հնարավոր է լուծել խնդիրը U, V տեքստերի կամայական զույգի համար:

OTAS ծրագրային գործիքով խնդիրը լուծելու համար անհրաժեշտ է.

1) ներդրված համեմատման պրեդիկատներից ընտրել P_{eq} .

2 C++ ծրագրավորման լեզվով նկարագրել ինֆորմացիոն մատրիցայի վրա աշխատող խճանկարային $\mathcal{A}$ ավտոմատի անցումների ֆունկցիան:

3) մուտքագրել համեմատվող տեքստերը պարունակող ֆայլերի հասցեները,

Որպես աշխատանքի արդյունք OTAS ծրագրային գործիքը վերադարձնում է աշխատող ծրագրային մոդուլ, որը մոդելավորում է $\mathcal{A}$ ավտոմատի աշխատանքը համեմատվող տեքստերի մատրիցային կողի վրա, այսինքն աշխատում է անմիջապես մուտքային տեքստերի վրա: Օգտագործելով ստացված ծրագրային մոդուլը, կարելի է լուծել ենթատեքստ գտնելու խնդիրը տեքստերի կամայական զույգի համար: ◊

Հատուկ նշենք, որ OTAS ծրագրային գործիքում որպես ներդրված լոկալ պրեդիկատներ օգտագործվում են $P_{eq}, P_{eqi}, P_{fact}, P_{seq}, P_{peq}, P_{perm}$ բառերի համեմատման պրեդիկատները, որոնք թույլ են տալիս լուծել բազմաթիվ Բառերի Համեմատման դասական Խնդիրները(ԲՀԽ) առանց որևէ նոր պրեդիկատ ավելացնելու: Բազմաթիվ ԲՀԽ-ներ կարելի է ներկայացնել որպես ՏՀԽ$(P_1, \mathcal{P})$, որտեղ P_1-ը ներդրված պրեդիկատներից որևէ մեկն է, իսկ $\mathcal{P}$-ն բոլոր մատրիցաները ճանաչող համապիտանի պրեդիկատը: Խնդիրը լուծելու համար ինտերակտիվ ռեժիմում ընտրվում է համապատասխան պրեդիկատը, իսկ $\mathcal{P}$-ն ներմուծելու անհրաժեշտություն չկա: Որպես ծրագրի արդյունք պետք է պահանջել ընտրված P_1 պրեդիկատի արժեքը: Համեմատվող u և v բառերը տրվում են մուտքային ֆայլերի մեջ, որոնք վերջանում են $*$ նիշով:

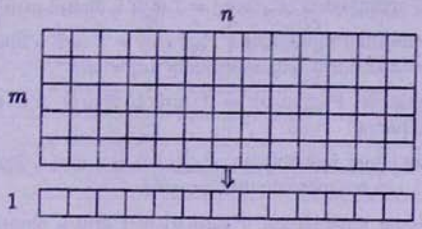
$\mathcal{A}_1$-ով նշանակենք տեքստերի համեմատման խնդիրը լուծող սինթեզված ՕԽԱ ավտոմատը:

Ծրագիրն իրականացված է երկու տարբերակով:

1.Հաջորդական աշխատող մոդուլի սինթեզման ժամանակ տվյալների երկչափ զանգվածի վրա աշխատող խճանկարային $\mathcal{A}_1$ ավտոմատի աշխատանքը մոդելավորված է տվյալների միաչափ զանգվածի վրա աշխատող $\mathcal{A}_2$ ավտոմատով: Ընդ որում, եթե $\mathcal{A}_1$ ավտոմատն աշխատում է (m, n) չափերով տվյալների երկչափ զանգվածի վրա, ապա $\mathcal{A}_2$ ավտոմատն աշխատում է $(1, n)$ չափերով տվյալների միաչափ զանգվածի վրա և նրա

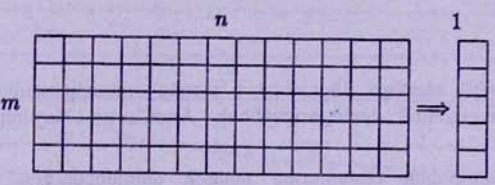
յուրաքանչյուր էլեմենտում պահված է նույն ծավալի ինֆորմացիա, ինչ որ A_1 ավտոմատի աշխատանքի դեպքում: Որպես բջջային ավտոմատ A_2 ավտոմատի աշխատանքի քայլերի քանակը $m+n+1$ է: Ինչպես նաև A_2 ավտոմատի աշխատանքի դեպքում միաչափ զանգվածի յուրաքանչյուր բջիջում աշխատում է այն նույն էլեմենտար ավտոմատը, ինչ որ A_1 ավտոմատի աշխատանքի դեպքում: Մեկ բջիջում աշխատող էլեմենտար ավտոմատներն աշխատում են ճիշտ այնքան անգամ, ինչքան նրանք աշխատում են A_1 ավտոմատում:

Նկար 7-ում ցույց է տրված A_1 ավտոմատից A_2 ավտոմատին անցնելու դեպքում ստացվող հիշողության խնայողությունը, երբ (m,n) չափերով երկչափ մատրիցայի դեպքում օգտագործվում է $(1,n)$ չափերով միաչափ զանգված:



Նկար 7: OTA ավտոմատի հաջորդական մոդելավորման սխեման

2.Զուգահեռ աշխատող մոդուլի սինթեզման դեպքում A_1 ավտոմատի աշխատանքը մոդելավորվում է միաչափ սիստոլիկ ալգորիթմի աշխատանքով, որի երկարությունը m է, իսկ քայլերի քանակը՝ $m+n+1$: Սիստոլիկ ալգորիթմի յուրաքանչյուր բջջում պահվում է նույն քանակի ինֆորմացիա և աշխատում է այն նույն էլեմենտար ավտոմատը, ինչ որ A_1 ավտոմատի աշխատանքի դեպքում: Սիստոլիկ ալգորիթմի աշխատանքը զուգահեռացվում և մոդելավորվում է կլաստերային միջավայրում:



Նկար 8: OTA ավտոմատի զուգահեռ մոդելավորման սխեման

Նկար 8-ում սխեմատիկ կերպով ցույց է տրված A_1 ավտոմատից սիստոլիկ ալգորիթմի անցնելու դեպքում մատրիցայի չափերի փոփոխությունը:

Որպես ավտոմատի մուտքային այբուբեն և համեմատվող տեքստերի այբուբեն ընտրված է C++ ծրագրավորման լեզվի «char» տիպի սիմվոլների բազմությունը՝ բացի ֆայլի վերջի սիմվոլից և '*' սիմվոլից, որը հանդիսանում է անջատիչ:

Մուտքային տվյալների ֆայլերը պետք է պարունակեն համեմատվող բառերի հաջորդականություններ՝ բաժանված '*' սիմվոլով, և ֆայլի վերջին սիմվոլը նույնպես պետք է լինի բաժանիչ սիմվոլ: Չի թույլատրվում հաջորդականության մեջ նշել դատարկ

բառ:

OTAS ծրագրային գործիքն օգտվողին հնարավորություն է տալիս նոր ծրագրային մոդուլի սինթեզման ժամանակ որպես համեմատման պրեդիկատ ինտերակտիվ միջավայրում ընտրել ծրագրում ներդրված $P_{eq}, P_{eql}, P_{fact}, P_{seq}, P_{pep}, P_{perm}$ վեց բառերի համեմատման լոկալ պրեդիկատներից մեկը կամ մի քանիսը միաժամանակ: Ինչպես նաև օգտվողը հնարավորություն ունի անհրաժեշտության դեպքում ավելացնել կամայական ՕՒԼ ճանաչելի բառերի համեմատման պրեդիկատ, որի համար անհրաժեշտ է մուտքագրել պրեդիկատի անունը և պրեդիկատին համարժեք խճանկարային ավտոմատի անցումային ֆունկցիան՝ նկարագրված C++ ծրագրավորման լեզվով:

Ծրագրում ներդրված լոկալ պրեդիկատներն են՝

1. Հավասարության պրեդիկատ՝ $P_{eq}(u, v) = 1$ այն և միայն այն դեպքում, երբ $u = v$:

2. Հավասար երկարության պրեդիկատ $P_{eql}(u, v) = 1$ այն և միայն այն դեպքում, երբ համեմատվող բառերը ունեն նույն երկարությունը՝ $|u| = |v|$:

3. Ֆակտոր պրեդիկատ՝ $P_{fact}(u, v) = 1$ այն և միայն այն դեպքում, երբ u բառը հանդիսանում է v-ի ֆակտորը:

4. Հաջորդական ենթաբառ պրեդիկատ՝ $P_{sep}(u, v) = 1$ այն և միայն այն դեպքում, երբ u բառը հանդիսանում է v-ի հաջորդական ենթաբառ:

5. Զուգահեռ ենթաբառ պրեդիկատ՝ $P_{pep}(u, v) = 1$ այն և միայն այն դեպքում, երբ u բառը հանդիսանում է v-ի զուգահեռ ենթաբառ:

6. Տեղաղրության պրեդիկատ՝ $P_{perm}(u, v) = 1$ այն և միայն այն դեպքում, երբ u բառը հանդիսանում է v-ի տեղաղրությունը, այսինքն բառերն ունեն նույն երկարությունը և հավասար քանակով այբուբենի յուրաքանչյուր տառից:

4 Փորձարկումներ

OTAS ծրագրային գործիքը ներդրված է բարձր արտադրողականություն ունեցող անձնական օգտագործման կոմպյուտերների ArmCluster հաշվողական կլաստերի միջավայրում:

Գեներացվող զուգահեռ ծրագրային մոդուլի զուգահեռության էֆեկտիվությունը ստուգելու համար ArmCluster բազմապրոցեսորային հաշվողական համակարգի վրա կատարվել են հետևյալ խնդրի փորձարկումները: Տեքստում հանդիպում է* արդյոք նմուշի որևէ բառ երկու անգամ հաջորդաբար:

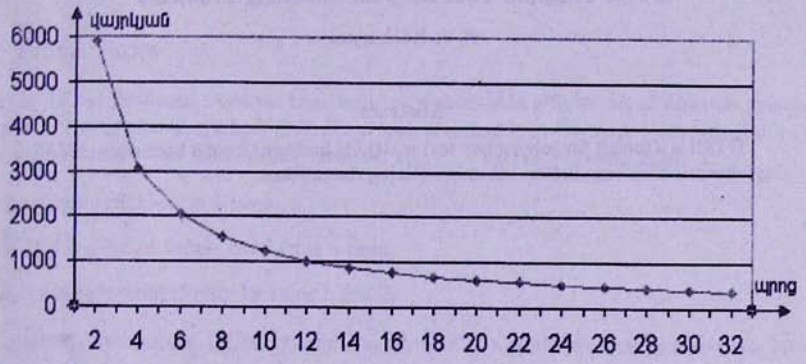
Փորձարկումները կատարվել են տեքստի և նմուշի երեք տարբեր երկարությունների համար 2,4,...,32 պրոցեսորների վրա:

Աղյուսակ 1-ում բերված են OTAS ծրագրային գործիքի միջոցով գեներացված ծրագրային մոդուլի աշխատանքի ժամանակը կախված պրոցեսորների քանակից: Երեք փորձարկումների դեպքում որպես համեմատվող տեքստ և նմուշ վերցվել են նույն հաջորդականությունները, որոնք առաջի փորձում ունեն 26510 սիմվոլ, երկրորդում՝ 122910 սիմվոլ և երրորդում՝ 1207410 սիմվոլ:

Աղյուսակ 1

Պ.Ք/S.Ե	26510	122910	1207410
2	269	5895	568222
4	139	3054	295375
6	94	2061	199073
8	71	1538	150334
10	57	1226	120938
12	48	1020	100795
14	41	876	86642
16	35	768	75760
18	32	682	67542
20	29	615	60864
22	27	559	55331
24	24	512	50759
26	22	473	46866
28	21	440	43572
30	19	410	40606
32	18	385	38069

Նկար 9-ում ցույց է տրված 122910 սիմվոլ պարունակող տեքստերով 2-ից 32 պրոցեսորների վրա կատարված փորձարկումների դեպքում ծախսված ժամանակի կախվածությունը օգտագործվող պրոցեսորների քանակից:



Նկար 9: Զուգահեռ ծրագրի աշխատանքի ժամանակի կախվածությունը պրոցեսորների քանակից

Փորձարկումների արդյունքները հիմնավորում են OTAS ծրագրային գործիքի սինթեզած ծրագրային մոդուլի համարյա իդեալական զուգահեռացումը: Սինթեզված ծրագրային մոդուլը աշխատանքի ընդացքում յուրաքանչյուր վայրկյանում մոտավորապես վերամշակում է $1,5 \times 10^4 \times N$ սիմվոլ մուտքային ֆայլերից, որտեղ N-ն աշխատող պրոցեսորների քանակն է:

Օգտագործված գրականություն

[1] K.Inoue and A.Nakamura. Some properties of two-dimensional on-line tessellation acceptors. *Information Sciences*,vol.13, p.95-121, 1977.

[2] K.Inoue and A.Nakamura. A Survey of two-dimensional automata theory. In *Proc. 5th Int. Meeting of Young Computer Scientists*.J Dasson and J.Kelemen (Eds.), p. 72-91. Lecture Notes in Computer Science 381, Springer-Verlag, Berlin, 1990.

[3] D.Giammarresi and A.Restivo. Two-dimensional languages. In *Handbook of Formal Language Theory*, v.3, Springer-Verlag, N.Y.,1996.

[4] D.Giammarresi, A.Restivo, S.Siebert and W.Thomas. Monadic second order logic over rectangular pictures and recognizability by tiling systems. *Information and computation*. Vol.125, No. 1, p. 32-45, 1996.

[5] L. Boasson, P. Cegielski, I. Guessarian, Y. Matiyasevich. Window accumulated subsequence matching is linear. *Annals of Pure and Applied Logic* Vol. 113(2001),pp.59-80.

[6] K. V. Shahbazyan, Yu. H. Shoukourian. The Finite-state Recognizability of sequences of Integers. *Transactions of the Institute for Informatic and Automation Problems of NAS RA*. Vol 27,(2006), 151-173

[7] A. V. Kocharyan, K. V. Shahbazyan. On-line Tessellation automata as a text Matching Problems Solver. *Transactions of the Institute for Informatic and Automation Problems of NAS RA*. Vol 27,(2006), 54-62.

[8] C.T. Djamegni and M. Tchuente. A New Dynamic Programming Algorithm on two dimensional arrays. *Parallel Processing Letters*, 10(1)(2000) 15-27.

OTAS Program Tool for Text Matching Problems

A.V. Kocharyan

Abstract

OTAS is a toolkit for solving any text matching problems from a fixed class. OTAS synthesize online tessallation automata solving the problem.