**UNIVERSITATEA BUCUREȘTI**
**FACULTEA DE MATEMATICĂ ȘI INFORMATICĂ**

# Aspecte Algoritmice ale unor Operații de Inspirație Biologică

**Rezumat**

*Îndrumător:*
Prof.univ.dr. Victor MITRANA

*Doctorand:*
Adrian Marius Dumitran

**București**
**2015**

# Capitolul 1

# Rezumat

## 1.1 Introducere

În teza discutăm predominant despre trei operații de inspirație biologică, și anume duplicarea prefix-sufix, duplicarea prefix-sufix mărginită si completarea de pătrate prefix-sufix.

Duplicarea este una din cele mai frecvente și mai puțin înțelese mutații în rearanjarea genomului [24]. În mare, duplicarea este procesul prin care o secvență de ADN este duplicată obținând două sau mai multe copii. Acest proces se poate intâmpla în orice poziție din cromozom, inclusiv la început și la sfârșit. De fapt, distribuția acestor repetiții variază mult între diverși cromozomi și unii autori consideră că aproximativ 5% din rearanjările genomului sunt diverse tipuri de duplicații [26].

Se consideră că în analiza filogenetică, care ar putea fi folosită în investigarea evoluției speciilor, este posibil ca prin studiul duplicărilor din cadrul genomului să se determine cea mai probabilă evoluție a duplicărilor [27]. Detectarea repetițiilor tandem și găsirea de algoritmi pentru reconstrucția repetițiilor tandem sunt subiecte care au primit multă atenție în bioinformatică [2, 1, 25]. Telomerii sunt un tip special de duplicări care apar doar la sfârșitul cromozomilor. În general, telomerii sunt formați din repetiții tandem ale unui număr mic de nucleotide. Se consideră că telomerii sunt complexe de proteine-ADN cu rol protectiv, ei se află la sfârșitul cromozonilor eukarioți și stabilizează structura cromozomilor [5, 24]. Lungimea ADN-ului telomeric este importantă pentru stabilitatea cromozomului; pierderea secvențelor de repetiție telomerică poate duce la fuziunea intre cromozomi și pot cauza instabilitatea lor.

Interpretarea duplicărilor ca operații formale pe cuvinte a inspirat mai multe lucrări în Combinatorica pe Cuvinte și în Limbaje Formale, începând cu [3, 14] și continuând cu [8, 28, 22] și cu referințele din aceste lucrari. Este de menționat că o teză de doctorat [21] studiază acest subiect. Aceste lucrări sunt punctul

de plecare pentru această teză.

Duplicările care apar la ambele capete ale unui cuvânt denumite duplicări prefix-sufix au fost prima dată considerate în [16]. Motivația principală pentru introducerea acestei operații a fost modelarea matematică a telomerilor. Altă motivare a fost modelarea procesului de generare a repetițiilor terminale lungi (LTRs): secvențe identice de ADN care se repetă de sute sau mii de ori la capetele unor secvențe de ADN. Astfel de secvențe sunt folosite, spre exemplu, de virusi pentru a-și introduce materialul genetic în genomul gazdei.

În [16] este investigată clasa limbajelor care pot fi definite prin aplicarea iterativă a duplicărilor prefix-sufix; de asemenea, clasa acestor limbaje este comparată cu alte clase de limbaje cunoscute. Se arată că limbajele din acastă clasă au o structură complicată chiar dacă șirul inițial este relativ simplu; anume, se arată că există cuvinte inițiale binare care generează limbaje care nu sunt independente de context. De asemenea, în respectiva lucrare sunt prezentați algorimti care rezolvă problema apartenenței (apartenența unui cuvânt la limbajul generat prin duplicare prefix-sufix pronind de la alt cuvânt), precum și algoritmi care calculează distanța prefix-sufix între două cuvinte. În acest context am considerat o variantă mărginită a acestei operații, numită duplicarea prefix-sufix mărginită. Noul model ne-a permis să rezolvăm unele probleme rămase deschise în lucrarea sus-menționată. De asemenea, noul model pare mai aproape de realitatea biochimică care a inspirat definiția acestei operații, pare mai aproape de adevărul biologic ca duplicările prefix-sufix să nu poată fi oricât de lungi.

În același context al operației de duplicare prefix-sufix am introdus operația de completare de pătrate prefix-sufix în [10]. Dacă în cazul operației de duplicare prefix-sufix se obțin pătrate la sfârșitul unui cuvânt prin duplicarea rădăcinii pătratului, în [10] am luat în calcul altă posibiltate, și anume am luat în considerare posiblitatea de a crea pătrate (cel mai simplu tip de repetiții) la capetele unui cuvânt prin completarea unui prefix sau a unui sufix ale acesutia până la un pătrat. De exemplu, cuvântul *banan* se poate completa la ba**nana**, de asemenea, se poate completa și cu o duplicare a întregului pătrat ca în cazul duplicării prefix-sufix (putem obține astfel de exemplu cuvântul **baba**nan ).

În contextul operațiilor menționate anterior ni s-a părut interesant să vedem cum putem folosi aceste operații pentru a genera cuvinte infinite [10]. Arătăm întâi că șirurile infinite Fibonacci, Period-doubling și Thue-Morse pot fi generate prin completarea sufix a pătratelor. Acest lucru arată o proprietate care pare interesantă pentru noi: orice cuvânt (infinit) generat prin completarea sufix a pătratelor cuprinde pătrate, dar există (o infinitate) de cuvinte generate prin aceasta operație care evită repetițiile de exponent rațional mai mare ca 2. În comparație, arătam că șirul infinit Thue-Morse nu poate fi generat prin duplicare prefix-sufix. Totuși, arătam că putem genera un cuvânt infinit liber

de cuburi prin duplicare sufix. Aceasta este o variantă mai slabă a rezultatului obținut pentru operația de completare de pătrate: orice cuvânt (infinit) generat prin duplicare sufix conține pătrate, dar există (o infinitate) de cuvinte generate de această operație care evită repetițiile de exponent întreg mai mare ca 2.

Ulterior investigăm problema detectării eficiente a existenței structurilor repetitive care apar la capetele unei secvențe. De exemplu, cuvintele care nu încep și nu se termină cu o repetiție pot modela secvențe de ADN care au trecut printr-un proces degenerativ care a distrus repetițiile terminale, afectându-i astfel stabilitatea sau funcțiile. Îndepărtându-ne de motivația biologică, cuvintele care nu încep și nici nu se termină cu repetiții par interesante și din puncte de vedere combinatoric. Într-adevăr, cuvintele ce nu conțin repetiții sunt foarte importante în combinatorica pe cuvinte, algoritmica pe cuvinte, precum și în aplicațiile acestora [23, 17]; cuvinte care nu au prefixe sau sufixe repetitive modelează o variantă mai slabă dar foarte strâns legată a acestei noțiuni. În [11] dar și în această teză arătăm cum putem să enumărăm factorii liberi de pătrate-prefix-sufix dintr-un cuvânt, cum sa numărăm acești factori sau cum să-l găsim pe cel mai lung asemenea factor.

Păstrând motivația biologică discutăm în continuarea tezei despre repetiții și repetiții palindromice spațiate(o repetiție spațiată este de exemplu cuvântul **arțar**). Repetițiile și palindoramele spațiate sunt investigate de mult timp ([17, 4, 20, 18, 19, 6, 7]), motivația venind în special din analiza structurilor de ADN și ARN, unde repetițiile tandem și cele hairpin joacă un rol important în a arăta structura și funcționalitatea secvențelor genetice analizate (vezi [17, 4, 18]). Mai precis, o repetiție spațiată (respectiv o repetiție palindormică spațiată) care apare într-un cuvânt $w$ este un factor $uvu$ (respectiv, $u^Rvu$) al $w$. Partea din mijloc $v$ a acestei structuri se numeste spațierea, în timp ce cei doi factori $u$ (respectiv, $u^r$ și $u$) se numesc brațele stânga și dreapta. În general, lucrările precedente erau interesate să găsească toate repetițiile și repetițiile palindromice spațiate, sub anumite restrictii asupra lungimii spațierii sau în contextul unei relații între braț și spațiere. În teză (precum și în [9]) am rezolvat această problema în trei contexte noi.

Trebuie notat că scopul acestor investigații nu este acela de a ataca probleme biologice reale și de a le rezolva. De fapt, scopul cercetării este acela de a înțelege mai bine proprietățile structurale ale cuvintelor obținute prin operațiile discutate precum și de obținerea unor unelte eficiente pentru a manipula asemenea șiruri.

## 1.2 Structura tezei

Teza este împărtită în cinci capitole după cum urmează:

1. Primul capitol **"Introducere"**, începe prin o prezentare a motivației tezei

și o plasare a rezultatelor în contextul cercetării în domeniu. În a doua parte a capitolului sunt prezentate contribuțiile personale.

2. În al doilea capitol, **"Preliminarii"**, definim majoritatea notațiilor folosite în teză. Capitolul este împărțit în patru secțiuni:

   (a) Prima secțiune prezintă definiți clasice despre cuvinte finite și infinite, factori, perioade, repetiții maximale și automate finite.

   (b) Secțiunea a doua introduce operațiile de inspirație biologică cu care lucrăm în teză.

   (c) În a treia secțiune sunt prezentate diverse structuri de date folosite în teză.

   (d) A patra și ultima secțiune prezintă câteva leme folosite frecvent în cadrul tezei.

3. În al treilea capitol **"Aspecte algoritmice și de teoria limbajelor"** prezentăm o mare parte din rezultatele principale punând accent după cum spune și titlul capitolului pe rezultate legate de algoritmi și limbaje de duplicare. Capitolul este împărțit în trei secțiuni astfel:

   (a) În prima secțiune discutăm despre diverse limbaje de duplicare.

   (b) În a doua secțiune discutăm despre probleme de apartanența în contextul celor trei operații. De asemenea, arătăm cum putem calcula eficient rădăcinile unui cuvânt, precum și rădăcinile comune a două cuvinte în raport cu operațiile studiate.

   (c) În a treia secțiune discutăm probleme legate de calculul distanțelor între două cuvinte (sau două limbaje de duplicare) în raport cu operațiile noastre.

4. Al patrulea capitol **"Aspecte combinatorice"** include o parte importanta a rezultatelor de ordin algorimtic și combinatorial. Acest capitol este împărțit în trei secțiuni:

   (a) În prima secțiune arătăm cum putem genera cuvinte finite și infinite folosind operațiile definite anterior.

   (b) În a doua secțiune arătăm cum putem detecta în mod eficient existența structurilor repetitive care apar la capetele unei secvențe.

   (c) În a treia secțiune rezolvăm problema repetițiilor spațiate și a repetițiilor palindormice spațiate în trei cazuri diferite.

5. În ultimul capitol **"Direcţii de cercetare"** prezentăm o serie de probleme deschise care apar în urma rezultatelor din teză.

## 1.3 Rezultate principale

După ce am discutat despre motivația muncii noastre și am prezentat pe scurt rezultate din această lucrare, acum vom prezenta în detaliu o parte din aceste rezultate.

Cu toate că majoritatea rezultatelor din această teză apar în capitolele trei și patru, trebuie menționat că considerăm că lemele din secțiunea 2.4 sunt importante. Nu numai pentru că multe din rezultatele din teză se bazează pe ele, dar și pentru că tehnica folosită pentru a le demonstra ar putea fi utilă pentru a demonstra și alte rezultate. În această sectiune arătăm cum putem calcula în timp liniar lungimea celui mai scurt, precum și a celui mai lung pătrat care începe sau se termină în fiecare poziție dintr-un cuvânt. În aceeași maniera arătam cum putem calcula pătratul de lungime minimă/maximă centrat în fiecare poziție precum și alte rezultate similare. Cu toate că unele din aceste rezultate au fost demonstrate anterior, noi aducem o soluție diferită și din anumite perspective mai generala și mai eficientă. Soluția pe care o prezentăm este bazată pe factorizarea Lempel-Ziv a cuvântului inițial și pe o serie de observații combinatoriale asupra structurii factorizării. Ca să putem demonstra aceste leme folosim de asemenea un caz special al problemei pădurilor de mulțimi disjuncte [15]. Prin urmare, la începutul sectiunii demonstrăm o lemă care arată cum putem folosi acest caz a problemei în contextul tezei noastre. Aceste rezultate au fost publicate în [10, 9].

În secțiunea 3.1 prezentăm câteva rezultate legate de limbaje de duplicare care au fost publicate în [12, 13]. Începem prin a da un rezultat care separă limbajele de duplicare generate de duplicarea mărginită prefix-sufix de limbajele de duplicare generate de duplicarea prefix-sufix:

**Theorem 1.3.1** *Mulțimea claselor nevide de limbaje închise la reuniune cu limbaje regulate, intersecție cu limbaje regulate, și substituție cu limbaje regulate este închisă la duplicarea mărginită prefix-sufix.*

În continuare demonstrăm că dacă fixăm o valoare $k$ pentru mărginire, putem determina dacă un limbaj este un limbaj obținut prin duplicarea mărginită prefix-sufix a unei mulțimi finite de cuvinte, de asemenea, putem determina mulțimea minimă dacă aceasta există.

**Theorem 1.3.2** *Fie $L$ un limbaj regulat care este un limbaj de duplicare prefix-sufix $k$-mărginită pentru un anume $k$ pozitiv. Atunci există și este minimal un limbaj regualt $E$, care poate fi calculat, astfel încât $L = PSD_k^*(E)$. În particular, se poate decide dacă un limbaj $L$ este un limbaj de duplicare, prefix-sufix $k$-mărginită, finit.*

În secțiunea 3.2 începem prin a rezolva problema apartenței în cazul limbajelor $PSD_k^*(L)$, în contextul în care știm deja să rezolvăm problema apartenenței pentru $L$.

**Theorem 1.3.3** *Dacă problema apartenenței pentru limbajul $L$ poate fi decisă în complexitatea timp $\mathcal{O}(f(n))$, atunci problema apartenței pentru $PSD_k^*(L)$ poate fi decisă în complexitatea timp $\mathcal{O}(nk\log(k) + n^2 f(n))$.*

Pornind de la rezultatul anterior aratăm că putem decide dacă un cuvânt este în $PSD_k^*(w)$, cu $|x| = n$ în complexitate timp $\mathcal{O}(n\log(k))$ dacă $|w| \geq k$ și în $\mathcal{O}(nk\log(k))$ altfel. Rezultatele anterioare au fost publicate în [12, 13, 10].

Trecem apoi la calcularea rădăcinilor unui cuvânt și obținem următorul rezultat pentru operația de completare prefix-sufix a pătratelor:

**Theorem 1.3.4** *Pentru un cuvânt $w$ de lungime $n$, putem calcula:*

1. *Numărul de strămoși ai lui $w$*
2. *Cel mai scurt strămoș a lui $w$*

*în relație cu operația de completare de pătrate prefix-sufix în complexitatea timp $\mathcal{O}(n)$.*

Rezultatul anterior ne ajută să obținem un algoritm care rezolvă problema apartenței pentru această operație în timp liniar.

Ulterior, arătăm cum putem calcula o rădăcina comună a două cuvinte în complexitatea timp $\mathcal{O}(nk\log(k) + n^2)$ în cazul operației de duplicare prefix-sufix și în timp liniar în cazul operației de completare de pătrate prefix-sufix. Pentru a obține aceste rezultate folosim diverse tehnici de la sortări în timp liniar, la vectori de sufixe, întrebări LCP, precum și lemele din secțiunea 2.4. Folosind aceleași unelte, precum și căutarea binară, obținem un algoritm de complexitate timp $\mathcal{O}(n\log(n))$, pentru a găsi cea mai scurtă rădăcină comună a dou;a cuvinte în relație cu operația de completare de pătrate prefix-sufix. Majoritatea rezultatelor legate de strămoși au fost publicate în [10, 12, 13, 11], iar rezultatele legate de strămoși comuni apar în teză pentru prima dată.

În secțiunea 3.3 prezentăm la început un algoritm de complexitate timp $\mathcal{O}(n\log(k))$, care calculează distanța de duplicare prefix-sufix între două cuvinte, folosind programare dinamică, structura de date deque și o serie de remarci combinatorice legate de repetițiile maximale dintr-un cuvânt.

**Theorem 1.3.5** *Fie $k \geq 1$, $x$ și $w$ două cuvinte de lungime $m$ și $n$, cu $n > m$. Dacă $m \geq k$, atunci $\delta_k(x, w)$ poate fi calculată în complexitate timp $\mathcal{O}(n\log(k))$. Dacă $m < k$, atunci $\delta_k(x, w)$ poate fi calculată în complexitate timp $\mathcal{O}(nk\log(k))$.*

Continuăm oferind o soluție total diferită pentru a rezolva problema calculării distanței între două cuvinte de complexitate timp $\mathcal{O}(n^3)$. Soluția este generală și poate fi folosită pentru oricare din operațiile studiate și se bazează pe

algoritmi pe grafuri. Apoi arătăm cum putem îmbunătăți acest algoritm pentru a obtine o soluție pătratică pentru aceasta problemă în cazul completării de pătrate prefix-sufix. Terminăm secțiunea cu un rezultat care arată că putem calcula distanța între două limbaje de duplicare mărginită prefix-sufix.

**Theorem 1.3.6** *Fie două limbaje regulate $L_1$ și $L_2$ peste un alfabet $V$, recunoscute de două automate finite cu mulțimea stărilor $Q$ și $S$, și o valoare $k \geq 1$, distanța $\delta_k(L_1, L_2)$ poate fi calculă în complexitate timp $\mathcal{O}((k+N)M^2|V|^{2k})$ time, unde $M = \max\{|Q|, |S|\}$ și $N = \min\{|Q|, |S|\}$.*

Majoritatea rezultatelor din această secțiune au fost publicate în [10, 12, 13].

În secțiunea 4.1 arătăm cum putem genera diferite cuvinte infinite (cum ar fi cuvintele Fibonacci și Thue Morse), folosind completarea de pătrate sufix. De asemenea, demonstrăm că nu putem genera cuvântul Thue Mrose prin duplicare prefix-sufix, dar putem genera secventă Stewart folosind această operație. Rezultatele au fost publicate în [10].

În secțiunea 4.2 arătăm cum putem obține în timp liniar o structură de date care ne permite să verificăm în timp constant dacă un factor al unui cuvânt este $SD$, $PD$, or $PSD$ primitiv(nu poate fi obținut din nici un alt cuvânt prin o operație de duplicare prefix/sufix/prefix-sufix). Apoi prezentăm un algoritm care găsește toți factorii liberi de pătrate prefix-sufix dintr-un cuvânt:

**Theorem 1.3.7** *Dacă se dă un cuvânt $w$ de lungime $n$, putem găsi setul $S$ de factori liberi de pătrate prefix-sufix a lui $w$ în complexitate timp $\mathcal{O}(n + |S|)$.*

Continuăm prin a prezenta un algoritm de complexitate timp $\mathcal{O}(n \log n)$ pentru a număra factorii liberi de pătrat prefix-sufix. Algoritmul folosește arbori de intervale, precum și lemele din secțiunea 2.4. O soluție similară este dată pentru a găsi cea mai lungă rădăcină a unui cuvânt în relație cu operația de completare de pătrate prefix-sufix. Ulterior, arătăm că cel mai lung factor liber de pătrat prefix-sufix al unui cuvânt poate fi calculat în timp liniar, deci fără a enumera setul $S$ calculat anterior.

**Theorem 1.3.8** *Dacă se dă un cuvânt $w$ de lungime $n$, putem găsi cel mai lung factor liber de pătrat prefix-sufix în timp liniar.*

Secțiunea se temrina cu căteva teoreme care oferă diverse factorizări a unui cuvânt inclusiv cea în pătrate consecutive:

**Theorem 1.3.9** *Dacă se dă un cuvânt $w$ de lungime $n$, putem găsi (dacă există) în complexitate timp $\mathcal{O}(n \log(n))$ o factorizare $w = s_1 \cdots s_k$ of $w$, astfel încât $s_i$ este pătrat pentru oricare $1 \leq i \leq k$.*

Majoritatea rezultatelor din această secțiune au fost publicate în [11], dar există și rezultate care au fost publicate în teză pentru prima dată.

În secțiunea 4.3 ne uităm la factori de forma $uvu$ and $u^rvu$ și calculăm doi vectori care ne dau pentru fiecare pozitie cel mai lung factor care începe pe acea poziție și care a apărut deja în vector, respectiv cel mai lung factor care începe pe acea poziție și imaginea sa în oglinda a apărut deja în vector ($LPF$, resepctiv $LPrF$) în trei cazuri diferite. În primul rând, luăm cazul în care $v$ este limitat și inferior și superior de constante. În acest caz, obținem vectorul $LPrF$ în timp liniar și vectorul $LPF$ în complexitate timp $\mathcal{O}(n \log n)$. Ca să obținem aceste rezultate folosim diverse tehnici combinatorice, vectori de sufixe, lemele din secțiunea 2.4, precum și dicționarul de factori simpli (DBF). Continuăm prin a calcula cei doi vectori în cazul în care $v$ este limitat inferior de o funcție ce depinde de poziția din vector. În acest caz, am reusit să obținem doi algoritmi liniari. Ultimul caz a fost cazul repetițiilor cu braț lung. În acest caz, am obținut doi algoritmi de complexitate timp $\mathcal{O}(n + |output|)$, respectiv $\mathcal{O}(n)$. Rezultatele din această secțiune au fost publicate în [9].

# Bibliografie


[1] Gary Benson. Tandem repeats finder: a program to analyze dna sequences. *Nucleic acids research*, 27(2):573, 1999.

[2] Gary Benson and Lan Dong. Reconstructing the duplication history of a tandem repeat. In *ISMB*, pages 44–53, 1999.

[3] Daniel P. Bovet and Stefano Varricchio. On the regularity of languages on a binary alphabet generated by copying systems. *Information Processing Letters*, 44(3):119 – 123, 1992.

[4] Gerth Stolting Brodal, Rune B. Lyngso, Christian N. S. Pedersen, and Jens Stoye. Finding maximal pairs with bounded gap. In *Proc. 10th Combinatorial Pattern Matching*, volume 1645 of *LNCS*, pages 134–149. Springer, 1999.

[5] S.R. Chan and E.H. Blackburn. Telomeres and telomerase. *Philos. Trans. R. Soc. Lond. B. Biol. Sci.*, 359:109–121, 2004.

[6] Maxime Crochemore, Costas S. Iliopoulos, Marcin Kubica, Wojciech Rytter, and Tomasz Walen. Efficient algorithms for two extensions of LPF table: The power of suffix arrays. In *Proc. SOFSEM 2010*, volume 5901 of *LNCS*, pages 296–307, 2010.

[7] Maxime Crochemore and German Tischler. Computing longest previous non-overlapping factors. *Inf. Process. Lett.*, 111(6):291–295, February 2011.

[8] J. Dassow, V. Mitrana, and Gh. Paun. On the regularity of duplication closure. *Bull. EATCS*, 69:133–136, 1999.

[9] M. Dumitran and F. Manea. Longest gapped repeats and palindromes. In *Proc. MFCS 2015*, LNCS, to appear 2015.

[10] M. Dumitran and F. Manea. Prefix-suffix square completion. In *Proc. WORDS 2015*, LNCS, to appear 2015.

[11] M. Dumitran, F. Manea, and D. Nowotka. On prefix/suffix-square free words. In *Proc. SPIRE 2015*, LNCS, to appear 2015.

[12] Marius Dumitran, Javier Gil, Florin Manea, and Victor Mitrana. Bounded prefix-suffix duplication. In *Proc. CIAA 2014*, volume 8587 of *LNCS*, pages 176–187, 2014.

[13] M. Dumtiran, J. Gil, F. Manea, and V. Mitrana. Bounded prefix-suffix duplication: language theoretic and algorithmic results. *International Journal of Foundations of Computer Science*, to appear.

[14] G. Ehrenfeucht A., Rozenberg. On the separating power of eol systems. *RAIRO - Theoretical Informatics and Applications - Informatique Thorique et Applications*, 17(1):13–22, 1983.

[15] Harold N. Gabow and Robert Endre Tarjan. A linear-time algorithm for a special case of disjoint set union. In *Proc. STOC*, pages 246–251, 1983.

[16] Jesús García-López, Florin Manea, and Victor Mitrana. Prefix-suffix duplication. *J. Comput. Syst. Sci.*, 80(7):1254–1265, 2014.

[17] Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA, 1997.

[18] Roman Kolpakov and Gregory Kucherov. Searching for gapped palindromes. *Theor. Comput. Sci.*, 410(51):5365–5373, 2009.

[19] Roman Kolpakov, Mikhail Podolskiy, Mikhail Posypkin, and Nickolay Khrapov. Searching of gapped repeats and subrepetitions in a word. In *Proc. CPM*, volume 8486 of *LNCS*, pages 212–221, 2014.

[20] Roman M. Kolpakov and Gregory Kucherov. Finding repeats with fixed gap. In *Proc. SPIRE*, pages 162–168, 2000.

[21] P. Leupold, J. Karhumäki, V. Mitrana, Universitat Rovira i Virgili. Facultat de Lletres de Tarragona, and Universitat Rovira i Virgili. Departament de Filologies Romàniques. *Languages Generated by Iterated Idempotencies: PhD Dissertation : Tesi Doctoral*. Universitat Rovira i Virgili, 2008.

[22] Peter Leupold, Victor Mitrana, and JosM. Sempere. Formal languages arising from gene repeated duplication. In Nataa Jonoska, Gheorghe Pun, and Grzegorz Rozenberg, editors, *Aspects of Molecular Computing*, volume 2950 of *Lecture Notes in Computer Science*, pages 297–308. Springer Berlin Heidelberg, 2004.

[23] M. Lothaire. *Combinatorics on Words*. Cambridge University Press, 1997.

[24] R.J. Preston. Telomeres, telomerase and chromosome stability. *Radiat. Res.*, 147:529–534, 1997.

[25] Dina Sokol, Gary Benson, and Justin Tojeira. Tandem repeats over the edit distance. *Bioinformatics*, 23(2):e30–e35, 2007.

[26] L. D. Stein. Human genome: end of the beginning. *Nature*, 431(7011):915–916, October 2004.

[27] I. Wapinski, A. Pfeffer, N. Friedman, and A. Regev. Natural history and evolutionary principles of gene duplication in fungi. *Nature*, 449:54–61, 2007.

[28] Ming wei Wang. On the irregularity of the duplication closure. *Bulletin of the EATCS*, 70:162–163, 2000.