Curriculum vitae et studiorum

Rosalba Zizza

Dicembre 2011

Indice

1 Informazioni generali

1.1 Dati personali

Nome: Rosalba
Cognome: Zizza
Luogo e data di nascita: Salerno, 11.06.1972
E-mail: zizza@dia.unisa.it
URL: http://www.dia.unisa.it/professori/zizza

1.2 Titoli di studio

- Dottore di Ricerca in Informatica, titolo conseguito in data 11.01.2002 (XIII ciclo). Sede amministrativa: Università Statale di Milano.

- Diploma di Laurea in Scienze dell'Informazione, conseguito presso l'Università degli Studi di Salerno, in data 17.07.1997 (votazione: 110/110 e lode). Titolo della dissertazione: *Sulla congettura della fattorizzazione*. Relatore: Prof.ssa Clelia De Felice.

- Diploma di Maturità Scientifica, conseguita presso il Liceo Scientifico Statale "G. Da Procida", Salerno, anno scolastico 1990/91 (votazione: 60/60).

1.3 Borse di studio

≫ **Maggio 2004 - Dicembre 2004:** Assegno per la collaborazione ad attività di ricerca relativo al settore scientifico-disciplinare INF/01, per la realizzazione del seguente Progetto di ricerca "Linguaggi formali, codici a lunghezza variabile, splicing systems", presso il Dipartimento di Informatica ed Applicazioni "R.M. Capocelli" dell'Università degli Studi di Salerno (D.R. 1145 del 1.03.2004). Responsabile: Prof.ssa Clelia De Felice.

≫ **Maggio 2002 - Maggio 2004:** Borsa di studio biennale per attività di ricerca post-dottorato nell'area disciplinare Informatica (tema: Splicing systems e teoria degli automi), presso il Dipartimento di Informatica ed Applicazioni, Università di Salerno. Data conferimento: 06.05.2002 (D.P.R. 516, 8 Febbraio 2002). Supervisore: Prof.ssa Clelia De Felice.

1.4 Posizione attuale

≫ Ricercatore confermato presso la Facoltà di Scienze MM.FF.NN., Dipartimento di Informatica, Università di Salerno (nomina: Settembre 2004; presa di servizio: 3 Gennaio 2005).

2 Attività formativa

≫ *Corsi di dottorato*

⋆ **Marzo-Giugno 1998:** *Algoritmi e complessità* (Proff. Bertoni, Mauri, Pighizzini), Univ. Statale di Milano.

⋆ **Aprile 1999:** *Sistemi informativi e tecnologie groupware* (Prof. T. Kakola, Finlandia), Università di Milano-Bicocca.

⋆ **Marzo 2000:** *Topics in Theoretical Computer Science: Randomness, Proofs and Computation* (Prof. G. Persiano), Univ. di Salerno.

√ Tutti i corsi di dottorato prevedevano un esame finale (seminario o dissertazione scritta su alcuni articoli relativi all'argomento del corso). Tutti gli esami sono stati sostenuti e superati.

≫ *Scuole estive*

⋆ **Maggio 1998:** *SNDIS98: Scuola Nazionale dei Dottorati in Informatica delle Facoltà di Scienze*, Bertinoro (Corsi: Fuzzy Logic - Prof. P. Hajek; Reti di Calcolatori - Prof. S. Gai; Apprendimento e reti neurali - Prof. A. Bertoni; Computer graphics - Prof.ssa L. De Floriani).

√ Tutti i corsi della scuola prevedevano un esame finale (esercizi o dissertazione scritta su alcuni articoli relativi all'argomento del corso). Tutti gli esami sono stati sostenuti e superati.

⋆ **Giugno 1999:** 11^{th} *International School for Computer Science Researchers on Computational Biology*, Lipari (Cicli di seminari di Biologia e Bioinformatica).

√ La scuola prevedeva un esame finale (seminario su alcuni articoli relativi ad uno degli argomenti del corso, da preparare in gruppo) da sostenere a conclusione della scuola. Esame superato.

⋆ **Settembre 2000:** *School on Quantum Computing*, Vietri S/M, Salerno (Corsi di Quantum Computing - Prof. J. Gruska; Meccanica quantistica - Prof. D. De Falco).

≫ *Cicli di seminari*

La sottoscritta ha partecipato a numerosi seminari, essenzialmente presso l'Università di Salerno, l'Università di Roma "La Sapienza", l'Università Statale di Milano e l'Università di Milano-Bicocca. Inoltre, ha partecipato ai seguenti cicli di seminari.

⋆ **Settembre 1998:** 42^{nd} *Session of "Seminaire Lotharingien de Combinatoire"*, Maratea.

⋆ **Ottobre 1999:** *Words, Formal Series, Codes and Symbolic Dynamic* (Prof. D. Perrin), durante il VI Incontro di Combinatorica Algebrica, Maratea.

3 Attività didattica, tutoraggio, incarichi

3.1 Attività didattica presso l'Università di Milano Bicocca

L'attività didattica ha riguardato il *Corso di Algoritmi e strutture dati* (II anno del Corso di Laurea in Informatica), svolta per gli anni accademici 2000-2001 e 2001-2002, presso l'Università di Milano-Bicocca. Il corso era strutturato in due moduli (Elementi e Complementi), ognuno con tre tipologie di lezioni: teoria, esercitazioni, laboratorio (lezioni frontali).

Gli argomenti trattati nel modulo di *Elementi* (*teoria*) sono stati: introduzione al concetto di algoritmo, analisi di algoritmi, notazioni asintotiche, equazioni di ricorrenza, algoritmi di ordinamento (heapsort, quicksort), ordinamento in tempo lineare, selezione, tabelle hash, alberi binari di ricerca.

Gli argomenti trattati nel modulo di *Complementi* (*teoria*) sono stati: programmazione dinamica, algoritmi greedy, teoria dei matroidi e Teorema di Rado, algoritmi elementari su grafi (visita in ampiezza e profondità , ordinamento topologico), alberi di copertura minimi (algoritmi di Kruskal e Prim), cammini minimi con sorgente singola, cammini minimi tra coppie, chiusura transitiva di un grafo orientato.

Gli argomenti trattati nei moduli sono stati completati con esercizi svolti in aula dal docente e dagli studenti (*esercitazioni*, *laboratorio*). Per i due suddetti anni accademici, la sottoscritta ha ricoperto entrambi i moduli (*Elementi*, *Complementi*) e nei tre ruoli (*teoria*, *esercitazioni*, *laboratorio*), come specificato nel seguito.

Infine, l'attività didattica svolta ha ricoperto anche la preparazione e la correzione delle prove scritte d'esame, esami orali e attività tutoria.

≫ *Anno accademico 2001-2002*

- ***Corso di Algoritmi e Strutture Dati*** (Complementi), Corso di Laurea in Informatica, II anno (12 ore): ***esercitatore*** (lezioni frontali). Docente del corso: Prof.ssa Paola Bonizzoni.

- ***Corso di Algoritmi e Strutture Dati*** (Complementi), Corso di Laurea in Informatica, II anno (12 ore): ***laboratorio*** (lezioni frontali). Docente del corso: Prof.ssa Paola Bonizzoni.

- ***Corso di Algoritmi e Strutture Dati*** (Complementi), Corso di Laurea in Informatica, II anno (40+12 ore): ***teoria ed esercitazioni***. Titolare del corso: Prof. Giancarlo Mauri.

≫ *Anno accademico 2000-2001*

- ***Corso di Algoritmi e Strutture Dati*** (Elementi), Corso di Laurea in Informatica (12 ore): ***esercitatore***. Docente del corso: Prof.ssa Paola Bonizzoni.

- ***Corso di Algoritmi e Strutture Dati*** (Complementi), Corso di Laurea in Informatica (12 ore): ***esercitatore***. Docente del corso: Prof.ssa Paola Bonizzoni.

≫ *Ulteriori prestazioni di supporto alla didattica*

- **Creazione e gestione della pagina web del corso** (in collaborazione con il Dott. A. Leporati, attualmente ricercatore presso l'Università di Milano-Bicocca), contenente avvisi, orari, scalette delle lezioni, tracce d'esame e relative soluzioni, esercizi.

- **Preparazione di dispense** con esercizi proposti e svolti, lezioni e altro materiale di supporto (anche in web).

3.2 Attività didattica presso l'Università di Salerno

≫ *Anno accademico 2010-2011*

- ***Attività di supporto per i corsi di Strutture Dati***, Corso di Laurea in Informatica, II semestre, 22 ore.

  L'attività svolta è stata di codocenza per il suddetto corso, per entrambe le classi attivate in questo anno accademico (la tipologia di attività didattica da offrire alla Facoltà è stata concordata con le effettive esigenze del Corso di laurea e in linea con la nuova modalità dell'impegno didattico per i Ricercatori in seguito alla Legge Gelmini).

≫ *Anno accademico 2009-2010*

- ***Corso di recupero di Laboratorio di Algoritmi e Strutture Dati***, Corso di Laurea in Informatica, I semestre, 24 ore.

  Le lezioni, svolte esclusivamente in laboratorio, erano state previste per sostenere gli studenti nella preparazione dell'esame, essendo un corso offerto nella laurea triennale vecchio ordinamento (ex DM 509) e quindi disattivato a partire da questo anno accademico. Gli argomenti trattati sono stati selezionati dal programma dell'anno acc. 2008-2009, ma concordati con gli studenti al fine di venire incontro alle loro principali difficoltà .

- ***Corso di Strutture Dati***, Corso di Laurea in Informatica nuovo ordinamento (ex DM 270), II semestre, II anno, 60 ore.

  Il corso tratta la definizione di noti tipi di dati astratti, precisamente Stack, Queue, Deque, ArrayList, Sequenze, Alberi generici e Alberi binari, Code a priorit, Mappe, Dizionari, Set e Partition, Grafi. Il corso prevede un'approfondita analisi delle diverse possibili implementazioni (in Java), usando array, liste, tabelle hash, a seconda delle strutture considerate, che sono messe a confronto analizzando la complessità in spazio e tempo.

  Le lezioni sono state corredate da esercitazioni sotto sorveglianza del docente, simulazione delle prove d'esame, esercitazioni autonome. L'esame si svolge in laboratorio, con una prova direttamente al calcolatore.

  Il corso ha un sito web contenente avvisi, orari, scalette delle lezioni, tracce d'esame e relative soluzioni, dispense con esercizi proposti e svolti, lezioni e altro materiale di supporto.

≫ *Anno accademico 2005-2006, 2006-2007, 2007-2008, 2008-2009*

- ***Corso di Laboratorio di Algoritmi e Strutture Dati***, Corso di Laurea in Informatica e in Informatica Applicata, I semestre, II anno, 64 ore.

  Il corso è organizzato in due parti:

  √ *Lezioni di Laboratorio.* Sono presentati alcuni tipi di dati astratti, algoritmi e alcuni design pattern implementati in Java. Il programma svolto ha riguardato: stack, code, deque, liste a puntatori semplici e doppie, vettori, sequenze, iteratori, alberi binari e generali, insiemi e partizioni, mappe, code a priorità , grafo.

  √ *Lezioni frontali teoriche.* Alcuni algoritmi su grafi: visite BFS e DFS, Minimum Spanning Tree (Algoritmi di Kruskal e Prim), Algoritmo di Dijkstra, Algoritmo di Floyd-Warshall.

  Le lezioni sono state corredate da esercitazioni sotto sorveglianza del docente, simulazione delle prove d'esame, esercitazioni autonome.

  Il corso ha un sito web contenente avvisi, orari, scalette delle lezioni, tracce d'esame e relative soluzioni, dispense con esercizi proposti e svolti, lezioni e altro materiale di supporto.

  A partire dall'anno accademico 2006-2007, l'esame si è svolto in laboratorio, con una prova direttamente al calcolatore.

≫ *Anno accademico 2004-2005*

- ***Corso di Esercitazioni di Fondamenti di Programmazione***, Corso di Laurea in Informatica e in Informatica Applicata, II semestre, I anno, 24 ore.

  √ Le lezioni (frontali) hanno riguardato esercitazioni su induzione e induzione strutturale, notazione asintotica e valutazione della complessità di algoritmi, invarianti di ciclo, strutture dati, automi finiti deterministici.

3.3 Attività di tutoraggio

In qualità di

≫ *supporto alla preparazione della Tesi di Laurea*

- Dott. Rocco Zaccagnino, Tesi di Laurea in Informatica v.o., Titolo: Sistemi splicing circolari, anno acc. 2003-2004.

≫ *co-relazione della Tesi di Laurea*

- Dott. Nicola Donadio, Tesi di Laurea in Informatica v.o., Titolo: Codici UD, MSD e SD su 3 elementi, anno acc. 2004-2005.

- Dott. Marco Frasca, Tesi di Laurea in Informatica v.o., Titolo: L'operazione di composizione nella famiglia dei codici massimali prefissi, anno acc. 2004-2005.

≫ *relatore della Tesi di Laurea in Informatica v.o.*

- Dott.ssa Giusi Maiorano, Titolo: Code word design problem via Linguaggi Formali, anno acc. 2005-2006.

- Dott.ssa Anna Maria D'Alia, Titolo: Il Road Coloring Problem: dalla congettura alla soluzione, anno acc. 2010-2011.

≫ *Tutor del Tirocinio (interno) e/o relatore della Tesi di Laurea triennale in Informatica*

- Dott. Luca D'Auria, Titolo: Implementazione in Java di un algoritmo di Capocelli-Hoffmann per decidere l'univoca decifrabilità e altre proprietà di un insieme finito di parole, anno acc. 2005-2006.

- Dott. Carmine Giordano, Titolo: Implementazione in Java di un algoritmo di McCreight per la costruzione dei suffix tree, anno acc. 2006-2007.

- Dott. Antonio Rinaldi, Titolo: Implementazione in Java di un modello generativo di parole ispirato da un meccanismo biologico: il caso dei sistemi splicing circolari (1,3), anno acc. 2008-2009.

- Dott. Ciro Galluccio, Titolo: Implementazione in Java di un modello generativo di parole ispirato da un meccanismo biologico: dall'analisi di un caso particolare al caso generale dei sistemi splicing circolari finiti, anno acc. 2008-2009.

- Dott. Marco Agostino, Titolo: Implementazione in Java di un pacchetto per gestire le parole circolari e analizzare loro proprietà , anno acc. 2008-2009.

- Dott. Daniele Vece, Titolo: Implementazione in Java dell'algoritmo di Rodeh per decidere l'univoca decifrabilità di un insieme finito di parole: uso del suffix tree e sue estensioni, anno acc. 2009-2010.

- Dott. Marco Bisignano, Titolo: Integrazione in JFLAP di un meccanismo generativo di parole circolari ispirato dalla biologia, anno acc. 2009-2010.

- Dott. Antonio Valva, Titolo: Integrazione in JFLAP di un pacchetto per gestire Parole Circolari, anno acc. 2009-2010.

≫ *relatore di Tesi di Laurea Specialistica in Informatica*

- Dott. Luca D'Auria, Titolo: Progetto, Analisi e Implementazione in Java di un algoritmo efficiente per decidere l'univoca decifrabilità di linguaggi regolari, anno acc. 2009-2010.

La sottoscritta ha seguito anche varie tesi esterne (derivate da attività di tirocinio presso aziende).

3.4 Incarichi

Dal 18.01.2007 la sottoscritta è membro della Commissione di Orientamento in Ingresso dell'Area Didattica di Informatica, Facoltà di Scienze MM.FF.NN. dell'Università di Salerno. Le funzioni principali della Commissione riguardano la pianificazione, l'attuazione ed il monitoraggio delle azioni di divulgazione sul territorio delle attività scientifiche e didattiche dell'Area Informatica (Facoltà di Scienze).

La sottoscritta è stata membro della commissione dell'esame per l'ammissione al Corso di Dottorato di Ricerca in Informatica (XIII ciclo, n.o., 2011).

4 Attività scientifica

4.1 Membership

- EATCS (European Association of Theoretical Computer Science)
- GRIN (Gruppo di Informatica)

4.2 Partecipazione a gruppi di ricerca

◇ Bioinformatics and Molecular Computing group (Università di Milano-Bicocca), 1999-2005.

◇ EMCC (European Molecular Computing Consortium), Presidente: G. Rozenberg. URL: http://openit.disco.unimib.it/emcc/

◇ Contatti di ricerca: Ion Petre (Univ. di Turku, Finlandia); Robert Nowak (Univ. di Varsavia, Polonia); Isabelle Fagnot (IGM, Univ. Marne-la-Vallée, Parigi, Francia) ospite del Dipartimento di Informatica ed Applicazioni, Università di Salerno, Gennaio 2004.

◇ Incontri di ricerca: Elena Pribavkina (Ural State University of Ekaterimburg) ospite del Dipartimento di Informatica ed Applicazioni, Università di Salerno, marzo 2006; Jacques Sakarovitch (CNRS France), ospite del Dipartimento di Informatica ed Applicazioni, Università di Salerno, maggio 2006.

◇ Incontro con il Prof. H.J. Hoogeboom (Univ. di Leiden, Olanda), presso Dipartimento di Computer Science (LIACS), Leiden University, 1-4 Dicembre 2003.

4.3 Collaborazioni

◇ *Università di Milano-Bicocca*: Prof.ssa Paola Bonizzoni, Dott. Claudio Ferretti, Giancarlo Mauri.

◇ *Università di Salerno*: Prof.ssa Clelia De Felice, Dott. Gabriele Fici.

◇ *Università di Parigi, Francia*: Dott. Sergey Verlan.

4.4 Presentazione attività di ricerca in Italia e all'estero

≫ La sottoscritta ha presentato l'attività di ricerca nei seguenti incontri dell'EMCC e del progetto europeo MolCoNet.

- Milano, Novembre 2000
- Milano, Dicembre 2001
- Metz (Francia), Maggio 2002
- Budapest (Ungheria), Novembre 2002

≫ La sottoscritta ha presentato l'attività di ricerca nei seguenti incontri nell'ambito dei progetti nazionali cofinanziati.

- Firenze, Novembre 1999;
- Milano, Dicembre 2000;
- Roma, Marzo 2002;
- Palermo, Febbraio 2003;
- Ravello (Salerno), Settembre 2003.

≫ La sottoscritta ha presentato l'attività di ricerca nelle conferenze nazionali e internazionali nelle quali sono stati presentati i lavori [dna00,ictcs98,words99,words01,ictcs10].

≫ Nell'ambito del progetto ESF AutoMathA, la sottoscritta ha svolto un seminario "Circular splicing languages" su invito per l'incontro ESF Science Meeting "Automata and formal languages for DNA computation and bioinformatics" (Como , October 18-20, 2006).

4.5 Attività di revisione

≫ Attività di revisione di articoli per varie edizioni delle seguenti conferenze nazionali e internazionali: ICTCS, DLT, MFCS, STACS, DNA, Words, ICTAC, Fun with Algorithms, FCT,CPM.

≫ Attività di revisione per le riviste internazionali: RAIRO-Theoretical Informatics and Applications, Theoretical Computer Science, Journal of Computer and Science Technology, Soft Computing, International Journal of Automata Languages and Combinatorics, Fundamenta Informaticae.

4.6 Attività organizzativa

La sottoscritta ha fatto parte del comitato organizzativo dei seguenti eventi:

≫ Incontro del Progetto PRIN 2001-2003, Ravello (Salerno), 19-21 Settembre 2003

(http://www.dia.unisa.it/conferences/cofin/).

≫ Advances on two-dimensional language theory (W2DL), European Science Foundation Science Meeting, Salerno, 3-5 Maggio 2006

(http://www.dia.unisa.it/conferences/W2DL/).

≫ WORDS 2009, 7th International Conference on Words, University of Salerno, 14-18 September, 2009

(http://words2009.dia.unisa.it/).

4.7 Partecipazione a progetti di ricerca

◇ **Progetti nazionali (PRIN)**

"Modelli di calcolo innovativi: Metodi sintattici e combinatori - Unconventional Computational Models: Syntactic and Combinatorial Methods" (MURST 1998-1999).

"Linguaggi formali ed automi: teoria ed applicazioni - Formal Languages and Automata: theory and applications" (MIUR 2001-2003).

"Linguaggi Formali ed Automi: Metodi, Modelli e Applicazioni - Formal Languages and Automata: Methods, Models and applications" (MIUR 2003-2005).

"Automi e Linguaggi Formali: aspetti matematici e applicativi - Automata and Formal languages: mathematical and application driven studies" (MIUR 2005-2007)

"Mathematical aspects and emerging applications of automata and formal languages" (MIUR 2007-2009)

◇ **Progetti 60%, FARB**

"Linguaggi Formali e Modelli di Calcolo" (Università di Salerno, 2001).

"Linguaggi Formali e Codici: Problemi classici e Modelli innovativi" (Università di Salerno, 2003).

"Linguaggi Formali e Codici: Modelli e caratterizzazioni strutturali" (Università di Salerno, 2004).

"Linguaggi Formali e Codici: Problemi classici e Modelli innovativi" (Università di Salerno, 2005).

"Linguaggi formali e codici a lunghezza variabile: proprietà strutturali e nuovi modelli di rappresentazione" (Università di Salerno, 2006).

"Proprietà strutturali e nuovi modelli di rappresentazione nella teoria dei linguaggi formali" (Università di Salerno, 2007).

"Estensioni della teoria dei linguaggi formali e loro proprietà strutturali" (Università di Salerno, 2008).

"Automi e Linguaggi Formali: aspetti emergenti e fondazionali" (Università di Salerno, FARB Project, 2009-2011).

"Aspetti emergenti e fondazionali nella teoria degli automi e dei linguaggi formali" (Università di Salerno, FARB Project, 2010-2012).

◇ **Progetto comune (italo-francese) CNR-CNRS 2000-2001**

"Linguaggi Formali e Dinamica Simbolica: metodi combinatori - Methodes combinatoires dans les langages formels et la dynamique symbolique".

Area della cooperazione: Informatica teorica - Scienze Matematiche.
Responsabile italiano della ricerca: C. De Felice (Univ. di Salerno).
Responsabile francese della ricerca: J. Berstel (Istitut G. Monge, Univ. Marne-la-Vallee).

◇ **Progetto comune (italo-francese) CNR-CNRS 2002-2003**

"Teoria degli automi - Theorie des automates".

Area della cooperazione: Informatica teorica - Scienze Matematiche.
Responsabile italiano della ricerca: C. De Felice (Univ. di Salerno).
Responsabile francese della ricerca: C. Choffrut (LIAFA, Univ. Paris VII).

◇ **Progetto europeo**

"MolCoNet: A Thematic Network on Molecular Computing" - Progetto europeo nell'ambito del V programma quadro (IST-2001-32008). Responsabile scientifico: Prof. G. Mauri. Durata del progetto: 1/ 12/ 2001 - 30/ 11/ 2004.

Co-responsabile della redazione del RoadMap Report annuale.

5 Lista delle pubblicazioni

Tesi di dottorato

Titolo: On the power of classes of splicing systems.
Relatori: Prof. Giancarlo Mauri (Univ. di Milano-Bicocca),
Prof.ssa Clelia De Felice (Univ. di Salerno).
Referees esterni: Prof. Antonio Restivo (Univ. di Palermo),
Prof. H.J. Hoogeboom (Univ. di Leiden, Olanda).
Sede amministrativa del dottorato: Università Statale di Milano.
Data esame finale: 11.01.2002.

Abstract pubblicato sul Bollettino dell'EATCS n.78 (Ottobre 2002).

Scholarperdia web page on Splicing systems

[scholar2010] R. Z. (2010), *Splicing systems*, Scholarpedia, 5(7):9397. Category: Molecular Computing

5.1 Riviste e atti di conferenze

- **Riviste internazionali**

[*ipl*01] P. Bonizzoni, C. Ferretti, G. Mauri, R. Zizza (2001), *Separating some splicing models*, Information Processing Letters, 79/6, pp. 255-259.

[*ita*04] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2004), *Circular splicing and regularity*, Theoretical Informatics and Applications 38, pp. 189-228.

[*tcs*05] P. Bonizzoni, C. De Felice, R. Zizza (2005), *The structure of reflexive regular splicing languages via Schützenberger constants*, Theoretical Computer Science, Vol 334, No. 1-3, pp.71-98.

[*dam*05] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2005), *On the power of circular splicing*, Discrete Applied Mathematics, 150, pp. 51-66.

[*dam*06] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2006), *Linear splicing and syntactic monoid*, Discrete Applied Mathematics, 154, pp. 452-470.

[*tcs*09] C. De Felice, G. Fici, R.Z. (2009), *A characterization of regular circular languages generated by marked splicing systems*, Theoretical Computer Science, 410, pp. 4937-4960.

[*survey*09] P. Bonizzoni, C. De Felice, G. Fici, R.Z. (2009), *On regularity of circular splicing languages: a survey and new developments*, Natural Computing, Natural Computing, Volume 9, Issue 2, pp. 397-420.

[*tcs*10] P. Bonizzoni, C. De Felice, R.Z. (2010), *A characterization of (regular) circular languages generated by monotone complete splicing systems*, Theoretical Computer Science, 411(48), pp. 4149-4161.

- **Atti di convegni internazionali**

[*dna*00] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2001), *DNA and circular splicing*, in "DNA Computing", Sixth International Workshop on DNA based computers, DNA2000 (Leiden, Olanda, 13-17 Giugno, 2000), Lecture Notes in Computer Science 2054, A. Condon and G. Rozenberg (Eds.), Springer-Verlag, pp. 117-129.

[*dlt*02] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2003), *Decision Problems for Linear and Circular Splicing Systems*, (invited talk G. Mauri), in "Developments in Language Theory", Sixth International Conference DLT2002 (Kyoto, Giappone, 18-21 Settembre 2002), Lecture Notes in Computer Science 2450, M. Ito and M. Toyama (Eds.), Springer-Verlag, pp. 78-92.

[*dlt*03] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2003), *Regular Languages Generated by Reflexive Finite Splicing Systems*, in "Developments in Language Theory", Seventh International Conference DLT2003 (Szeged, Hungary, 7-11 Luglio, 2003), Lecture Notes in Computer Science 2710, Z. Esik and Z. Fulop (Eds.), Springer-Verlag, pp. 134-145.

[fct07] C. De Felice, G. Fici, R. Zizza (2007), *Marked Systems and Circular Splicing*, in "Foundamentals of Computational Theory", 16th International Symposium on FCT07 (Budapest, Hungary, 27-30 Agosto, 2007), Lecture Notes in Computer Science 4639, E.Csuhaj-Varju, Z. Esik (Eds.) Springer-Verlag, Berlin, pp. 238-249.

[*dcm*09] P. Bonizzoni, C. De Felice, R.Z. (2009), *Circular languages generated by complete splicing systems and pure unitary languages*, in "DCM 2009", 5th International workshop on Developments in Computational Models (11 July, Rhodes, Greece), satellite workshop of ICALP 2009, download of the preproceedings. Electronic Proceedings in Theoretical Computer Science, S. Barry Cooper, Vincent Danos Eds., vol. 9, p. 22-31, ArXiv http://arxiv.org/abs/0911.2320v1.

- **Atti di convegni nazionali**

[*ictcs*98] C. De Felice, R. Zizza (1998), *Factorizing codes and Krasner factorizations*, in "Sixth Italian Conference on Theoretical Computer Science" (Prato, Italia, 9-11 Novembre 1998), P. Degano, U. Vaccaro, G. Pirillo (Eds.), World Scientific, pp. 347-358.

- **Atti di convegni internazionali (pubblicati da Università)**

  [*gs*00] P. Bonizzoni, C. Ferretti, G. Mauri, R. Zizza (2000), *Separating some splicing models*, Proceedings of the "International Workshop on Grammar Systems 2000" (Bad Ischl, Austria, 3-7 Luglio 2000), R. Freund, A. Kelemenova (Eds.), Silesian University at Opava, ISBN 80-7248-067-7, pp. 55-60.

  [*words*03] S. Verlan, R. Zizza (2003), *1-splicing vs. 2-splicing: separating results*, Proceedings of "WORDS'03", 4^{th} International Conference on Combinatorics on Words (10-13 Settembre 2003, Turku, Finlandia), T. Harju and J. Karhumaki (Eds.), TUCS General Publication (Turku Centre for Computer Science), No. 27, August 2003, ISBN 952-12-1211-X, pp. 320-331.

- **Rapporti interni**

  [*ri*00] P. Bonizzoni, R. Zizza (2000), *A characterization of regular splicing languages*, Rapporto Interno n. 254-00, del Dip. di Scienze dell'Informazione, Università degli Studi di Milano, 15 pagine.

5.2 Comunicazioni a Workshops Internazionali (senza pubblicazione di atti)

[*words*99] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (1999), *Linear and circular splicing*, comunicato a "WORDS99" (Rouen, Francia, 20-25 Settembre 1999).

[*words*01] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2001), *Developments on circular splicing*, comunicato a "WORDS2001" (Palermo, Italia, 17-21 Settembre 2001).

[*ams*02] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2002), *Splicing systems and automata theory* (invited talk C. De Felice), First Joint Meeting AMS-UMI (Pisa, Italia, 12-16 Giugno 2002).

[*ama*07] C. De Felice, G. Fici, R. Zizza (2007), *Marked Systems and Circular Splicing*, comunicato a "AutoMatha 2007", Palermo, Italy, 18-22 Giugno 2007.

[*ama*09] P. Bonizzoni, C. De Felice, R.Z. (2009), *On circular semi-simple splicing systems*, communicated at "AutoMathA 2009" (8-12 June, Liege, Belgium).

[*ictcs*2010] L. D'Auria, R.Z. (2010), *A note on the McCloskey's algorithm for deciding whether a regular language is a code*, communicated at "ICTCS 2010" (Camerino, Italy, September 15-17, 2010).

6 Descrizione dell'attività di ricerca

L'attività di ricerca della sottoscritta ha riguardato essenzialmente alcuni problemi della Teoria dei Codici a lunghezza variabile e alcuni modelli teorici per il Calcolo Molecolare.

Teoria dei Codici

I codici a lunghezza variabile sono insiemi di parole che permettono un'unica decodifica del messaggio sorgente. La teoria algebrica dei codici è stata sviluppata da M.-P. Schützenberger e dalla sua scuola, circa 40 anni fa. Essa trova una naturale collocazione nell'area dei Linguaggi Formali e della Combinatoria delle parole. Una delle principali questioni aperte è la Congettura della Fattorizzazione, proposta da Schützenberger, ossia l'esistenza di una fattorizzazione non ambigua $A^* = SC^*P$ del monoide libero, con C codice (finito, massimale) e P, S polinomi finiti. Tale congettura è legata anche all'ottimalità dei codici prefissi nella trasmissione. Difatti, se la Congettura della Fattorizzazione fosse vera, allora la classe dei codici prefissi sarebbe quella a minimo costo di trasmissione, anche nel caso più generale rispetto a quello mostrato dalla famosa disuguaglianza di Kraft-McMillan, ossia con costi delle lettere dell'alfabeto non uniformi. Risultati parziali (rispetto alla congettura di Schützenberger) sono noti ed evidenziano anche relazioni tra codici che fattorizzano il monoide libero e le fattorizzazioni dei gruppi ciclici. La sottoscritta ha iniziato ad affrontare lo studio della congettura durante la Tesi di Laurea (relatore Prof.ssa C. De Felice). In [ictcs98] sono stati caratterizzati i polinomi P tali che $A^* = SC^*P$ quando S è un polinomio di cardinalità tre e inoltre sono state confermate connessioni con le fattorizzazioni di Krasner.

Decidere se un linguaggio X su un alfabeto finito è un codice può essere fatto in diversi modi. Il test per la codicità risale a Sardinas e Patterson (1953) e vari algoritmi efficienti sono stati progettati quando X è un insieme finito, usando differenti approcci. Un ben noto algoritmo è stato progettato se X è un linguaggio regolare, dato attraverso un automa non ambiguo (costruito a partire dal flower automaton e dallo square automaton). Purtroppo, nel caso peggiore, non è possibile evitare l'esplosione del numero degli stati, dovuta alla necessità di avere un automa non ambiguo in input all'algoritmo e quindi di doverlo disambiguare. Nel 1996 McCloskey disegnò un algoritmo che dato in input un automa a stati finiti $\mathcal{A}$, anche ambiguo, decide la codicità del linguaggio accettato da $\mathcal{A}$. Il cuore della procedura è la nozione di automa ristretto $\mathcal{A}_\mathcal{R}$, che viene costruito a partire da $\mathcal{A}$ ed è equivalente ad $\mathcal{A}$. Comunque la prova dell'equivalenza tra $\mathcal{A}$ e $\mathcal{A}_\mathcal{R}$ non è data e la costruzione di $\mathcal{A}_\mathcal{R}$ presenta una piccola mancanza. In [ictcs10] viene fornita la procedura completa per costruire l'automa ristretto, mantenendo la complessità in tempo $O(n^2)$, dove n è la taglia di $\mathcal{A}$. L'algoritmo è stato efficientemente implementato in Java 6.0.

Calcolo Molecolare e Linguaggi Formali

L'argomento della Tesi di Dottorato, anche attuale area di ricerca della sottoscritta, è lo studio di modelli formali per il Calcolo Molecolare con l'utilizzo di strumenti della Teoria dei Linguaggi Formali. Difatti, tale teoria ha recentemente ricevuto nuova linfa vitale anche in seguito all'introduzione di nuovi modelli di computazione. In particolare, è stata proposta la formalizzazione di un modello matematico per fenomeni biologici di ricombinazione molecolare. Sebbene questo possa inizialmente sorprendere, l'analogia tra processi biologici e matematici è alquanto evidente: in entrambi, le strutture biologiche complesse o

l'applicazione di funzioni calcolabili ad un argomento sono il risultato della combinazione di operazioni semplici a oggetti-entità di base. Questa analogia è la chiave di interpretazione del *Calcolo Molecolare*.

Il Calcolo Molecolare ha avuto forte risonanza in seguito ad un esperimento condotto da Adleman (1994) che, codificando in sequenze di DNA un'istanza di un noto problema NP-completo (Hamiltonian Path), ne ha proposto una soluzione molecolare "efficiente", lavorando con enzimi e reazioni chimiche. Tra esse, un ruolo centrale è ricoperto dalla ibridazione, meccanismo per il quale due filamenti di DNA (sequenze delle quattro basi Adenina, Guanina, Citosina, Timina) si uniscono se sono Watson-Crick complementari, ossia se le basi in corrispondenza sono coppie complementari: Adenina-Timina; Citosina-Guanina. In tal modo si forma il doppio filamento (double-strand) che assume la nota forma a doppia elica. Sono stati sviluppati numerosi algoritmi molecolari, con l'obiettivo di riuscire a costruire un calcolatore molecolare, sfruttando l'estremo parallelismo delle operazioni molecolari e l'efficienza in spazio di tali meccanismi. Notevoli sono, però , i punti deboli della tecnica di Adleman, che rendono difficile la sua reale fattibilità , tra cui il peso della soluzione chimica necessaria per la soluzione del problema e la presenza di errori. Questi ultimi possono essere causati dalla possibilità di ibridazioni che avvengono anche senza un match perfetto (in termini di complementarietà Watson-Crick) tra le coppie di basi dei due filamenti, determinando falsi positivi e falsi negativi.

Precedentemente già Head (1987) aveva evidenziato un collegamento tra Biologia e Calcolo Molecolare attraverso la Teoria dei Linguaggi Formali, proponendo un meccanismo di generazione di parole (*splicing*), come modello del comportamento ricombinante delle molecole di DNA sotto l'azione di enzimi. Lo splicing è un'operazione di cut and paste biologico operato su molecole di DNA da enzimi di restrizione e di ligasi. Gli enzimi di restrizione sono proteine capaci di riconoscere un fissato pattern all'interno di una molecola di DNA e di tagliare tale molecola in un punto specifico all'interno del pattern, spezzando così la molecola originaria in due frammenti. Siccome la molecola di DNA è double-stranded, tale taglio può essere netto o tale da lasciare sporgenze di diversa lunghezza tra i due filamenti di DNA (blunt o sticky ends). Gli enzimi di ligasi attaccano due frammenti di DNA, se questi presentano un'opportuna struttura chimica. Pertanto la presenza di tali enzimi in una soluzione acquea con DNA, provoca la generazione di nuove molecole ottenute dalla fase di taglio (restrizione) e dalla fase di incollaggio (ligasi).

Sono stati poi introdotti gli splicing system come modello generativo di parole. In particolare, oggetto di ricerca sono gli splicing system iterati, sistemi formali che alle sequenze di DNA fanno corrispondere un insieme I (iniziale) di parole su un alfabeto finito e agli enzimi fanno corrispondere un insieme di parole speciali R (regole) in cui è contenuta la specifica sia dei due patterns da riconoscere, sia del punto del taglio, sia della ricomposizione dei pezzi tagliati. Dunque, a partire dall'insieme iniziale, per l'applicazione iterata delle regole, si genera un linguaggio, detto linguaggio splicing generato da I e R. In questo modello si suppone di disporre di un numero non limitato di copie di ogni parola. Esistono tre definizioni di operazione splicing (proposte da Head, Paun, Pixton) e numerose varianti di sistemi splicing, che differiscono tra loro per la diversa maniera di utilizzare le regole e di definire il linguaggio splicing generato. Ogni nuova variante di splicing system presentato in letteratura è accompagnata dallo studio delle sue capacità computazionali, cioè dalla caratterizzazione della classe dei linguaggi che da esso vengono generati. Spesso si è provata l'universalità di tali modelli (equivalenza con la Macchina di Turing). Anche per i modelli di calcolo molecolare, la finalità non è porre in discussione la Tesi di Church-Turing, ma

progettare un sistema di calcolo che, in maniera più efficiente rispetto a quelli convenzionali, esegua computazioni difficili. Al variare di I, R nella gerarchia di Chomsky, si ottengono sistemi splicing che generano una classe della gerarchia o una classe intermedia tra due di esse e, in tal caso, si pone il problema di caratterizzare la classe generata. Ad esempio, utilizzando un insieme regolare di regole e un linguaggio iniziale finito, si ottengono già alcuni linguaggi ricorsivamente enumerabili.

Nello studio dello splicing lineare, l'attività di ricerca della sottoscritta si è ristretta allo studio della struttura dei linguaggi che possono essere prodotti da uno splicing system con I e R insiemi finiti (splicing system finiti). È noto che in questo caso i sistemi splicing generano una sottoclasse propria di linguaggi regolari. Head aveva posto il problema, tuttora aperto, se fosse decidibile stabilire l'appartenenza di un linguaggio regolare a tale sottoclasse e, in caso di risposta affermativa, dare una caratterizzazione di questa famiglia. In tal modo si otterrebbe un sistema che effettivamente riproduce un automa con materiale biologico (perché gli elementi che intervengono sono in numero finito), almeno per alcune classi di linguaggi regolari. La letteratura presenta diversi tentativi in tal senso, che hanno portato anche a caratterizzare, ad esempio, la classe dei linguaggi Strictly Locally Testable in termini di splicing finito.

In [dam06] si è affrontato tale problema, dando uno splicing system finito (di Paun) per generare alcuni linguaggi infiniti (regolari). Le regole hanno il compito di "allungare" le parole del linguaggio iniziale e quelle via via ottenute, simulando così l'attraversamento illimitato dei cammini chiusi (cicli) dell'automa. Tale sottoclasse dei linguaggi regolari è definita mediante una condizione sulle classi del monoide sintattico. Questa famiglia ha suggerito la caratterizzazione della classe di linguaggi generabili da splicing lineare finito (di Paun) riflessivo, ossia sistemi in cui l'insieme delle regole soddisfa una ipotesi di riflessività [tcs05,dlt03]. La nozione di costante, introdotta da Schützenberger (1975), interviene in modo determinante. La caratterizzazione su menzionata è stata ottenuta anche per i sistemi finiti di Pixton. Come conseguenza sono stati caratterizzati i linguaggi regolari generati dai sistemi finiti di Head [dlt03]. L'obiettivo resta la caratterizzazione dei linguaggi regolari finitamente generati dagli altri sistemi e un corrispondente problema di complessità descrizionale: se un linguaggio regolare è generabile mediante un sistema splicing finito, fornire una limitazione alla lunghezza delle regole (o sulla cardinalità di R) di un sistema che lo genera. Studi preliminari del problema sono contenuti in [words99].

Una naturale domanda è il confronto tra le tre varianti della definizione di splicing lineare date da Head, Paun e Pixton. Nel caso finito si è dimostrato che la classe di linguaggi generabili da splicing finito usando la definizione di Pixton (risp. Paun) contiene propriamente la classe di linguaggi generati da splicing finito usando la definizione di Paun (risp. Head) [ipl01,gs00].

L'operazione di splicing lineare applicata a due parole può generare una sola parola (1-splicing) o due parole (2-splicing). Quest'ultimo caso è equivalente a considerare sistemi in cui l'operazione è 1-splicing e l'insieme delle regole gode della proprietà simmetrica. Una questione naturale è lo studio del confronto tra il potere generativo dei sistemi in cui l'operazione è 1-splicing, detti H_1 splicing systems, e quelli in cui l'operazione è 2-splicing, detti H_2 splicing systems. Nel caso finito, si è provando che questi ultimi generano una classe di linguaggi regolari propriamente contenuta nella classe di linguaggi generati dagli H_1 splicing systems [words03]. Primi approcci al potere generativo dello splicing sono in [ri00].

Un altro aspetto della problematica dei sistemi splicing fa intervenire la relazione di

coniugazione, una relazione di equivalenza ben nota nella combinatoria delle parole e che interviene anche in problemi di fattorizzazione di linguaggi. Le classi di equivalenza rispetto a tale relazione di coniugazione sono anche chiamate parole circolari. Infatti, due parole coniugate sono due parole che differiscono per una permutazione ciclica delle lettere e quindi sono identificabili se disponiamo le parole su un cerchio. Per estensione si possono definire i linguaggi circolari. Anche il concetto di regolarità può essere introdotto e i linguaggi circolari regolari corrispondono ai linguaggi formali regolari chiusi rispetto alla relazione di coniugazione. In letteratura è stato formalizzato un fenomeno di cut and paste biologico operato dagli enzimi su DNA circolare (come i plasmidi o, in alcuni casi, i batteri). Questo ha stimolato l'interesse verso lo studio dello splicing su parole circolari. Come per il caso lineare, sono note tre definizioni di splicing circolare (date da Head, Paun, Pixton) e sul potere computazionale degli associati sistemi splicing la letteratura esistente è scarna. La difficoltà intrinseca che si presenta nel maneggiare classi di equivalenza (di una relazione che non è una congruenza) piuttosto che parole del monoide libero, probabilmente ne è la ragione.

Il meccanismo biologico alla base della definizione di Head e Paun consiste nel riconoscimento di due patterns in due parole circolari, nel taglio (apertura) delle parole all'interno dei patterns riconosciuti, nella concatenazione delle due stringhe lineari così ottenute e nella ricircolarizzazione della parola risultante. Nella definizione data da Pixton invece, l'ultima operazione è preceduta da una sostituzione dei patterns individuati.

Abbiamo affrontato per il caso dello splicing circolare il problema analogo a quello visto per il lineare, ossia caratterizzare linguaggi circolari regolari finitamente generati. Sempre ponendoci nelle ipotesi di finitezza del linguaggio (circolare) iniziale e dell'insieme delle regole (sistemi splicing circolari finiti), abbiamo confrontato le tre definizioni, provando un'inclusione tra le classi di linguaggi generati dai modelli descritti da Head e Paun. Abbiamo provato che esistono linguaggi circolari regolari che non sono generati da sistemi splicing circolari finiti [dam05,words01]. Se si escludono le ipotesi aggiuntive e le varianti introdotte in letteratura per definire opportuni sistemi splicing circolari universali, i sistemi splicing circolari finiti generano sia linguaggi circolari regolari sia linguaggi circolari context-free non regolari. Non è nota ancora la loro capacità computazionale. Dunque, con l'obiettivo di caratterizzare i linguaggi circolari regolari generati da splicing circolare finito (di Paun), abbiamo individuato classi di linguaggi regolari la cui circolarizzazione è finitamente generata. Esempi significativi di tali classi sono i linguaggi X^* chiusi rispetto alla relazione di coniugazione e con X codice a gruppo regolare oppure con X insieme finito. Le prove di questi risultati sono basate su tecniche di combinatoria delle parole e automi [ita04,dam05,dna00].

Recentemente sono stati studiati i sistemi marked, particolari sistemi circolari di Paun finiti semi-semplici che soddisfano alcune ipotesi aggiuntive [ama07,fct07,tcs09]. I sistemi semi-semplici, introdotti da Ceterchi, Martin-Vide e Subramanian (2004), sono sistemi splicing circolari finiti in cui per ogni regola $u_1\#u_2\$u_3\#u_4$ si ha che $u_1u_2, u_3u_4 \in A$, ossia i siti sono lettere dell'alfabeto. Se $u_1u_2 = u_3u_4 \in A$ allora si tratta di sistemi semplici. La posizione delle lettere non è ininfluente. La simmetria implicita dello splicing circolare determina tre tipologie di regole, a seconda della posizione della lettera nella regola, cioè a seconda dell'$u_i \in A$. Il problema di caratterizzare il potere computazionale di tali sistemi viene semplificato se si impone che le regole abbiano tutte la stessa forma, ossia per tutte le regole $u_i\#u_j\$u_k\#u_l$ si abbia $u_i, u_k \in A$ (sistemi (1,3)-CSSH) oppure $u_j, u_l \in A$ (sistemi (2,4)-CSSH) oppure $u_j, u_k \in A$ (sistemi (2,3)-CSSH). E' noto che la classe dei linguaggi

generati da (1,3)-CSSH systems non è confrontabile con la classe dei linguaggi generati da (2,4)-CSSH systems. In [tcs09] viene provato che la classe dei linguaggi generati da (2,4)-CSSH systems è la classe del reversal dei linguaggi circolari generati da (1,3)-CSSH systems. Quindi i sistemi marked sono (1,3)-CSSH in cui l'alfabeto è il linguaggio iniziale che coincide con i siti delle regole. Si è caratterizzata per alcuni di essi la struttura dei linguaggi generati, mostrando che è sufficiente caratterizzarla per i sistemi marked transitivi, ossia, sistemi marked in cui I è una classe di equivalenza rispetto ad una precisa relazione di equivalenza, introdotta in [ama07,fct07,tcs09]. I linguaggi regolari generati da sistemi marked sono caratterizzati da una proprietà definita attraverso due nozioni: distanza tra due lettere e diametro $d(S)$ di un sistema S. Queste consentono di classificare sistemi marked transitivi S che generano linguaggi regolari circolari L in termini del diametro: se $d(S) < 3$ allora L è regolare, se $d(S) > 3$ allora L non è regolare. La classe dei sistemi marked transitivi S tali che $d(S) = 3$ è un confine tra la famiglia dei linguaggi regolari e il suo complemento. Infatti in [fct07,tcs09] si forniscono esempi di sistemi S con $d(S) = 3$ e tali che i linguaggi generati corrispondenti sono regolari ma anche altri non regolari. I risultati appena descritti sono stati ottenuti anche se si considerano (2,4)-CSSH systems piuttosto che (1,3)-CSSH systems. In [survey09] i risultati sono rivisti anche in un ambito di teoria dei grafi.

In [ama07, fct07, tcs09], il potere computazionale dei sistemi marked è stato completamente descritto quando si aggiunge l'operazione di self-splicing, nota in letteratura, dimostrando che si generano solo linguaggi circolari regolari.

In [tcs10] viene risolto il problema di trovare una caratterizzazione della classe dei linguaggi circolari generati da sistemi splicing circolari per i sistemi monotoni completi, ossia sistemi semi-semplici in cui esiste una regola per ogni coppia di simboli dell'alfabeto. Viene provato che un linguaggio circolare L è generato da un sistema monotono completo se e solo se l'insieme $Lin(L)$ di tutte le parole che corrispondono a L è un linguaggio pure unitary generato da un insieme chiuso rispetto alla relazione di coniugazione. La classe dei linguaggi pure unitary è stata introdotta da A. Ehrenfeucht, D. Haussler, G. Rozenberg nel 1983, come una sottoclasse della classe dei linguaggi context-free, insieme con una caratterizzazione dei linguaggi regolari pure unitary attraverso una proprietà decidibile. Come diretta conseguenza, sono stati caratterizzati i linguaggi circolari regolari generati da sistemi monotoni completi. Viene anche provato che è decidibile stabilire se il linguaggio generato da tali sistemi sia regolare.

Riguardo allo studio del potere computazionale dei sistemi semplici, in [dcm09,tcs10] viene provato che i sistemi monotoni completi hanno lo stesso potere computazionale dei sistemi semplici, quando solo una regola è consentita. Dai risultati sui sistemi completi, segue che i sistemi semplici finiti generano una classe di linguaggi context-free che contiene linguaggi non regolari, in contraddizione con un risultato noto in letteratura in cui si affermava che tali sistemi generano esclusivamente linguaggi regolari (un controesempio era stato fornito già in [survey09]).

È stata data una caratterizzazione dei linguaggi unari generabili con sistemi circolari finiti di Paun che fa intervenire i sottogruppi di gruppi ciclici finiti [ita04]. Abbiamo anche provato che è decidibile stabilire se un linguaggio regolare unario è generabile da splicing finito circolare (di Paun) [dam05,words01].

La sottoscritta è curatore della pagina relativa agli Splicing Systems per Scholarpedia [scholar2010].

Lo studio dello splicing circolare ha validità anche da un punto di vista applicativo,

dato che molto recentemente i plasmidi (molecole di DNA circolare) hanno sostituito il DNA lineare per simulare algoritmi in provetta. Dunque, le problematiche per ricerche future ispirate da quest'area si differenziano per ambiti e grado di difficoltà . Tra i problemi più complessi relativi ai sistemi splicing resta aperto quello della descrizione del potere computazionale dei sistemi circolari splicing finiti, mentre quello della caratterizzazione dei linguaggi regolari chiusi sotto la relazione di coniugazione appartiene più in generale al contesto della Teoria dei Linguaggi Formali. Alcune questioni vengono affrontate in [ita04] e altre, proposte in [words01], vengono risolte in [dam05]. Una survey sullo stato dell'arte e sui più recenti sviluppi è presentata in [dlt02,survey09].

Implementazione di algoritmi per DNA computing affidabile

L'ibridazione è il processo chiave nell'approccio di Adleman, descritto all'inizio della sezione precedente. Purtroppo, questa stessa operazione può compromettere i risultati di un DNA program se la codifica non è scelta in maniera opportuna. L'informazione potrebbe anche essere persa, perchè resa inutilizzabile, se le codifiche scelte per l'input si legano (ibridizzano) tra loro o con se stesse. Inoltre, potrebbero crearsi molecole di DNA derivate da un errato legame di due strati di DNA tra loro non perfettamente complementari (ibridazioni non volute). Infatti, il presupposto che il DNA sia error-free non è corretto. Da qui la necessità di studiare codifiche opportune che limitino falsi negativi, e una nuova teoria di prevenzione e correzione dell'errore (in letteratura noto come il DNA codewords design problem).

Nel tentativo di rispondere alla presenza di errori nell'approccio di Adleman, in letteratura sono state proposte varie strategie. In alcuni casi, una minima distanza di Hamming viene proposta come criterio necessario per una attendibile computazione basata su DNA. Tale misura non è adeguata se si considerano anche possibili shift dei nucleotidi (che difatto ibridizzano anche se sono shiftati) o l'ibridazione di un nucleotide con se stesso (hairpin).

Molte proprietà vengono descritte come "forbidden bonds". Recentemente, Kari et al. (2004) hanno definito bond-free languages, Θ-freedomness, Θ-compliance (dove Θ rappresenta il morfismo involutivo della complementarietà) ed altre proprietà che generalizzano le condizioni per progettare "buone" DNA codewords.

I codici a lunghezza variabile sono necessari in alcuni ambiti, in cui non sono sufficienti i codici a lunghezza fissa. Sono stati progettati vari applicativi, tra cui CodeGen (Kephart, Lefevre 2004) che, dato in input un linguaggio, decidono innanzitutto la codicità , e poi alcune proprietà come Θ-compliance (i.e., nessuna parola può ibridizzare con un fattore di un'altra parola del codice) o Θ-freedomness (i.e., nessuna parola ibridizza con la concatenazione di due altre parole di codice). Se infatti tali proprietà non si verificassero, le molecole sarebbero inutilizzabili per la computazione.

In questo ambito, recentemente, si sta lavorando sulla possibilità di implementare efficientemente in Java il pacchetto CodeGen, scritto originariamente in Visual Basic, tenendo fortemente presente la complessità in spazio delle strutture dati utilizzate. Inoltre si sta lavorando sulla possibilità di integrare la tecnica di McCloskey per decidere l'univoca decifrabilità di un insieme regolare di parole con le proprietà che determinano la bontà di un codice per il DNA Computing. Si tratta di trovare una relazione tra queste proprietà e i cammini nell'automa ristretto di McCloskey proprio come viene fatto per CodeGen, che utilizza invece la tecnica classica del flower automaton e dello square automaton, per decidere la codicità . Infine si stanno integrando le summenzionate proprietà con altre proprietà (e.g., absent words: dato un testo, cercare tutte le parole più corte che non compaiono come fattori nel testo), usando anche l'approccio più classico di algoritmi su stringhe.