CV SUMMARY – MATTEO PRADELLA

DATE OF BIRTH MAY 31, 1971

EDUCATION

2001 **PhD Degree** in Computer Science, Politecnico di Milano (Chorafas Prize).
1996 **Master of Science Degree** in Computer Engineering, Politecnico di Milano.

CAREER

2011 – present **Associate Professor**, DEIB Politecnico di Milano
2001 – 2011 **Tenured Researcher**, IEIIT CNR, Milano
2001, 2002 **Visiting Researcher** at Naval Research Laboratory, Software Engineering group (code 5546), Washington, DC.
2001 **Post-Doc Researcher** (Assegnista di Ricerca), Politecnico di Milano, Grant by MIUR.
2000 **Visiting PhD student** at Naval Research Laboratory, Software Engineering group (code 5546), Washington, DC.
1998 – 2001 **Ph.D. Student** in Computer Science, Politecnico di Milano
1996 – 1997 **Developer**, working on the TRIO temporal logic semantic tools, in a joint project between Politecnico di Milano and CESI.

LEAVE PERIODS

June 1997 – April 1998 **Military Service**

HABILITATION

Type of habilitation	Country	SSD (if Italian habilitation) or topic area	Date of achievement
Full Professor	Italy	ING-INF/05	August 3, 2017

RESEARCH INTERESTS

- **Formal Methods for Software Engineering:** Software Verification, Temporal Logic, Model Checking.
- **Formal Languages and Compilers:** Operator Precedence and Input Driven Languages, Parallel Parsing, Context-Oriented Programming Languages, 2D Languages.

LEADERSHIP IN COMPETITIVE RESEARCH PROJECTS

Project Name	**Time Period**	**Funding Institution**	**Funding Scheme**	**Role of the applicant**	**Budget for the applicant's institution**
2D Grammars for Picture description	2008	CNR	CNR RSTL program	PI	33K€ (99K€ original program)
RSTL was a competitive national research program introduced by CNR for basic research: of the 800 projects presented only 311 were financed (39%). My original project was for a 3 years span with a total net contribution of 99K€. Unfortunately, after 2008 the RSTL program was canceled, and only the first year of the accepted projects was covered.					
D-ASAP	2007-8	MIUR	PRIN	Leader of the CNR unit of Milano	120K€ for CNR
AUTOVAM: Automatic Verification: Ad Maiora.	2018-2019	Fondazione Cariplo	Bando congiunto Regione Lombardia - Fondazione Cariplo	Co-PI	125K€
Programming trustworthy Infrastructure As Code in a sEcuRE framework (PIACERE)	12/2020-11/2023	EU Commission	H2020	Leader of Verification Tools	592K€

LEADERSHIP IN INDUSTRIAL RESEARCH PROJECTS

Project Name	**Time Period**	**Funding Company**	**Role of the applicant**	**Budget for the applicant's institution**
Development of R++	2016-2017	Robox SpA	PI	50K€

SCIENTIFIC PRODUCTION AND METRICS

- **Scientific Productivity: 97** publications (**83** entries on Scopus, **47** co-authors according to Scopus):
  - Author/Co-author of **15** top ranked journal papers (including **SIAM Journal on Computing, ACM Transaction on SW Engineering, IEEE Transactions on Software Engineering, Theoretical Computer Science, Pattern recognition**).
  - Author/Co-author of **60** scientific publications on peer-reviewed conferences including **4** top-level **CORE A*** conferences (**ICSE, ESEC/FSE, CAV**).
- **Publication Impact:** Based on Google Scholar: h-index **24** citations **1779**
  Based on Scopus: h-index **17** citations **904**

AWARDS AND RECOGNITIONS

2015 IBM Faculty Award for the PAPAGENO parallel parser generator (proposer: D. Mandrioli).

2012 Google Faculty Award for the PAPAGENO parallel parser generator (proposer: S. Crespi Reghizzi).

2007 Formal Methods Europe (FME) grant for the model checker *Zot*.

2001 Chorafas Foundation prize

INVITED TALKS AND SEMINARS (SELECTION OF PAST 10-YEARS)

Invited speaker @QuantLA (Quantitative Logics and Automata), Universitat Leipzig, 2016. Title: *Parallel parsing of Operator Precedence languages*.

TEACHING EXPERIENCE (SELECTION OF PAST 5-YEARS)

Institution name	Course name	Credits	No. of students	Reference Study Course	Time period	Students Evaluation
POLIMI	Principles of Programming Languages	5	136	MCSE	2023-24	NYA
POLIMI	Algoritmi e Principi dell'Informatica	10	285	CSE	2023-24	NYA
POLIMI	Prova Finale di Algoritmi e Strutture Dati	1	340	CSE	2023-24	NA
POLIMI	Principles of Programming Languages	5	205	MCSE	2022-23	3.2/4
POLIMI	Algoritmi e Principi dell'Informatica	10	343	CSE	2022-23	3.1/4
POLIMI	Prova Finale di Algoritmi e Strutture Dati	1	424	CSE	2021-22	NA
POLIMI	Principles of Programming Languages	5	110	MCSE	2021-22	3.1/4
POLIMI	Algoritmi e Principi dell'Informatica	10	337	CSE	2021-22	3.2/4
POLIMI	Prova Finale di Algoritmi e Strutture Dati	1	344	CSE	2021-22	NA
POLIMI	Principles of Programming Languages	5	84	MCSE	2020-21	3.3/4
POLIMI	Algoritmi e Principi dell'Informatica	10	318	CSE	2020-21	3.0/4
POLIMI	Prova Finale di Algoritmi e Strutture Dati	1	363	CSE	2020-21	NA
POLIMI	Principles of Programming Languages	5	103	MCSE	2019-20	3.3/4
POLIMI	Algoritmi e Principi dell'Informatica	10	274	CSE	2019-20	3.1/4
POLIMI	Prova Finale di Algoritmi e Strutture Dati	1	286	CSE	2019-20	NA

INSTITUTIONAL RESPONSIBILITIES (SELECTION OF PAST 10-YEARS)

2024 **Scientific Committee Member** for the university libraries, Politecnico di Milano.
2021 **Commission Member** for the selection of a RTDb (Senior) Researcher in Computer Science, Politecnico di Milano.
2019 **Commission Member** for admittance to the PhD program in Computer Science.
2017 – 2021 **Member of the PhD Board**, PhD Programme in Computer Science, Politecnico di Milano.
2012 – present **Supervisor for the Evaluation of Research Products** in Computer Science and Engineering of my Department**.**
2011 – present **Associate Researcher**, IEIIT, CNR.
2011 – 2017 **Member of the Board** of the AICT Society, the National Association of Information Technology and Communication Engineers. I served as **head of the TIAI** (Internet and Information Technologies) group
2016 **Commission Member** for the selection of a RTDa (Junior) Researcher in Computer Science, University of Padova.

SUPERVISION OF MASTER, DOCTORAL STUDENTS AND POSTDOCTORAL RESEARCHERS

2010 – present **Advisor/Co-advisor** of **5** Doctoral Students at Politecnico di Milano.
2017 – 2021 **Tutor** of 11 Doctoral Students, and **3** Postdoctoral Students at Politecnico di Milano.
1998 – present **Advisor/Co-advisor** of **24** students in Computer Engineering, Politecnico di Milano.

ORGANIZATION OF SCIENTIFIC MEETINGS

2024 **General Chair of FM'24**, the 26th International Symposium on Formal Methods https://www.fm24.polimi.it/. FM is the leading conference on Formal Methods and its **CORE ranking is A**. FM'24 has 5 co-located conferences: TAP, FMICS, LOPSTR, PPDP, FACS, and 6 workshops. This year acceptance rate was 24%.
2024 **General Chair of PPDP'24**, the 26th International Symposium on Principles and Practice of Declarative Programming.
2011 - 2016 **Co-Chair** of 6 AICT Workshops, Politecnico di Milano:

- *The New Paradigm of Energy Efficiency* (2011)
- *Smart and Sustainable Mobility: New Paradigms* (2012)
- *Cloud Computing* (2013)
- *Big Data: Applications and Enabling Technologies* (2014)
- *IoT: a World of Sensors* (2015)
- *Accessibility of ICT Solutions for Disability in Work and Formation* (2016).

COMMISSIONS OF TRUST (SELECTION OF PAST 10-YEARS)

2023 **Expert project reviewer** for Regular Fondecyt National Projects Competition, National Research and Development Agency (ANID), Chile.
2018 – present **Expert committee member** of the European Association for Programming Languages and Systems (EAPLS) for the Best PhD Dissertation Award (EP17, EP18, EP19, EBDA20, EAPLS21, EAPLS22).
2019 **Program committee member** of 13th International Conference on Language and Automata Theory and Applications (LATA), Saint Petersburg, Russia, March 26-29, 2019.
2018 **Program committee member** of 12th International Conference on Language and Automata Theory and Applications (LATA), Bar-Ilan near Tel Aviv, Israel April 9-11, 2018.
2017 **Program committee member** of 9th Workshop on Context-Oriented Programming (COP) Co-located with ECOOP 2017, Sun 18-Fri 23 June 2017, Barcelona, Spain
2015 **Independent Expert** for the International PhD School in Formal Languages and Applications, URV, Tarragona
2014 **Program committee member** of 4PAD/PDP 2014 22nd Euromicro International Conference on Parallel, Distributed and network-based Processing, Special Session on Formal Approaches to Parallel and Distributed Systems.

TECHNOLOGY TRANSFER

DEVELOPMENT OF OPEN-SOURCE TOOLS

Zot (since 2006) is an open-source extendable *bounded model/satisfiability checker*. I designed and implemented Zot and maintained it for several years. Zot supports operational, descriptive, or hybrid models. Its plug-in based architecture permits both mono- and bi-infinite discrete temporal domains, and supports dense-time-based formalisms. Zot is used as a verification engine in several research activities and funded projects of our department; also, it was used for years in the course on *Formal Methods in Concurrent and Distributed Systems.*
Link: https://github.com/fm-polimi/zot

PAPAGENO (since 2013) I collaborated in the design and development of the open-source tool PAPAGENO,

an efficient Parallel Parser Generator based on Operator Precedence Languages. We developed very efficient parallel parsers for JSON, Lua, JavaScript with it, and we are currently investigating its applicability for parallel querying on very large structured documents (e.g. XML). Its development in recent years has always been through temporary researchers and students of mine assisted by A. Barenghi.
Links: https://github.com/PAPAGENO-devels/papageno,
https://github.com/simoneguidi94/gopapageno

POMC (since 2020) an open-source model checker for Operator Precedence Languages (OPL), based on a first-order complete temporal logic called POTL. Besides OPL, it naturally supports Visibly Pushdown languages, but can also check "non-visibly" properties of programs, e.g. related to exceptions, transactions or continuations. Still in a prototypal stage, it already exhibits very promising performance on real-world cases, and it will be presented at Computer Aided Verification (CAV) 2021, the most important conference on automatic verification. POMC is developed with the aid of Michele Chiari, a former PhD student of mine, and two MS students.
Link: https://github.com/michiari/POMC

COLLABORATION WITH INDUSTRY

My interests, teaching and research on programming languages are a natural aspect of dissemination and collaboration with the industry.

Leonardo SpA I recently started a collaboration with *Leonardo Helicopters* together with Prof. Maria Prandini. The objective of this collaboration is to define a structured requirement specification, validation and verification framework to support the development of rotorcraft Flight Control Systems (FCS) in line with certification regulation and applicable standards. This collaboration will start with a PNRR PhD grant, starting in November 2022.

Robox SpA I have a collaboration started in 2016 for the development of their next-generation programming language for motion control, called R++. As reported before, I was the head of the Robox-financed research project that lead to the design and implementation of R++. R++ is closed-source and considered one of the most important assets of the firm.
My former student L. Nardo works at Robox and is in charge of the R++ compiler. At present one of my students of Principles of Programming Languages is developing a joint Master thesis on a number of linguistic extensions to R++.

Industrial Training I taught a number of courses on programming paradigms in various companies reported here.
In particular, in 2019 I designed a 20-day course with Cefriel for Nokia, on *Machine Learning & Deep Learning SW Development for Big Data Platforms*, where I covered programming languages and paradigm.

Company	**Course name**	**Length**	**No. of students**	**Year**
Nokia	Functional Programming in Python and Scala	16 hours	~10	2020
SM-Optics	New Programming Languages	16 hours	~10	2019
MOXOFF	Functional and Monadic Programming in Scala	8 hours	~10	2019
SM-Optics	New Programming Languages	16 hours	~10	2018
CESI SpA	The Python Programming Language	18 hours	~10	2008

TWELVE MOST RELEVANT PUBLICATIONS

1. D. Mandrioli, M. Pradella, S. Crespi Reghizzi, Aperiodicity, Star-freeness, and First-order Definability of Operator Precedence Languages, Logical Methods in Computer Science, Volume 19, Issue 4 (2023)
   CORE ranking: A.

2. M. Chiari, D. Mandrioli, F. Pontiggia, M. Pradella, A Model Checker for Operator Precedence Languages, ACM Transactions on Programming Languages and Systems, Vol. 45, No. 3, Article 19 (2023)
   CORE ranking: A*.

3. Michele Chiari, Dino Mandrioli, Matteo Pradella: A First-Order Complete Temporal Logic for Structured Context-Free Languages, Logical Methods in Computer Science, Volume 18, Issue 3, pp. 11:1-11:49 (2022)
   CORE ranking: A.

4. Michele Chiari, Dino Mandrioli, Matteo Pradella: Operator precedence temporal logic and model checking, Theoretical Computer Science, vol 848, pp. 47–81 (2020)
   CORE ranking: A.

5. Stefano Crespi Reghizzi, Matteo Pradella: Beyond operator-precedence grammars and languages, Journal of Computer and System Sciences (JCSS), 113 18–41 (2020)
   CORE ranking: A*.

6. Violetta Lonati, Dino Mandrioli, Federica Panella, Matteo Pradella: Operator Precedence Languages: Their Automata-Theoretic and Logic Characterization. SIAM J. Comput. 44(4): 1026-1088 (2015)
   CORE ranking: A*.

7. Matteo Pradella, Angelo Morzenti, Pierluigi San Pietro: Bounded satisfiability checking of metric temporal logic specifications. ACM Transactions on Software Engineering and Methodology 22(3): 20:1-20:54 (2013)
   CORE ranking: A*.

8. Guido Salvaneschi, Carlo Ghezzi, Matteo Pradella: An Analysis of Language-Level Support for Self-Adaptive Software. ACM Transactions on Autonomous and Adaptive Systems 8(2): 7:1-7:29 (2013)
   CORE ranking: B.

9. Guido Salvaneschi, Carlo Ghezzi, Matteo Pradella: Context-oriented programming: A software engineering perspective. Journal of Systems and Software 85(8): 1801-1817 (2012)
   CORE ranking: A.

10. Matteo Pradella, Stefano Crespi-Reghizzi: A SAT-based parser and completer for pictures specified by tiling. Pattern Recognition 41(2): 555-566 (2008)
    CORE ranking: A*.

11. Stefano Crespi Reghizzi, Matteo Pradella: Tile rewriting grammars and picture languages. Theoretical Computer Science 340(1): 257-272 (2005)
    CORE ranking: A.

12. Alberto Coen-Porisini, Matteo Pradella, Matteo Rossi, Dino Mandrioli: A formal approach for designing CORBA-based applications. ACM Transactions on Software Engineering and Methodology 12(2): 107-151 (2003)
    CORE ranking: A*.

RESEARCH STATEMENT

ON-GOING RESEARCH DIRECTIONS AND RECENT ACHIEVEMENTS

Since the start of my scientific activity, my main interest has been on basic, long-term research in **formal verification of software**. Verification is a fundamental step in every engineering process: after designing and possibly building a new artifact it is mandatory to verify whether it complies with the desired standard (safety, efficiency, etc.). Traditional engineering has a well-established history of verification methodologies which span from the analysis of mathematical models of the designed product to the experimental test in the field.

Not so with the verification of software artifacts. Software in fact exhibits distinguishing features that make it hard to apply verification techniques that are effective in other fields of engineering. In most practical cases, software verification consists just in supplying some test cases and checking whether the system behaves as expected at least in those cases. Such approach, however, suffers from obvious weaknesses and is at the origin of the "Software Crisis", a term introduced in the late '60s to express that large software projects are often unreliable, out of budget, out of schedule, and is still largely present in the literature of the field. The lack of software reliability is *even more critical nowadays* when software is so invasive and pervasive that practically every moment of our life depends on it, whether we are on a self-driving car or visiting a web site.

My approach to the research on software verification has always been "two-pronged":

1) **Basic theory** on notations, languages, automata, logics.
2) Design and development of **tools**.

My strong belief is that the availability of good, industrial-strength tools will be a natural "gateway drug" for the adoption of formal methods in industry. Of course, Point 1) is a natural pre-requisite for having Point 2). For this reason, I spent years studying and proposing theories, e.g. on various variants of modal and temporal logics, automata models, and specification languages, that in several cases brought the development of tools.

To cite here the foremost:

- My research activity on the various variants of **TRIO metric temporal logic** was the necessary step that led to the design and development of the Zot bounded model checker (2007-2014), still actively developed and used by colleagues in my group.
- The more recent activity on **Operator Precedence languages**, based on the definition of an automaton model, and characterization through logics, both monadic second order (MSO) and temporal, and an extended version of regular expressions, was a necessary stepping stone for developing the recent POMC model checker (2021).

Operator Precedence languages (OPL) were defined by Robert Floyd in the '60s for deterministic parsing. In recent years, we "re-discovered" them as they are much more expressive than a very successful family of languages, the one of Nested Words (NW) (also known as Visibly-Pushdown or Input-Driven languages). R. Alur, one of the most renowned researchers in model checking, proposed Nested Words for verification of software, since they easily and naturally model programs and enjoy a lot of the properties that make Regular Languages the natural basis of model checking verification. This spurred a thriving stream of research that spreads to this day.

In the last decade we were able to prove that:

1) OPL are **much more expressive** than NW. Indeed, while NW are limited to parenthesis-like languages such as XML, HTML, and Lisp, OPL do not suffer from this limitation and can easily describe syntactically more complex programming and data languages, such as JSON, JavaScript, Lua, HTML5; more interestingly, as far as software verification is concerned, OPL can be naturally used to model behavior of e.g. exceptions, continuations, transactions, etc.
2) OPL enjoy the **same closure properties** of NW, retaining also basically the **same theoretical complexities**.

One of the critical aspects of the research on Nested Words is that it is for the most part **only theoretical**. Various kinds of interesting logics, models and verification algorithms for NW were proposed through the years, with very few applicative accomplishments. On the other hand, after spending years in defining and refining the necessary theory, models, logics and algorithms, **we were finally able to design and implement a complete model checker for OPL**, called POMC (Precedence-Oriented Model Checking) and based on the first-order complete temporal logic POTL (Precedence-Oriented Temporal Logic). This tool can of course be directly used on less expressive notation, most notably Nested Words. The tool is still a prototype; nonetheless we were able to

obtain good performance on quite extensive benchmarks, and a paper on was presented at CAV'21 (Computer Aided Verification), the most important conference on automatic verification.

SHORT- TO MID-TERM RESEARCH DIRECTIONS

Besides the many interesting "theoretical tiles" we still need to find to complete the suggestive mosaic of Operator Precedence languages, the natural short- to mid-term direction of my research is on **applications**.

- Having a model checker suitable for software verification, it is only natural to think to apply it. We want to adapt and extend POMC to support *de facto* standard formats for verifications, such as Boolean programs, and of course to support **real programming languages**, e.g. through abstractions.
- Natural applications based on such kind of model checker are obvious, from security to safety-critical systems, so this could be a good point for collaborations with people working in the fields of **security** and **embedded systems**.
- OPLs, unlike more general DCFLs, enjoy the **local parsability property**. This means that a document can be parsed deterministically by starting its analysis in any position with no need to make a nondeterministic guess on the parser's state, which would expose to the risk of backtrack with the obvious consequences in terms of complexity. This property opens the door to parallelizing the analysis of OPLs in quite an effective way. PAPAGENO, a parser generator based on this principle has been realized and applied to various programming and data representation languages with experimental results that fully confirmed the theoretical expectations in terms of speed-up. The local parsability property naturally supports also incremental parsing in a simpler way than previous algorithms for general deterministic context-free languages.
- By combining the automata-theoretic approach of POMC with PAPAGENO, it is natural to go not only toward **runtime verification**, but also to work on **parallel querying on complex tree-structured data** (possibly streaming). This activity could be developed with people working on **big data** and **data-base**.

As far as **theory** is concerned, we recently proved a major result in the theory of formal languages, by extending to **all structured context-free languages** (OP but also Nested Words) the seminal result proved in 1971 by McNaughton and Papert which states that, for regular languages, **star-free languages, first-order definable languages and aperiodic languages are equivalent**. In many years there were various scientific works in the literature for bringing this seminal result to tree-like structures, but were all partial.

TEACHING STATEMENT

SHORT- TO MID-TERM TEACHING PLAN

My main teaching activities are on two courses, one at the under-graduate level and one at graduate level:

1) Algorithms and Principles of Computer Science.
2) Principles of Programming Languages.

Algorithms and Principles of Computer Science

I collaborated to the definition and creation of the course on Algorithms and Principles of Computer Science, originally a merge of the former *Theoretical Computer Science* and *Algorithms and Data Structures* (*Informatica 3*). The main object of the course is to cover the basic parts of theoretical computer science, in particular formal models and languages, e.g. automata, Turing machines, logic; computability theory; basic complexity theory; basic and standard data structures and algorithms.
This course did not change much in many years; recently we added a more practical part, i.e. the **Project on Algorithms and Data Structures**. The approach we decided to follow is based on an automatic contest, where we define a project for the current year, and the students apply their code to an evaluation web-site that is continuously running for a few months. The evaluation is totally automatic and based on a set of tests, one group that is publicly available, and the other that is private, with some strict requirements for both memory and time. The final grade depends on how many of the tests are passed, and the student has a feedback in real-time.
I believe that this more practical part of the course is a strength that we could use also for other parts as well. For instance, one of the projects was to implement a non-deterministic Turing machine

simulator, and I strongly believe that the actual implementation of such theoretical models gave the students precious insights on the abstract but fundamental concept of **non-determinism**.
Also, I think that this course is the natural place for preparing students for the participation in international contests such as the International Collegiate Programming Contest (ICPC), for which I served for some years as coach of some of the PoliMi teams. Contests can be an important spur for learning, so I envision that we could create some internal contests also for this course, adapting the techniques we developed for the project evaluation.

Principles of Programming Languages
I designed the graduate course on Principles of Programming languages since its inception in 2011. The main scope of the course is to offer a broad overview of the most important programming paradigms. The course presents the salient features in the landscape of programming languages, by analyzing similarities and differences, traditional and recent approaches and paradigms. I show significant fragments of some important programming languages, given as examples of those paradigms considered in class. The course aims to provide the means to better understand the essence of defining concepts of programming languages, so to allow critical choice about the level of abstraction, and consequently the language necessary to implement a particular system. Also, the student must attain a good mental flexibility before of an aspect, i.e. the choice of a language, that is constantly changing in computer science and software.
The main programming paradigms covered start for **procedural** for the basic concepts, to re-visit the fundamental ideas of **object orientation**, studied in depth in previous courses, e.g. with respect to **class-less or prototype-based object systems**. Other important approaches covered are **meta-programming techniques**, **functional** programming, **monadic** programming, and **concurrent** programming.
In the last years, I shifted the interest from some of the classical programming paradigms, e.g. the logic one of Prolog, toward Erlang's reactive and concurrency-oriented one. Nevertheless, I think it is important to always keep track of the roots, from which the covered concepts originate.
My first plan is to keep up with current trends in programming, to cover some new promising approaches; to name a few:

- **Dependent typing** (e.g. Agda, Idris).
- **Safe compile-time techniques for memory safety** (Rust, Zig).

For the time being, the course approach is quite traditional: class and exercises with a standard open-book exam where I ask the students to write small programs in different programming languages. On the other hand, I strongly encourage experimentation with the compilers, both in class and at home.
I think that it is natural for this course to experiment with **small projects and contests**. In particular, I think that in a few years this approach could encompass small contests modeled like the *Computer Language Benchmarks Game*, where different language implementations are compared on a number of standard benchmarks.