

# Functional and Logic Programming

*Bachelor in Informatics and Computing Engineering*  
2025/2026 - 1<sup>st</sup> Semester

## Introduction to Logic Programming

# Agenda

- Logic Programming
- Prolog
- Applications
- Extensions

# Logic Programming

- Logic Programming is a *Declarative* style of programming
  - The programmer describes **what** they want the result to be
  - You already knew at least two examples:
    - SQL
    - Haskell
- Contrasts with *Imperative Programming*
  - The programmer specifies **how** the computer should obtain the result
  - Most of what you have worked with, so far

# How vs. What (1)

## Imperative

```
var clients;  
forall (client in clients)  
{  
    if(client.age > 18)  
        print client.name;  
}
```

## Declarative

```
SELECT name  
FROM Clients  
WHERE age > 18;
```

## How vs What (2)

**Imperative**  
(e.g., Python)

```
total = 0
for x in [1, 2, 3, 4]:
    total += x
print(total)
```

**Declarative**  
(e.g., Haskell/Python)

```
sum([1, 2, 3, 4])
```

**Declarative**  
(e.g., SQL)

```
SELECT SUM(x)
FROM numbers;
```

# Imperative vs Declarative

Aspect	Imperative Programming	Declarative Programming
<b>Definition</b>	Focuses on <i>how</i> to perform tasks — the programmer specifies the exact sequence of steps to achieve the result.	Focuses on <i>what</i> the desired result is — the system figures out <i>how</i> to achieve it.
<b>Control Flow</b>	Explicit — the program controls the order of execution.	Implicit — the system determines the execution order.
<b>Analogy</b>	Giving someone a recipe step by step.	Ordering a meal and letting the chef decide how to cook it.

# Logic Programming

- The idea behind logic programming is:
  - when given a problem, instead of designing and writing an algorithm to solve it, we simply specify the problem, and the computer solves the problem
  - In reality, we have to be mindful about how the underlying solver works

# Advantages

- There are several reasons to use Logic Programming
  - Rapid prototyping
  - Usually small code footprint
  - Flexible and intuitive (once you get to know it)
  - Intrinsic explainability of results
- Also provides with better reasoning skills (from learning a different ‘way of thinking about problems’)

Logic programming is usually easier to learn than traditional programming

Yet, expert computer programmers often have

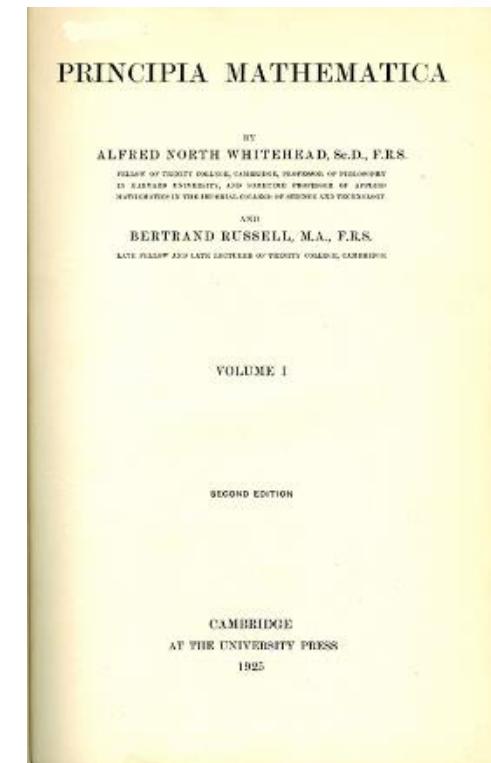
more trouble with logic programming than novice learners

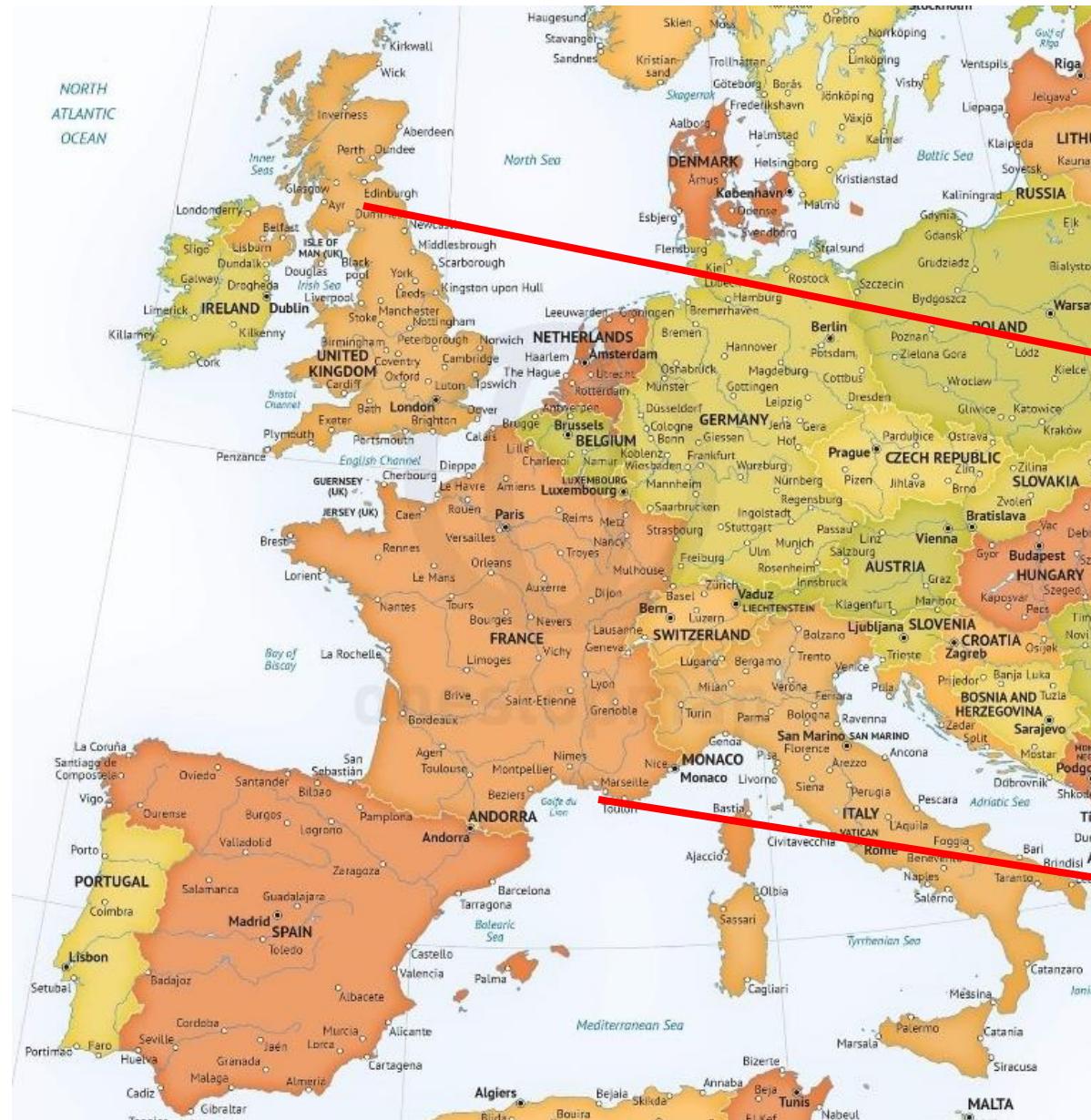
# Prolog

- Prolog is the most well-known Logic Programming language
  - Programs are descriptions of knowledge / relations
    - In the form of first-order logic predicates
  - A computation starts as a query, the program tries to prove the query
  - Not purely declarative (a compromise needed to make the language efficient, practical and useful)

# History of Prolog

- Origins in logic
  - Aristotle's works on logic (*Organon*, 40BC)
  - 1879: *Begriffsschrift*, by Gottlob Frege
    - Foundation of first-order logic, introduces quantifier notation, and solves some problems
  - 1910-1913: *Principia Mathematica*, by Alfred N. Whitehead and Bertrand Russell
    - Foundations of mathematics, attempting to derive mathematical truths from axioms and inference rules in symbolic logic

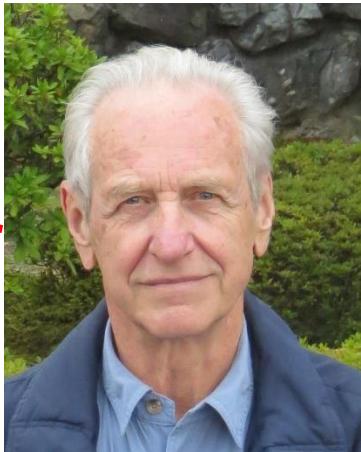




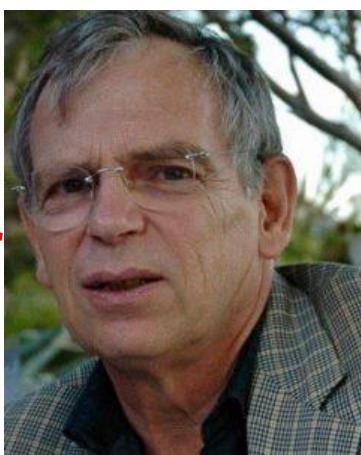
# Key People

- Automated theorem-proving in first-order logic
  - Question-answering system using natural language (French)

# Robert Kowalski



Robert  
Pasero  
  
Jean  
Trudel



# Alain Colmérauer



# Philippe Roussel

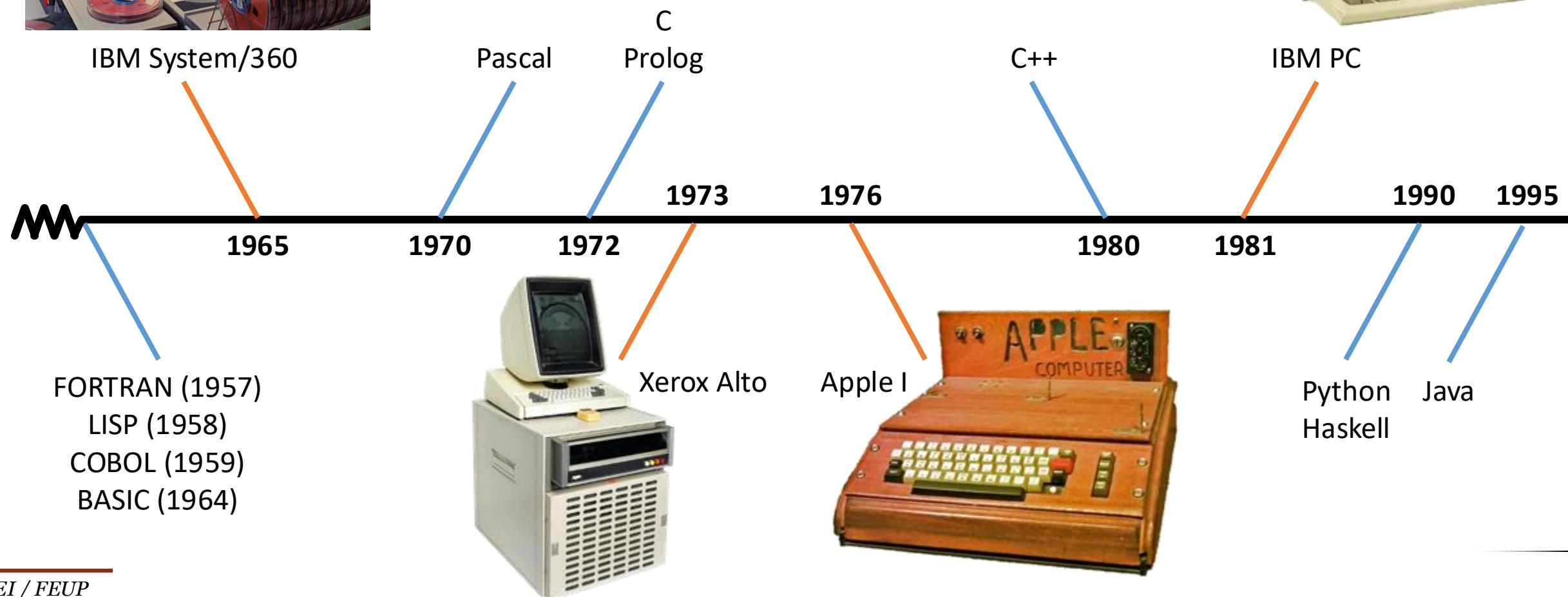
## Abridged Timeline

- 1960s: Early developments on automated theorem-proving in first-order logic and natural language processing
- 1972: Prolog (*Programmation en Logique*) is born
- 1970s: Advances in Prolog systems (Compiler, DCGs, ...)
- 1980s: Prolog gains popularity, especially in Europe and Japan
  - 1982: Fifth Generation Computer Systems Project
- 1990s: ISO Prolog (1995), parallel and concurrent systems
- 2000s: Several extensions and optimizations

According to an old joke, Logic Programming  
was invented in Edinburgh in 1974 and implemented in Marseille in 1972



## Contextual Timeline



# 50 Years of Prolog

*TPLP* 22 (6): 776–858, 2022. © The Author(s), 2022. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution-ShareAlike licence (<http://creativecommons.org/licenses/by/4.0/>), which permits re-use, distribution, and reproduction in any medium, provided the same Creative Commons licence is used to distribute the re-used or adapted article and the original article is properly cited.  
doi:10.1017/S1471068422000102 First published online 17 May 2022

## *Fifty Years of Prolog and Beyond\**

PHILIPP KÖRNER and MICHAEL LEUSCHEL

Institut für Informatik, Universität Düsseldorf, Universitätsstr. 1, D-40225 Düsseldorf, Germany  
(e-mails: [p.koerner@uni-duesseldorf.de](mailto:p.koerner@uni-duesseldorf.de), [leuschel@uni-duesseldorf.de](mailto:leuschel@uni-duesseldorf.de))

JOÃO BARBOSA and VÍTOR SANTOS COSTA

Department of Computer Science, Faculty of Science of the University of Porto, Porto, Portugal  
(e-mails: [joao.barbosa@fc.up.pt](mailto:joao.barbosa@fc.up.pt), [vs costa@fc.up.pt](mailto:vs costa@fc.up.pt))

VERÓNICA DAHL

Computing Sciences Department, Simon Fraser University, Burnaby, Canada  
(e-mail: [veronica\\_dahl@sfu.ca](mailto:veronica_dahl@sfu.ca))

MANUEL V. HERMENEGILDO and JOSE F. MORALES

IMDEA Software Institute and Universidad Politécnica de Madrid (UPM), Madrid, Spain  
(e-mails: [manuel.hermenegildo@imdea.org](mailto:manuel.hermenegildo@imdea.org), [josef.morales@imdea.org](mailto:josef.morales@imdea.org))

JAN WIELEMAKER

Centrum voor Wiskunde en Informatica (CWI), Amsterdam, Netherlands  
(e-mail: [J.Wielemaker@cwi.nl](mailto:J.Wielemaker@cwi.nl))

DANIEL DIAZ

Centre de Recherche en Informatique, University Paris-1, Paris, France  
(e-mail: [daniel.diaz@univ-paris1.fr](mailto:daniel.diaz@univ-paris1.fr))

SALVADOR ABREU

NOVA-LINCS, University of Évora, Evora, Portugal  
(e-mail: [spa@uevora.pt](mailto:spa@uevora.pt))

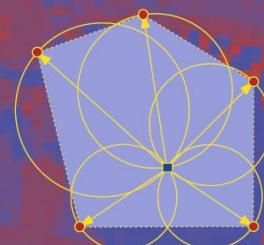
GIOVANNI CIATTO

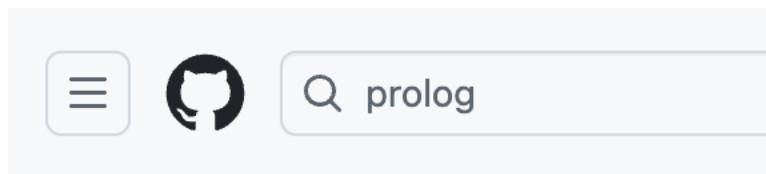
Department of Computer Science and Engineering, Alma Mater Studiorum—Università di Bologna, Bologna, Italy  
(e-mail: [giovanni.ciatto@unibo.it](mailto:giovanni.ciatto@unibo.it))

submitted 30 June 2020; revised 27 February 2022; accepted 6 March 2022

### Abstract

Both logic programming in general and Prolog in particular have a long and fascinating history, intermingled with that of many disciplines they inherited from or catalyzed. A large body of research has been gathered over the last 50 years, supported by many Prolog implementations. Many implementations are still actively developed, while new ones keep appearing. Often, the





Filter by

# Applications

- Prolog is not the most widely-used or popular programming language in the world, but...
- Used in several projects and areas:
  - [NASA Clarissa](#)
  - [IBM Watson](#)
  - [First Erlang interpreter](#)
  - New Zealand's main [stock broking system](#)
  - ...

See <https://www.quora.com/What-is-Prolog-used-for-today>

# Applications

- For many years, Prolog was one of the foremost languages used in Artificial Intelligence, especially in some noticeable application areas, such as:
  - Natural Language Processing
  - Expert Systems
  - Knowledge-Based Systems
  - Business Rules and Workflow
  - Computational Law
  - Planning and Scheduling
  - ...

# Extensions

- There are several extensions to Logic Programming, such as
  - Abductive Logic Programming (ALP)
  - Inductive Logic Programming (ILP)
  - Concurrent Logic Programming
  - Constraint (Logic) Programming

*“Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.”*

Eugene Freuder, 1997

(‘In Pursuit of the Holy Grail’, Constraints: An International Journal, 2, 57-61)

# Additional Readings & Resources

- Origins of Prolog
  - Robert A. Kowalski (1988). The Early Years of Logic Programming. *Communications of the ACM*, 31(1), pp. 38-43 (DOI: 10.1145/35043.35046)
  - Alain Colmerauer and Philippe Roussel (1996). The Birth of Prolog. In *History of Programming Languages*, pp. 331-367 (DOI: 10.1145/234286.1057820)
- ALP - The Association for Logic Programming
- ICLP - International Conference on Logic Programming
  - 2025 edition: <https://iclp25.demacs.unical.it/home-page>
- TPLP - Theory and Practice of Logic Programming. Cambridge University Press (<http://journals.cambridge.org/tlp>)

# Additional Readings & Resources

- Prolog Applications
  - Fumio Mizoguchi (1991). Prolog and its Applications: A Japanese Perspective. Springer (ISBN: 978-0-412-37770-9)
  - Alex M. Andrew (2005). The commercial use of PROLOG. *Kybernetes*, 34(5), pp. 599-601 (DOI: 10.1108/03684920510595300)
  - Manny Rayner, Beth Ann Hockey, Jean-Michel Renders, Nikos Chatzichrisafis and Kim Farrell (2005). Spoken Language Processing in the Clarissa Procedure Browser. *Natural Language Engineering* 1(1), 28 pages (ICSI Technical Report TR-05-005)
  - Joseph Armstrong (2003). Making reliable distributed systems in the presence of software errors. PhD thesis. Royal Institute of Technology, Stockholm, Sweden.
  - Alessandro Dal Palù and Paolo Torroni (2010). 25 Years of Applications of Logic Programming in Italy. A 25-Year Perspective on Logic Programming, pp. 300-328
  - [Awesome Prolog](#) - Curated list of Prolog resources (by Klaudio Sinani)

## Q & A

