What Can We Learn from the History of Programming Languages?

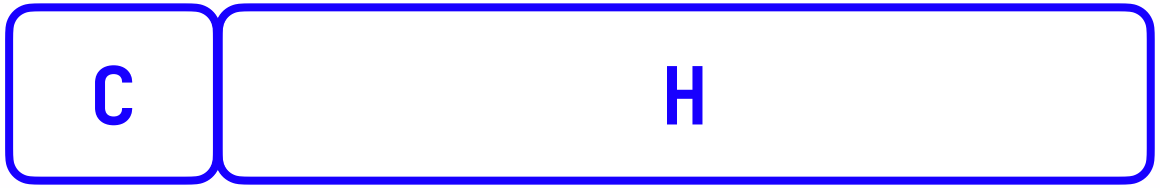
PLNL, 28 November 2025

Dr. Vadim Zaytsev aka @grammarware

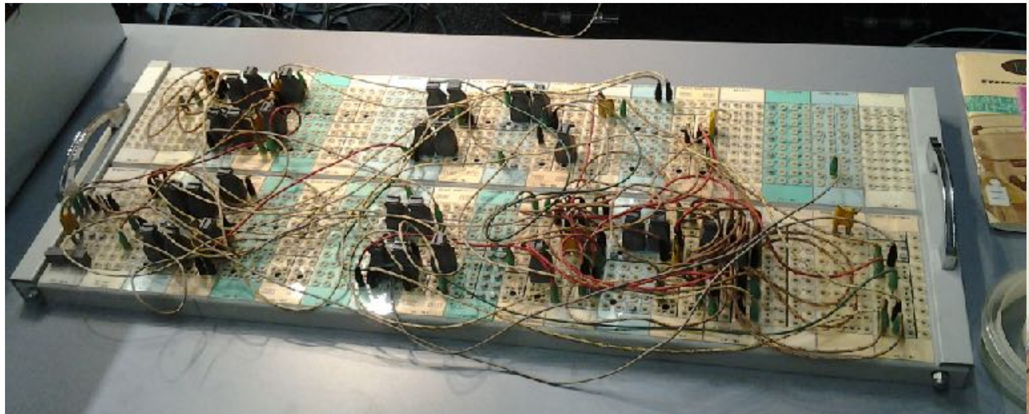
UNIVERSITY OF TWENTE. | FMT Formal Methods & Tools

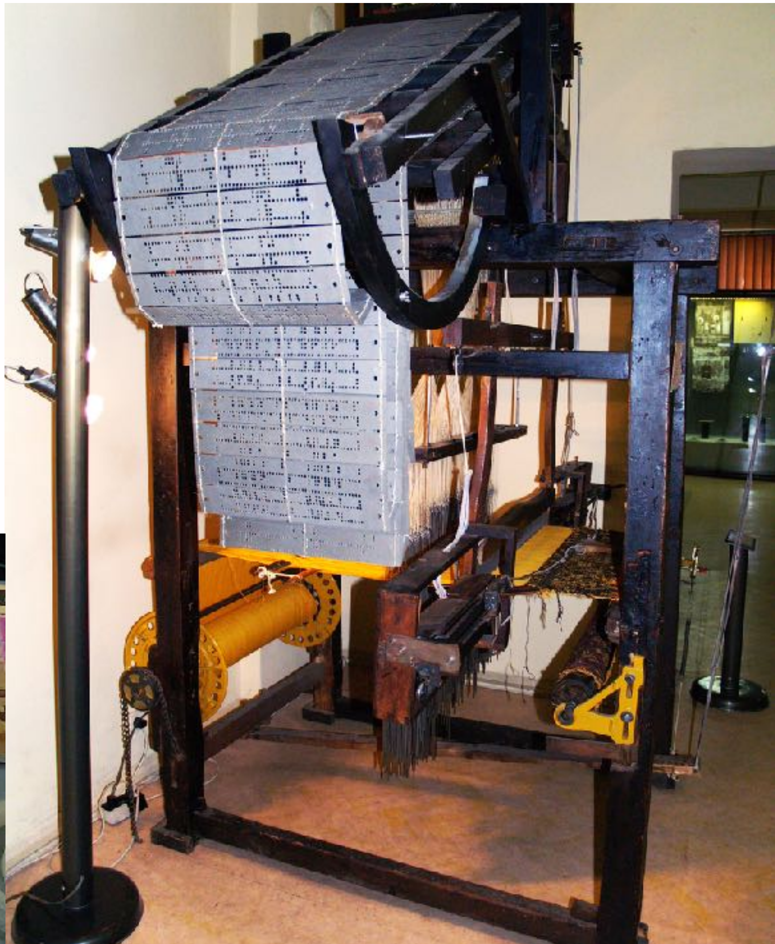
C

- Exhibits:
  - Napier (1617)
  - Pascal (1642)
  - de Colmar (1820)
  - Felt (1887)



- Exhibits:
  - Jacquard (1805)
  - Votey (1896)

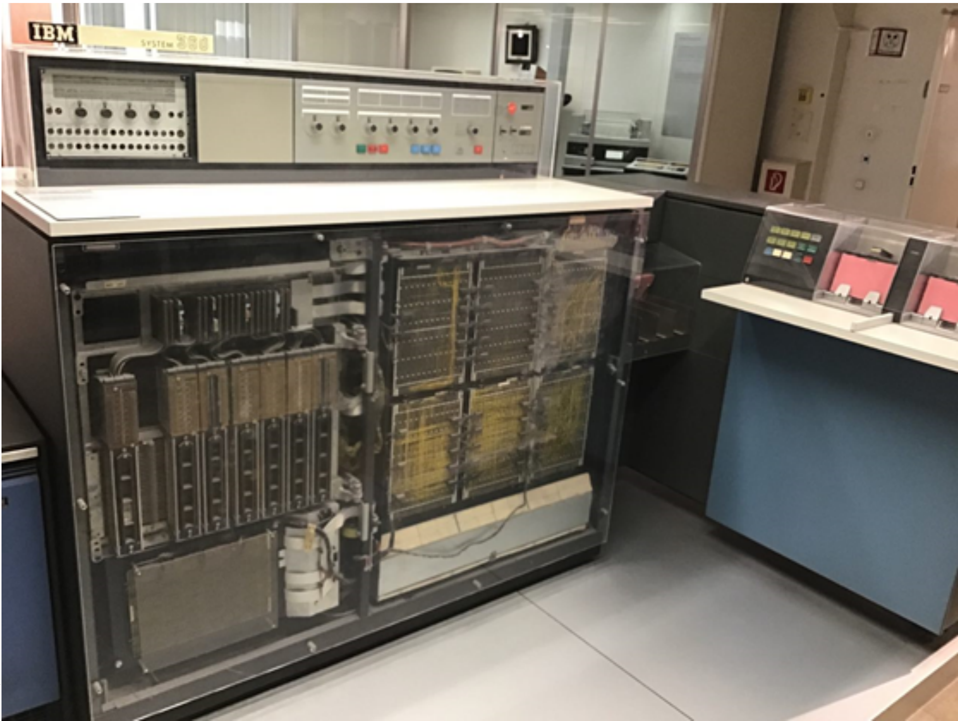


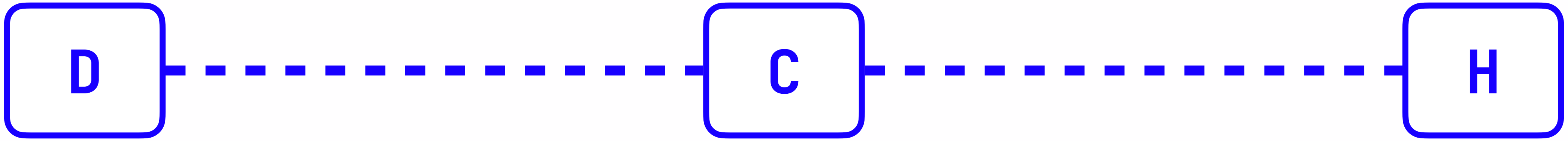


C

H

- Exhibits:
  - Turing 1936
  - Zuse 1936
  - ENIAC/SSEC/ARC2 1948
  - Brooks 1964



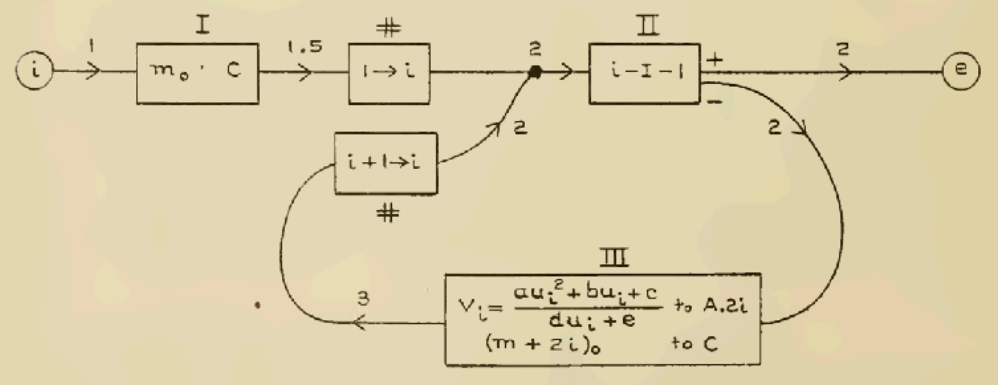


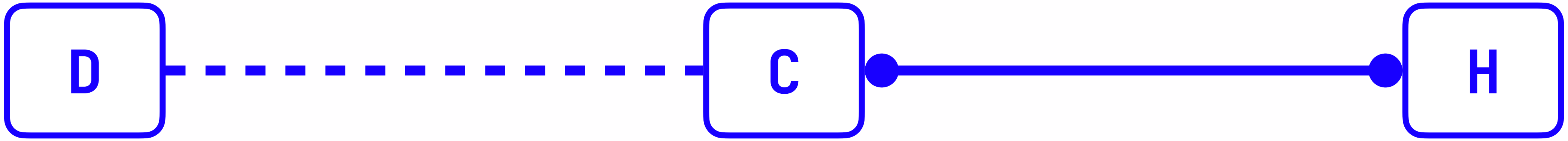
D

C

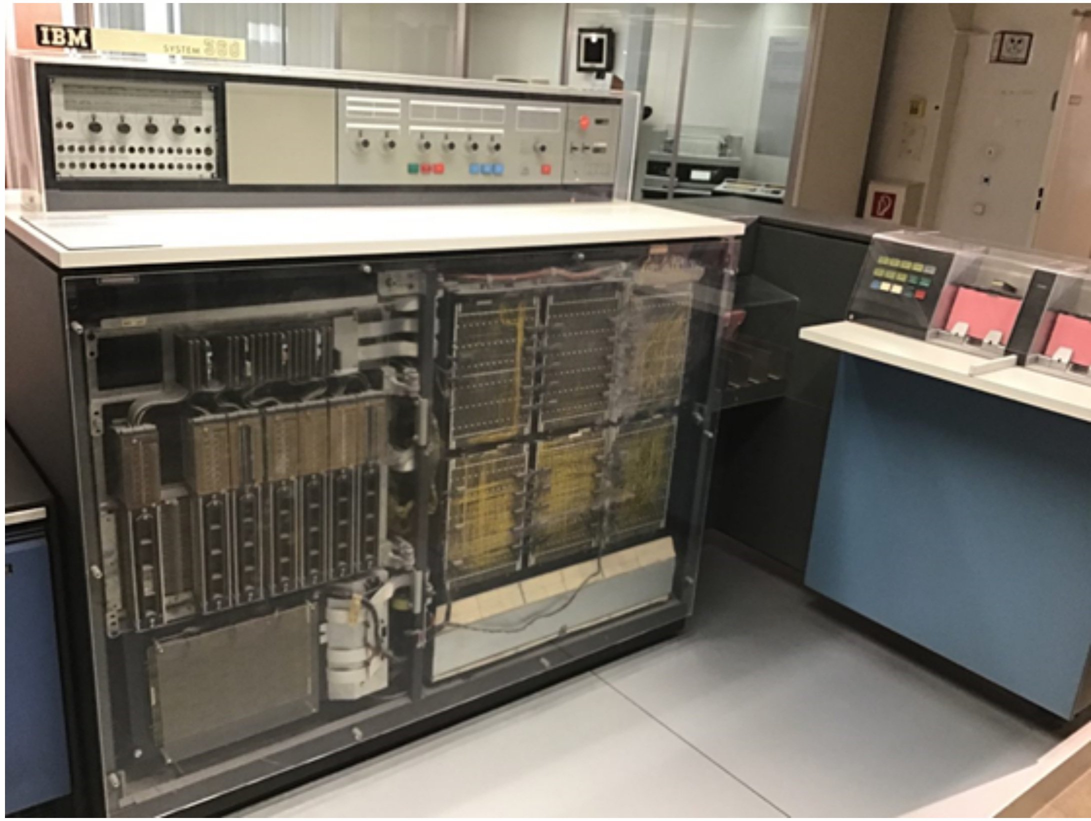
H

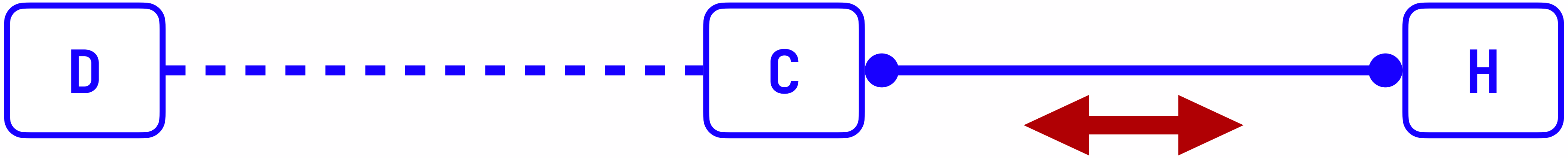
- Exhibits:
  - Goldstine & von Neumann 1947



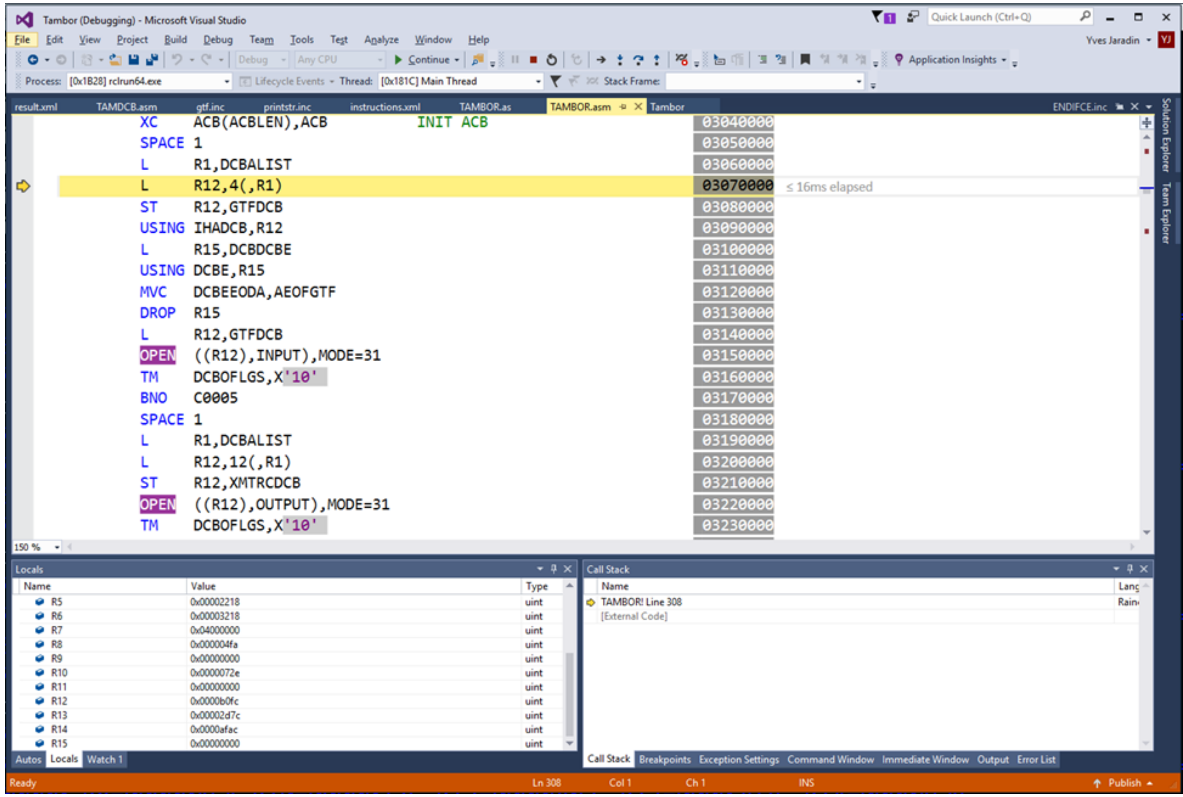


- Exhibits:
  - assembler (1950s+)





- Exhibits:
  - HLASM



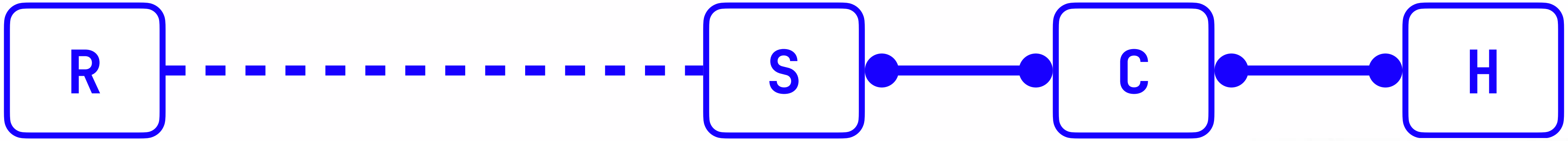
```
         XC    ACB(ACBLEN),ACB          INIT ACB                 03040000
         SPACE 1                                                 03050000
         L     R1,DCBALIST                                       03060000
         L     R12,4(,R1)                                        03070000  ≤ 16ms elapsed
         ST    R12,GTFDCB                                        03080000
         USING IHADCB,R12                                        03090000
         L     R15,DCBDCBE                                       03100000
         USING DCBE,R15                                          03110000
         MVC   DCBEEODA,AEOFGTF                                  03120000
         DROP  R15                                               03130000
         L     R12,GTFDCB                                        03140000
         OPEN  ((R12),INPUT),MODE=31                             03150000
         TM    DCBOFLGS,X'10'                                    03160000
         BNO   C0005                                             03170000
         SPACE 1                                                 03180000
         L     R1,DCBALIST                                       03190000
         L     R12,12(,R1)                                       03200000
         ST    R12,XMTRCDCB                                      03210000
         OPEN  ((R12),OUTPUT),MODE=31                            03220000
         TM    DCBOFLGS,X'10'                                    03230000
```

Locals

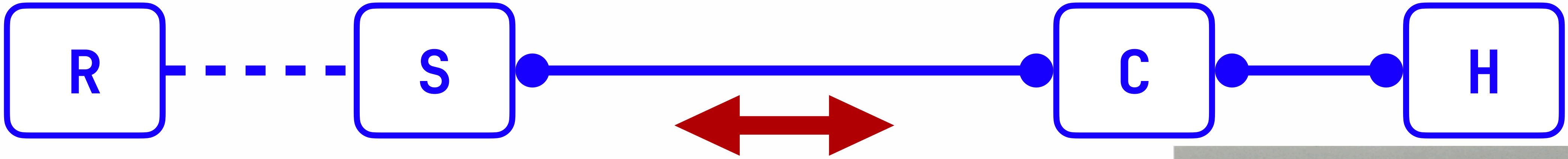
Name	Value	Type
R5	0x00002218	uint
R6	0x00003218	uint
R7	0x04000000	uint
R8	0x000004fa	uint
R9	0x00000000	uint
R10	0x0000072e	uint
R11	0x00000000	uint
R12	0x0000b0fc	uint
R13	0x00002d7c	uint
R14	0x0000afac	uint
R15	0x00000000	uint

Call Stack

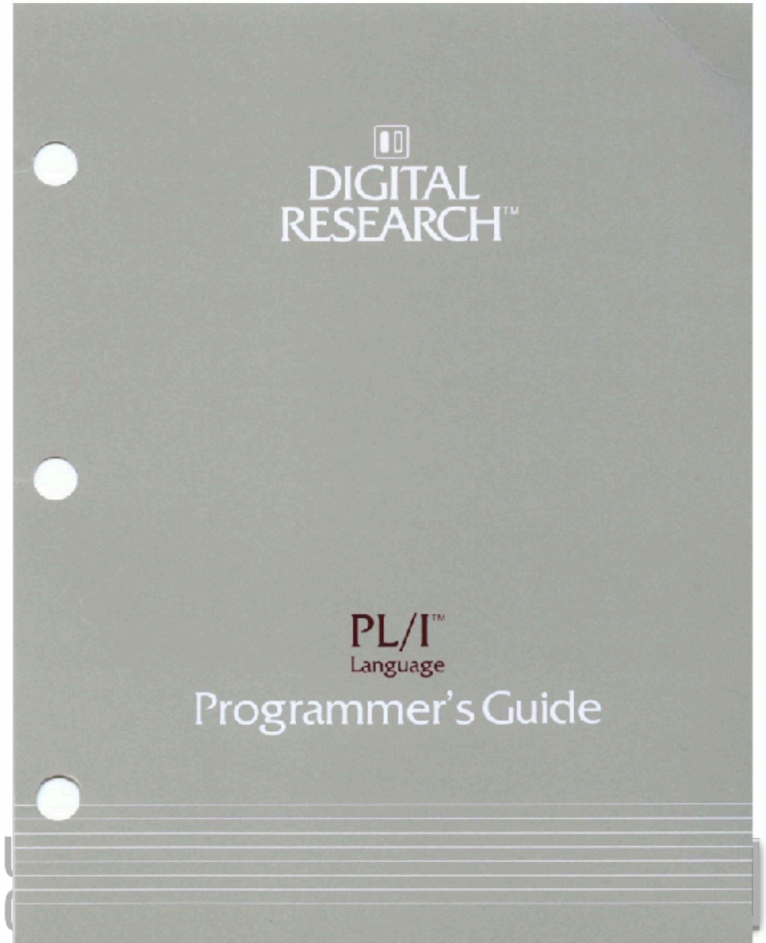
Name	Lang
TAMBOR! Line 308	Rain
[External Code]	



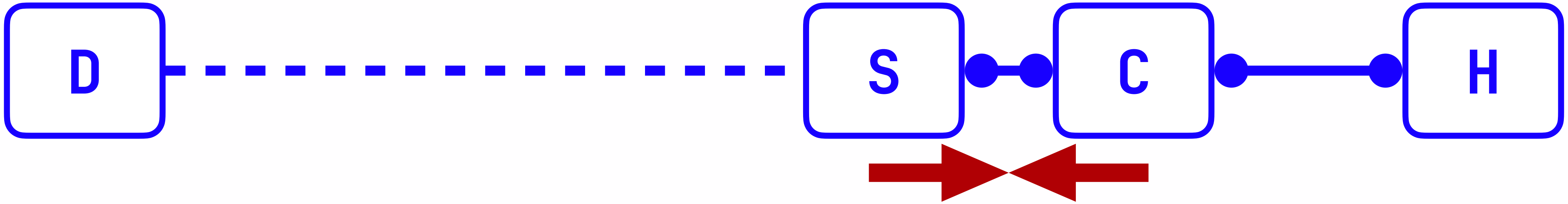
- Exhibits:
  - Rutishauser 1951
  - Hopper 1952
  - Backus 1957
  - CODASYL 1959
  - McCarthy 1958+
  - Hamilton 1969



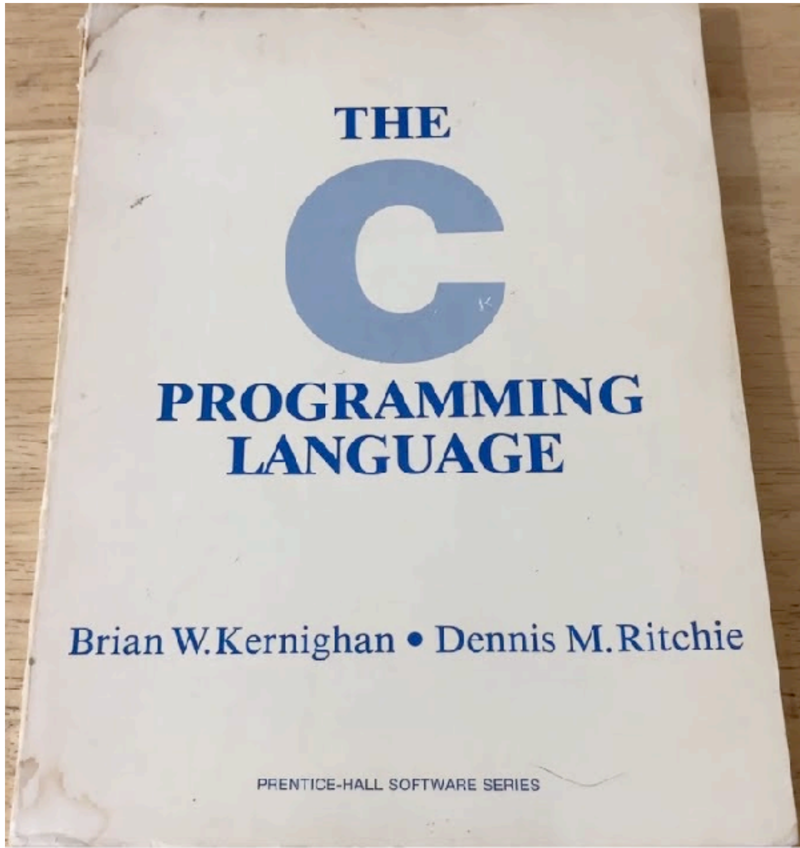
- Exhibits:
  - PL/I 1964


DIGITAL RESEARCH™

PL/I™
Language
Programmer's Guide



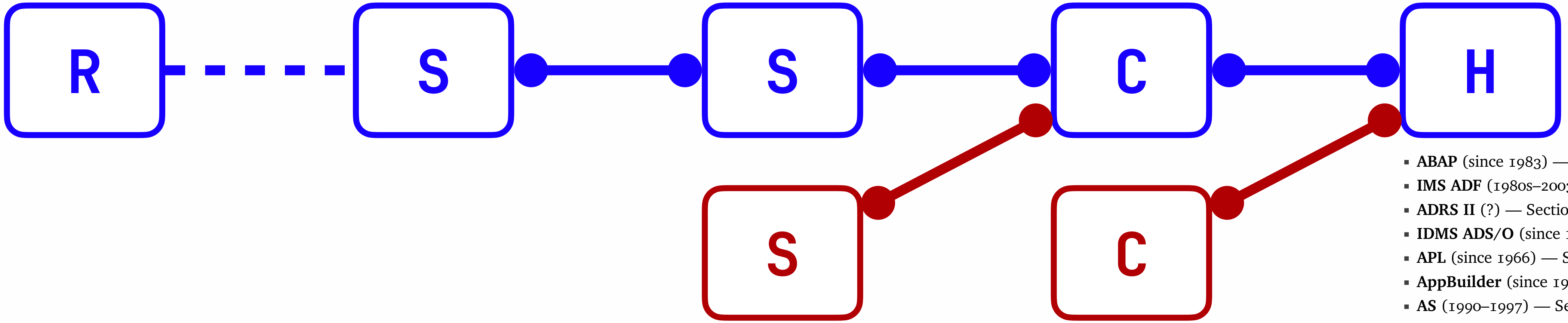
- Exhibits:
  - Moore 1970
  - Ritchie 1972+
  - Hoare 2006



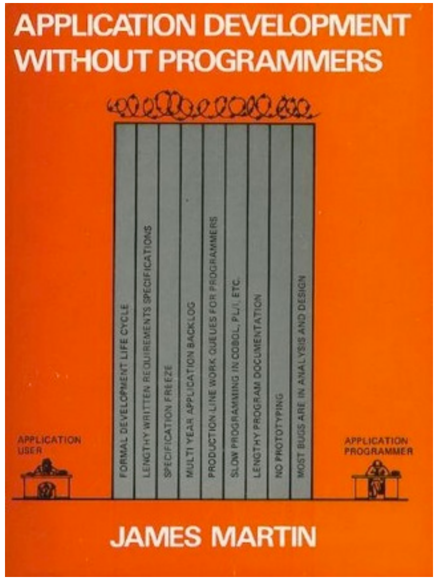
THE C PROGRAMMING LANGUAGE

Brian W. Kernighan • Dennis M. Ritchie

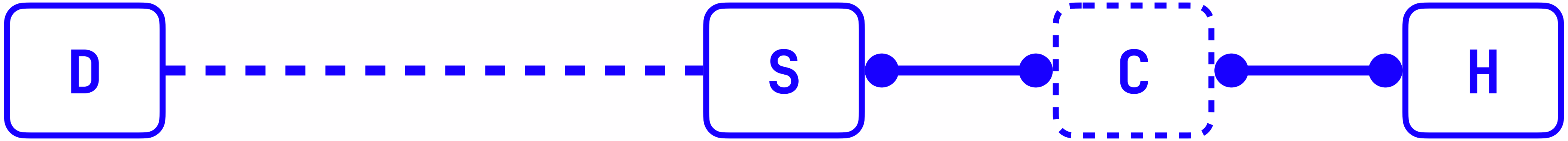
PRENTICE-HALL SOFTWARE SERIES



- Exhibits:
  - 4GLs


APPLICATION DEVELOPMENT WITHOUT PROGRAMMERS
FORMAL DEVELOPMENT LIFE CYCLE
LENGTHY WRITTEN REQUIREMENTS SPECIFICATIONS
SPECIFICATION FREEZE
MULTI YEAR APPLICATION BACKLOG
PRODUCTION LINE WORK QUEUES FOR PROGRAMMERS
SLOW PROGRAMMING IN COBOL, PL/I, ETC
LENGTHY PROGRAM DOCUMENTATION
NO PROTOTYPING
MOST BUGS ARE IN ANALYSIS AND DESIGN
APPLICATION USER
APPLICATION PROGRAMMER
JAMES MARTIN

- **ABAP** (since 1983) — Section 3.1;
- **IMS ADF** (1980s–2003) — Section 3.2;
- **ADRS II** (?) — Section 3.3;
- **IDMS ADS/O** (since 1980s) — Section 3.4;
- **APL** (since 1966) — Section 3.5;
- **AppBuilder** (since 1992) — Section 3.6;
- **AS** (1990–1997) — Section 3.7;
- **BUILDER** (1972–1994) — Section 3.8;
- **CICS** (since 1968) — Section 3.9;
- **IBM SAA CSP** (1980s–2001) — Section 3.10;
- **DATATRIEVE** (1978–2017) — Section 3.11;
- **DMS** (1980s–2005) — Section 3.12;
- **FOCUS** (1975) — Section 3.13;
- **GIS** (1966–?) — Section 3.14;
- **IDEAL** (since 1980s) — Section 3.15;
- **INTELLECT** (1960s?) — Section 3.16;
- **MANTIS** (since 1978) — Section 3.17;
- **MIMER** (1975/1981–?) — Section 3.18;
- **NATURAL** (1970s–2050) — Section 3.19;
- **NOMAD** (since 1970s) — Section 3.20;
- **PACBASE** (1970s–2015) — Section 3.21;
- **IBM db2 QMF** (?) — Section 3.22;
- **RAMIS** (1960s) — Section 3.23;
- **SYSTEM W** (?) — Section 3.24;
- **USE-IT** (1976–?) — Section 3.25.



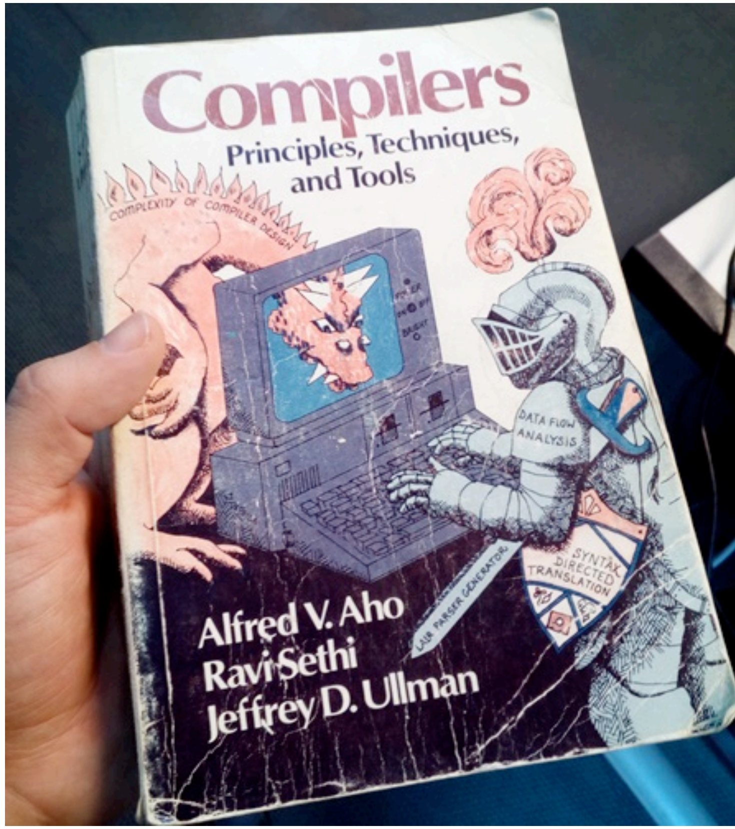
D

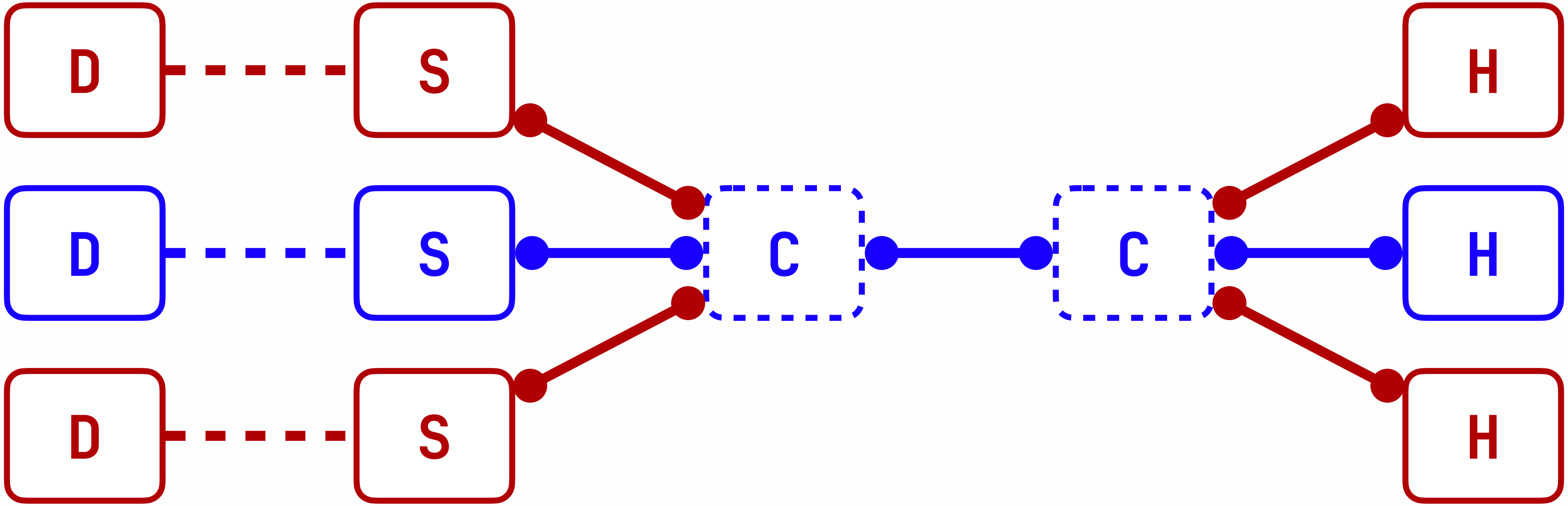
S

C

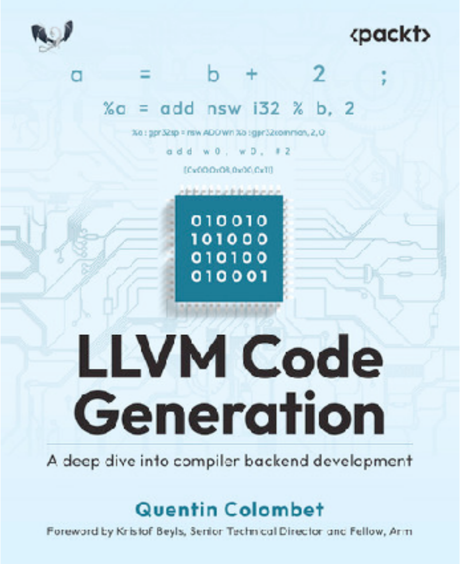
H

- Exhibits:
  - reliable compilers


Compilers
Principles, Techniques, and Tools
COMPLEXITY OF COMPILER DESIGN
DATA FLOW ANALYSIS
SYNTAX DIRECTED TRANSLATION
LALR PARSER GENERATOR
Alfred V. Aho
Ravi Sethi
Jeffrey D. Ullman



- Exhibits:
  - Dybvig 1984+
  - Adve & Lattner 2003+


<packt>

a = b + 2 ;

%a = add nsw i32 % b, 2

%a : gpr32sp = nsw ADDWri %b : gpr32common, 2, 0

add w0, w0, #2

[0x00,0x08,0x00,0x11]

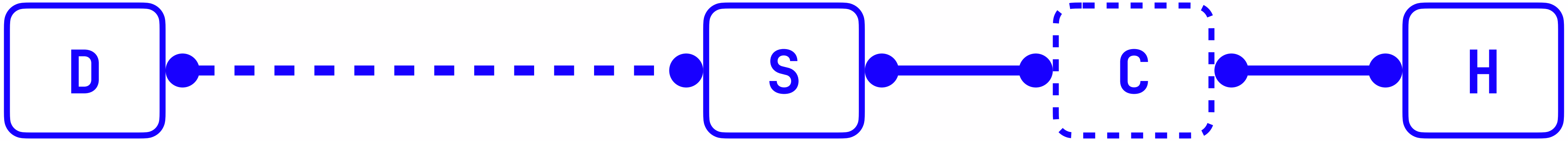
010010
101000
010100
010001

LLVM Code Generation

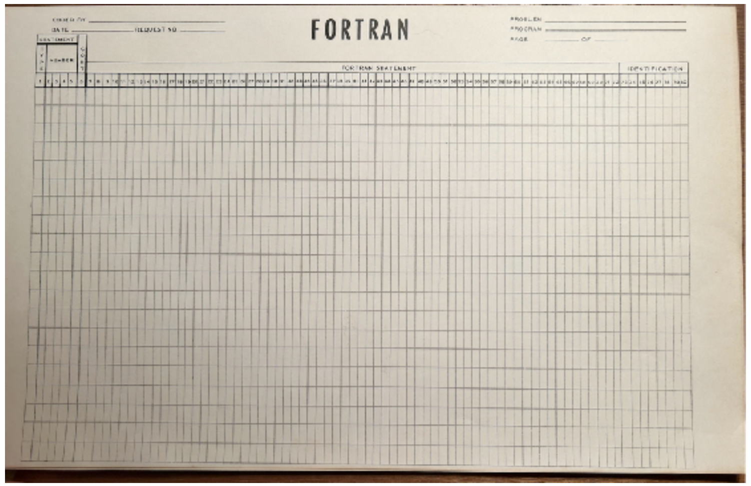
A deep dive into compiler backend development

Quentin Colombet

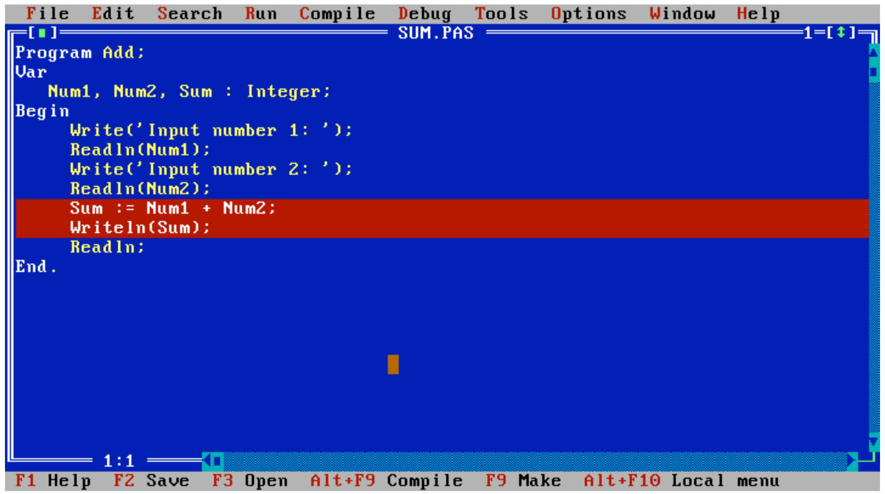
Foreword by Kristof Beyls, Senior Technical Director and Fellow, Arm



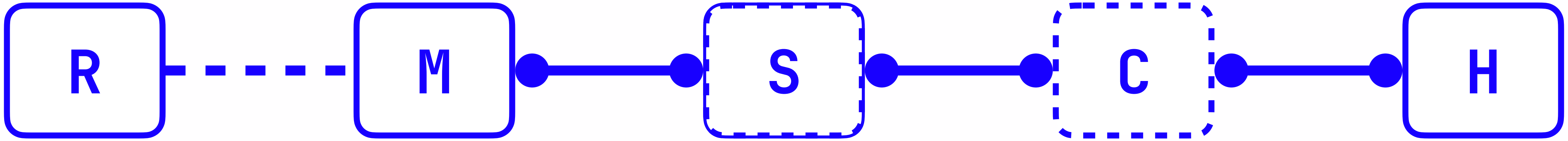
- Exhibits:



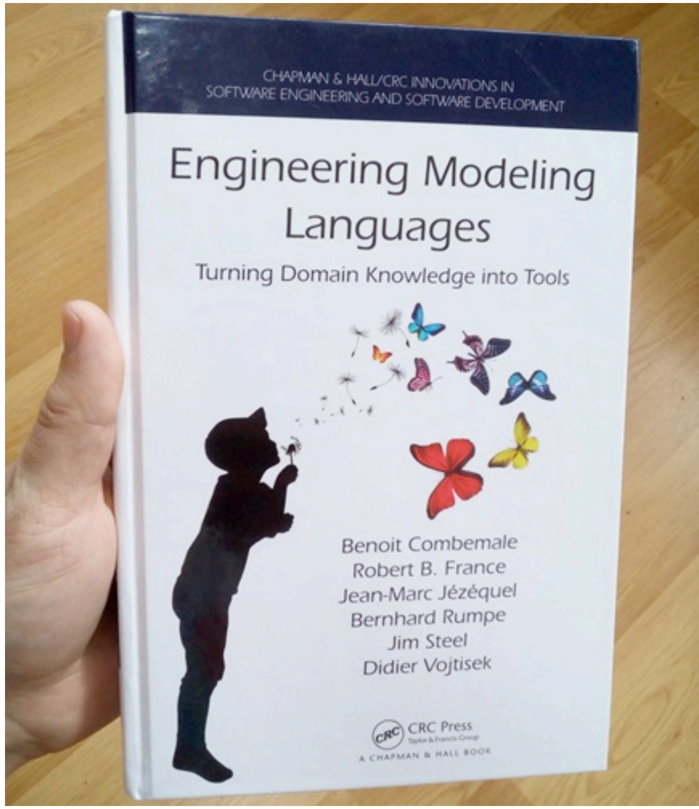
⟹


File Edit Search Run Compile Debug Tools Options Window Help
SUM.PAS
1:1
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu

```
Program Add;
Var
   Num1, Num2, Sum : Integer;
Begin
     Write('Input number 1: ');
     Readln(Num1);
     Write('Input number 2: ');
     Readln(Num2);
     Sum := Num1 + Num2;
     Writeln(Sum);
     Readln;
End.
```



- Exhibits:
  - MDE



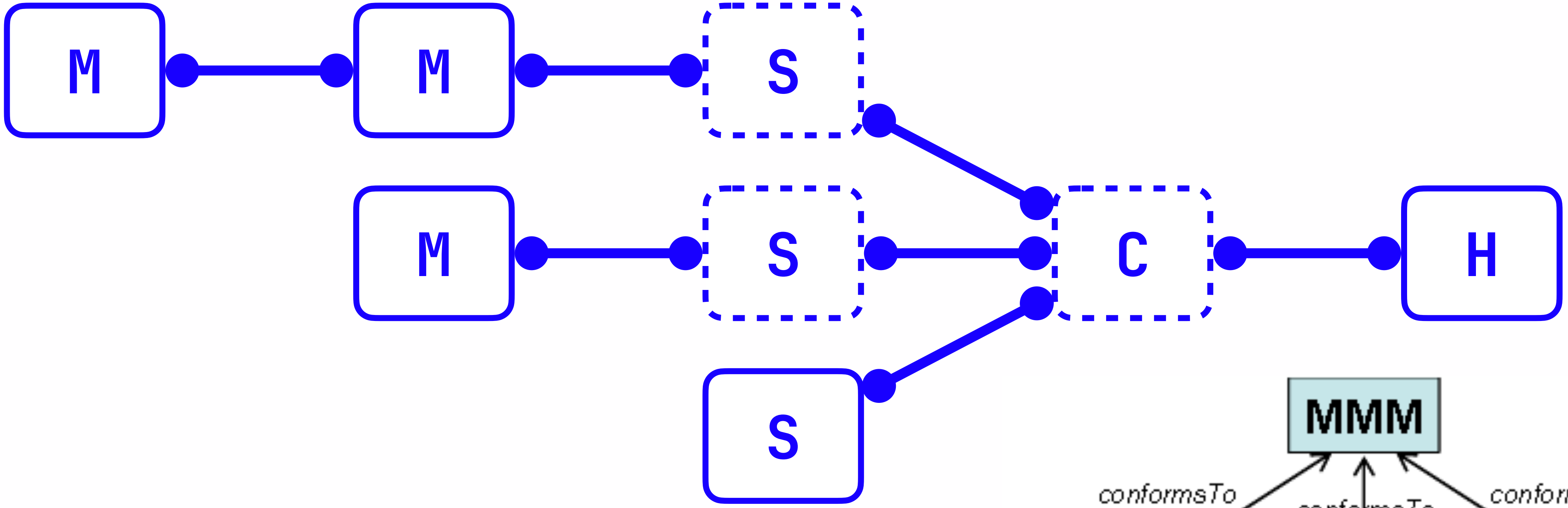
CHAPMAN & HALL/CRC INNOVATIONS IN SOFTWARE ENGINEERING AND SOFTWARE DEVELOPMENT

Engineering Modeling Languages

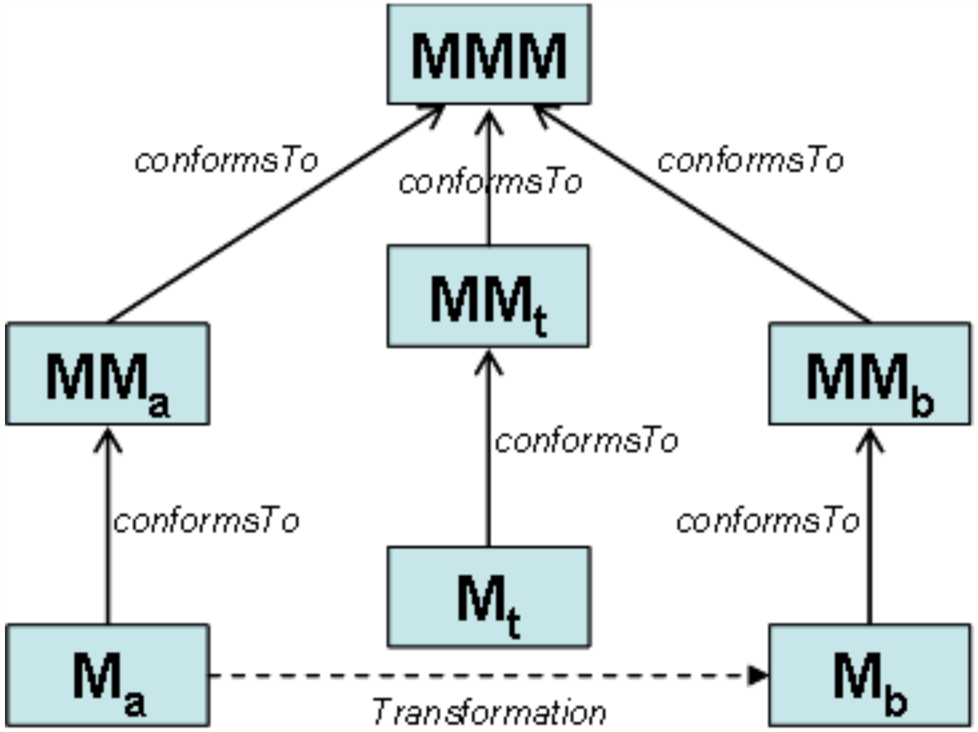
Turning Domain Knowledge into Tools

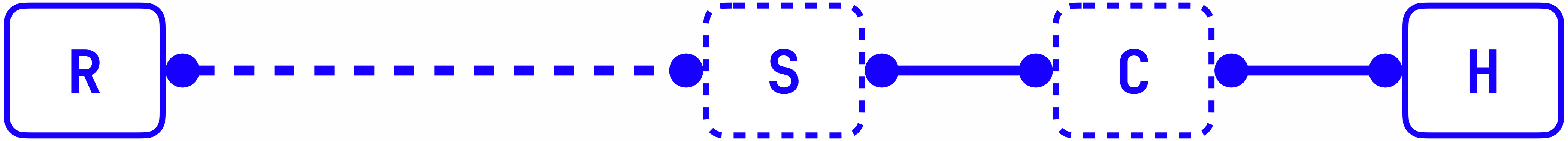
Benoit Combemale
Robert B. France
Jean-Marc Jézéquel
Bernhard Rumpe
Jim Steel
Didier Vojtisek

CRC Press
A CHAPMAN & HALL BOOK

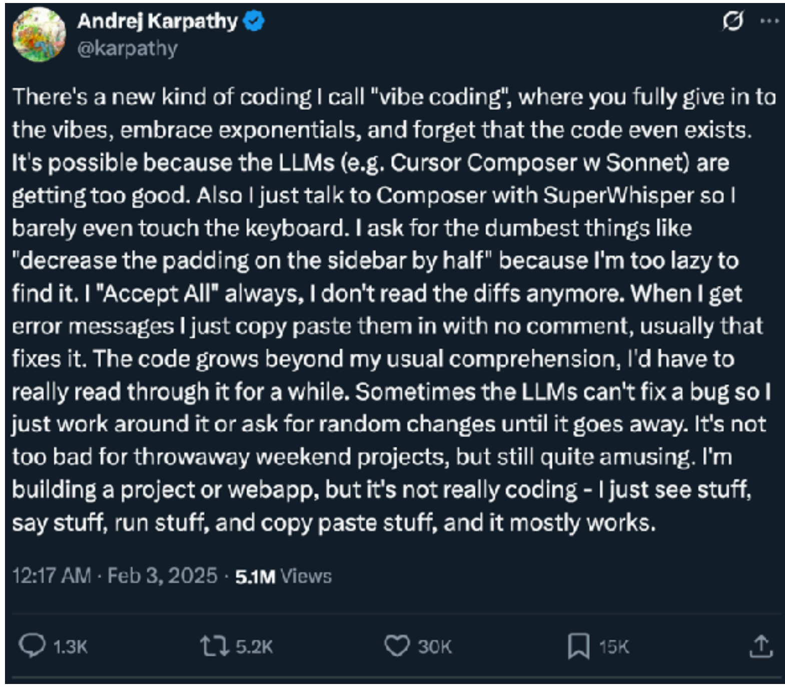


- Exhibits:
  - m2m



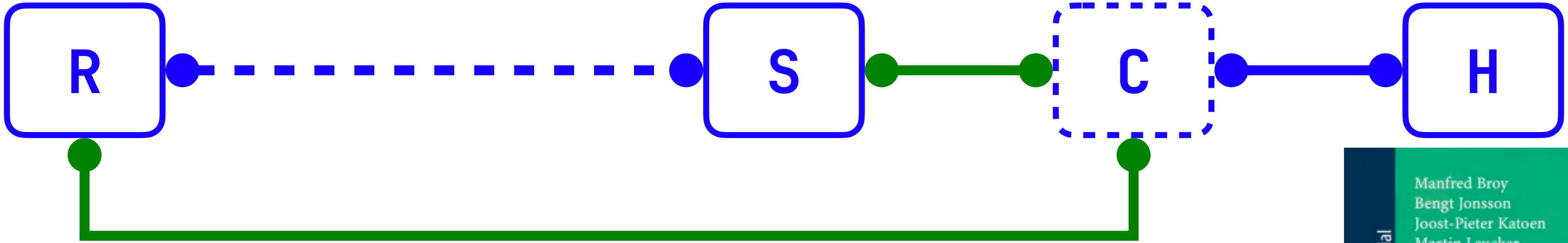


- Exhibits:
  - Karpathy 2025
  - Kruger 1999

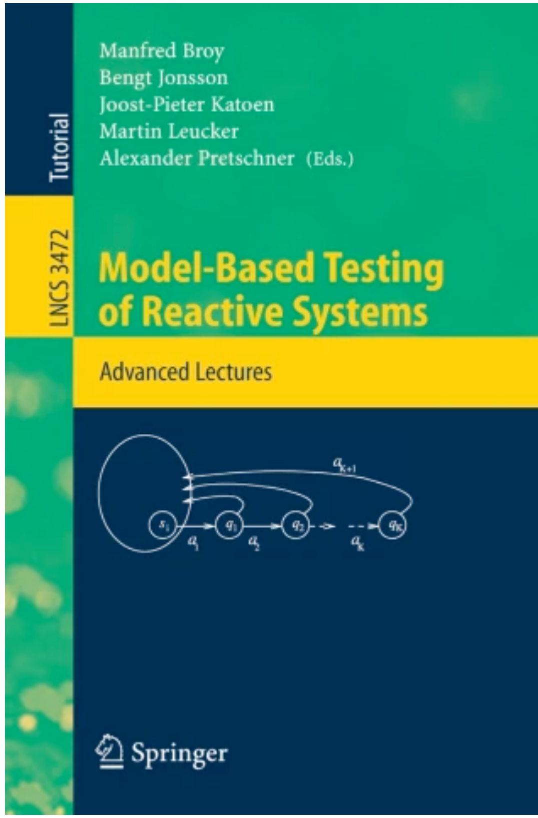


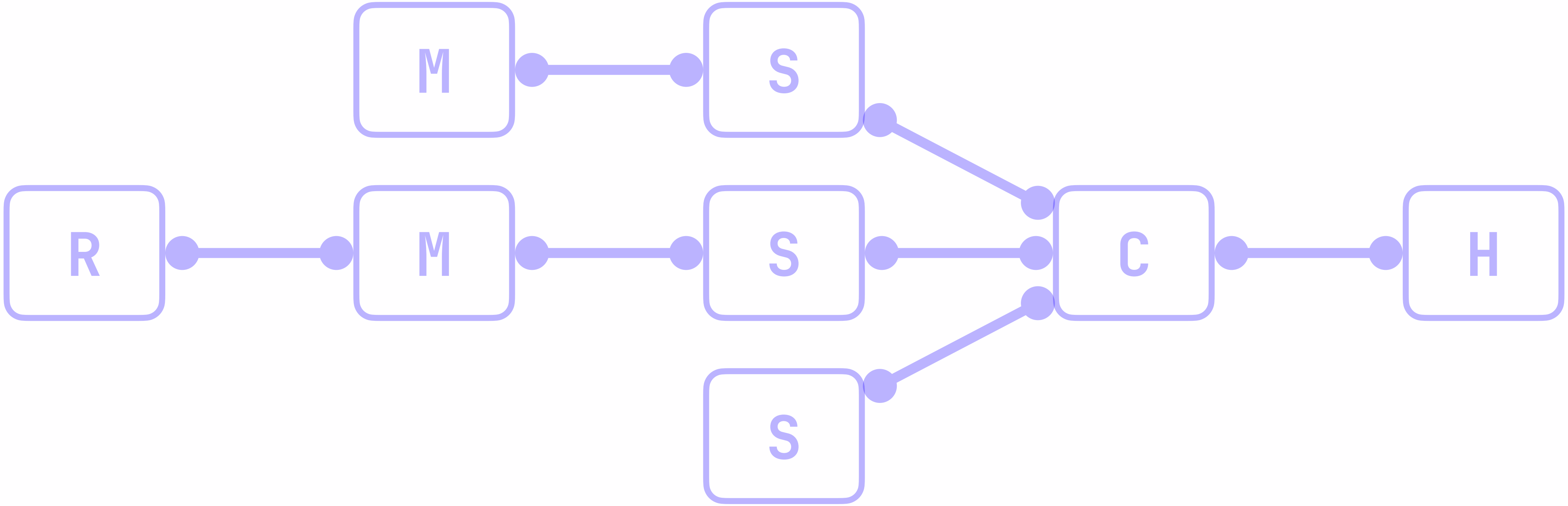
There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

12:17 AM · Feb 3, 2025 · 5.1M Views



- Exhibits:
  - Meyer 1986
  - Diehl 2000
  - Leroy & Blazy 2005+
  - Kumar et al 2012+





- Future exhibits:
  - ???