Compilers in the wild 2015

RAINCODE
COMPILER EXPERTS

About us…

3 partners
18 years
2 sites
24/31 engineers
5 continents

Looking for bright people for our technical staff in Brussels,…
Lödz,
and elsewhere

Legacy modernization

Quiz 0:
What language would you use for a system that must live for at least 30 years ?

Diversified skillset:

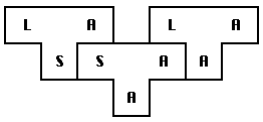
Compilation, IDE's, TP monitors, Databases, Mainframes, Unix, XML, Java, .NET

and more…

2 strategic axis: transformation and compilation

Very different approaches

In abstracto:



a →b

But maintainability vs. executability

RAINCODE

Example in transformation:

```
F05DC.
     MOVE        1                          TO ICATR.          (1)
     GO TO     F05DC-B.
 F05DC-A.
     ADD         1                          TO ICATR.          (2)
 F05DC-B.
     IF          10                         <  ICATR THEN      (3)
       GO TO     F05DC-FN
     END-IF.
     IF    CATX(ICATR) = '0' THEN                              (4)
       NEXT SENTENCE
     ELSE
       GOTO F05DC-C
     END-IF.
     MOVE 'X' TO CATM(ICATR).                                  (5)
 F05DC-C.
     IF    CATX(ICATR) NOT = '0' THEN                          (6)
       NEXT SENTENCE
     ELSE
       GOTO F05DC-D
     END-IF.
     MOVE 'Y' TO CATM(ICATR).                                  (7)
 F05DC-D.
     GO TO F05DC-A.                                            (8)
 F05DC-FN.
     EXIT.
```


RAINCODE

Resulting code:

```
PERFORM WITH TEST BEFORE                    (8)
  VARYING ICATR FROM 1                      (1)
                  BY 1                      (2)
  UNTIL 10 < ICATR                          (3)
  IF CATX(ICATR) = '0' THEN                 (4)
    MOVE 'X' TO CATM(ICATR)                 (5)
  ELSE                                      (6)
    MOVE 'Y' TO CATM(ICATR)                 (7)
  END-IF
END-PERFORM.
```

- Smaller
- Intention is clear
- No more GOTOs, no more labels
- Better performance
- 100% Safe

RAINCODE

Exemp

```
EVALUATE TRUE
  WHEN GC06-QPSTD NUMERIC AND GC06-QPSTD > ZEROES
    MOVE KB00-QSHOWP        TO 7-WORK-ATRAR
  WHEN GC06-PPOTD NUMERIC AND GC06-PPOTD > ZEROES
    IF KB00-ICKPY = 'Y' THEN
      MOVE KB00-PACT1       TO 7-WORK-ATRAR
    ELSE
      MOVE 7-FULL-PERCENT   TO 7-WORK-ATRAR
    END-IF
  WHEN GC06-ACOTD NUMERIC AND GC06-ACOTD > ZEROES
    MOVE KB70-AEDRQ         TO 7-WORK-ATRAR
END-EVALUATE
```

```
IF GC06-ACOT
   GC06-ACOT
  MOVE KB70-
END-IF

EVALUATE TRU
  WHEN GC06-
       AND G
       AND KB00-ICKPY = 'Y'
    MOVE KB00-PACT1 TO 7-WORK-ATRAR
  WHEN GC06-PPOTD NUMERIC
       AND GC06-PPOTD > ZEROES
       AND KBX0-ICKPY NOT = 'Y'
    MOVE 7-FULL-PERCENT TO 7-WORK-AT
  WHEN OTHER
    CONTINUE
END-EVALUATE

IF GC06-QPSTD NUMERIC
    AND GC06-QPSTD > ZEROES
  MOVE KB00-QSHOWP    TO 7-WORK- ATRA
END-IF
```

```
> ZEROES THEN
AR
> ZEROES THEN
AR
AR
> ZEROES THEN
AR
END-IF
```

```
IF GC06-QPSTD NUMERIC AND GC06-QPSTD > ZEROES THEN
  MOVE KB00-QSHOWP        TO 7-WORK-ATRAR
ELSE
  IF GC06-PPOTD NUMERIC AND GC06-PPOTD > ZEROES THEN
    IF KB00-ICKPY = 'Y' THEN
      MOVE KB00-PACT1       TO 7-WORK-ATRAR
    ELSE
      MOVE 7-FULL-PERCENT   TO 7-WORK-ATRAR
    END-IF
  ELSE
    IF GC06-ACOTD NUMERIC AND GC06-ACOTD > ZEROES THEN
      MOVE KB70-AEDRQ         TO 7-WORK-ATRAR
    END-IF
  END-IF
END-IF
```

http://www.raincode.com - 45 rue de la Caserne, 1000 Brussels, Belgium

RAINCODE

Legacy
throug

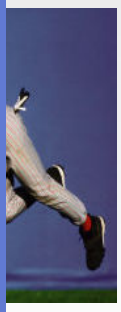
- Allow
  SQLS
  perfo
  - The c
    DB/2
  - Requi
    chang
  - A sing
    DB/2
    perio
- Allow
  predictable migration
  - No discontinuity nor disruption

Original DB/2:

```
EXEC SQL DECLARE C1 CURSOR FOR
      SELECT CONCAT(CONCAT(STRIP(UPPER(LastName),
BOTH), ', '), STRIP(FirstName, TRAILING))
      FROM People WHERE BirthDate + 30 YEARS >
:Today
      WITH UR
      FETCH FIRST ROW ONLY;
```

Converted SQLServer:

```
SELECT TOP(1)
       ((dbo.STRIP(UPPER(LastName), 'BOTH',
DEFAULT))+(', ')+(dbo.STRIP(FirstName,
'TRAILING', DEFAULT))
       FROM People WITH(READUNCOMMITTED)
       WHERE(DATEADD(YEAR, 30,
             BirthDate)>CONVERT(DATE, @V0))
```



http://www.raincode.com - 45 rue de la caserne, 1000 Brussels, Belgium – RainCode USA 13245 Atlantic Blvd., Jacksonville, FL 32225

Trends in compilation

Trend 1: specialization

1) Language designers

Tradition started with Niklaus Wirth and Stroustrup

More recently: Scala and C#

2) Targeting new hardware architectures

DSP, distributed, embedded, FPGA, etc.

GCC toolchain, now llvm retargeting (clang?)

3) Compiling existing languages for new target platforms

Legacy compilers

RAINCODE

Basic arithmetic

- *a* is a fixed bin (d,s) :

$$a = \frac{n}{2^s}$$

- If I want to convert *a* to a fixed dec(e,t), I must find *m* such that:

$$a = \frac{m}{10^t}$$

$$a = \frac{n}{2^s} \cdot \frac{5^s}{5^s} = \frac{n.5^s}{10^s} = \frac{\frac{n.5^s}{10^{s-t}}}{10^t}$$

which yields:

$$m = \frac{n.5^s}{10^{s-t}}$$

Where the division by a power of 10 is trivial (merely removing extra digits on the right)

http://www.raincode.com - 1 rue de l'Autonomie, 1070 Brussels, Belgium

RAINCODE

The problem is:

- IBM indicates that its PL/1 compiler approximates

$$n.5^s$$

- by

$$n.(5^s + 1)$$

- For some unspecified reason

It is clear that this bias is only introduced for large values of s.

On smaller values, there is no point in using approximations anyway

After a few experiments, it seems that the bias is introduced for s >= 14

Which makes sense as 14 is the smallest value of s such that:

$$\left| \log_2 5^s \right| > 31$$

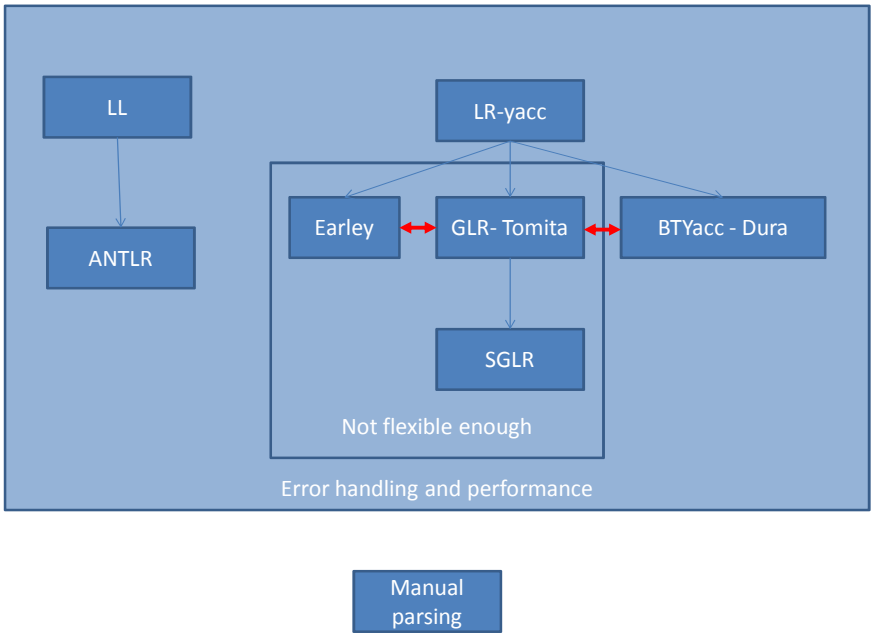
- We reproduce this (odd) behaviour scrupulously, and all our tests deliver exactly the same result as on the mainframe.

http://www.raincode.com - 1 rue de l'Autonomie, 1070 Brussels, Belgium

Trend 2: parsing techniques

Lots of research, little industrial acceptance

Observation: yacc is a problemless solution



Trend 3:
technology ≠ product

Compiler

Compiler, IDE

Compiler, IDE, Database connector

Compiler, IDE, Database connector, Debugger support

Compiler, IDE, Database connector, Debugger support, Compile-time SQL dialect conversion

Compiler, IDE, Database connector, Debugger support, Compile-time SQL dialect conversion, Interface manager

Compiler, IDE, Database connector, Debugger support, Compile-time SQL dialect conversion, Interface manager, Support infrastructure

Compiler, IDE, Database connector,
Debugger support, Compile-time SQL dialect conversion,
Interface manager, Support infrastructure,
Regression testing infrastructure

Compiler, IDE, Database connector,
Debugger support, Compile-time SQL dialect conversion, Interface manager, Support infrastructure,
Regression testing infrastructure,
Documentation

Trend 4:
reusable compiler components

Back-ends: Intermediate C code, JVM, .NET, llvm

Front-end: EDG, GCC(?)

Consequence: graph coloring is falling into oblivion

Consequence: despite its ever-increasing complexity, one can still develop analysis tools for C++ reasonably easily

Trend 5:
inference and analysis

Fuelled by hardware horsepower

…and new language designs

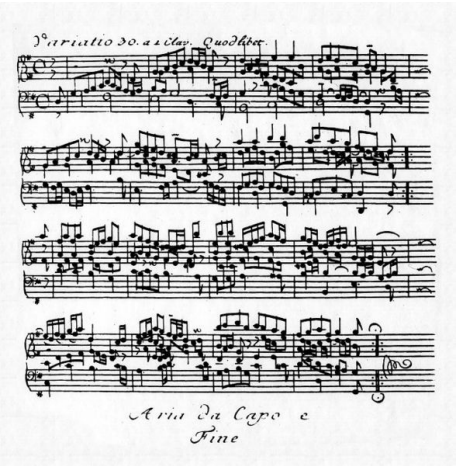
Global compilation and optimisation

C (Performance)
Eiffel (Foot print)
Scala (Type inference)
PHP (Reduce dynamism)

Trend 6: DSL

S

Lexical variations



Quiz 1:
What is the worst case for lexical analysis based on Unix's lex ?

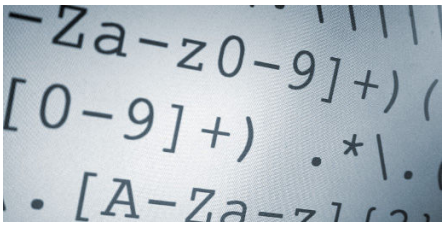
A -> a*b
->a

Input: a^n

Quiz 2:
How to distinguish between commented code and “real” comments?

n-grams

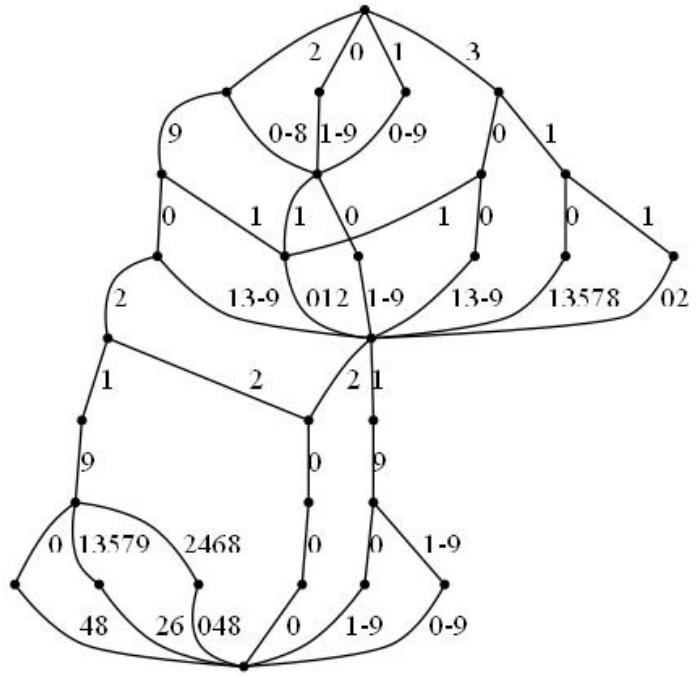
Quiz 3:
Right to left lexical analysis?

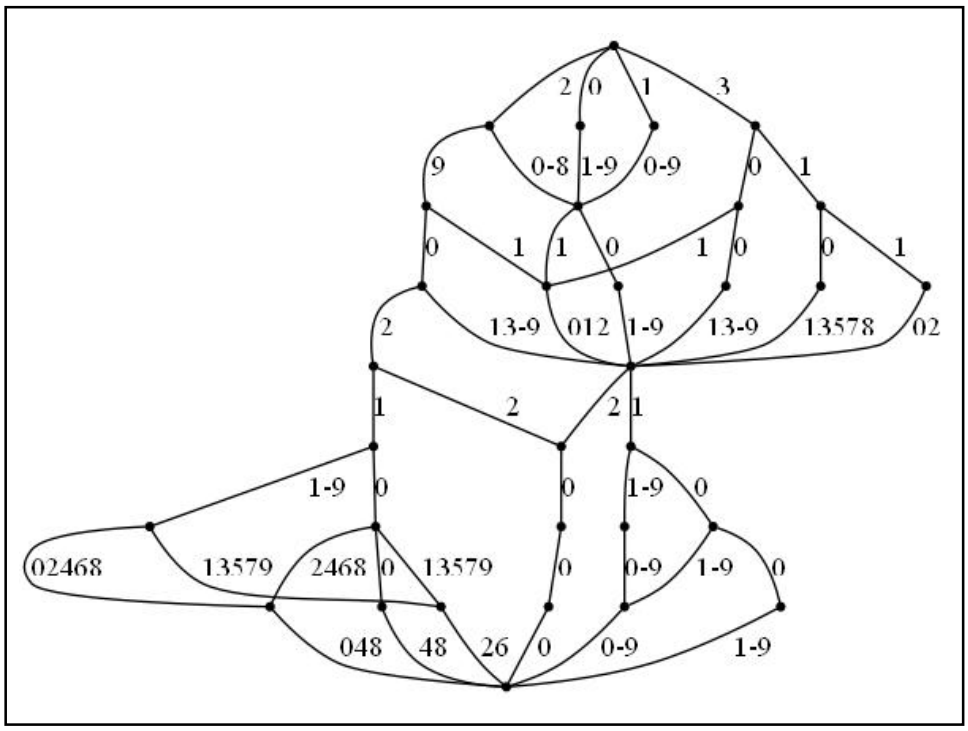


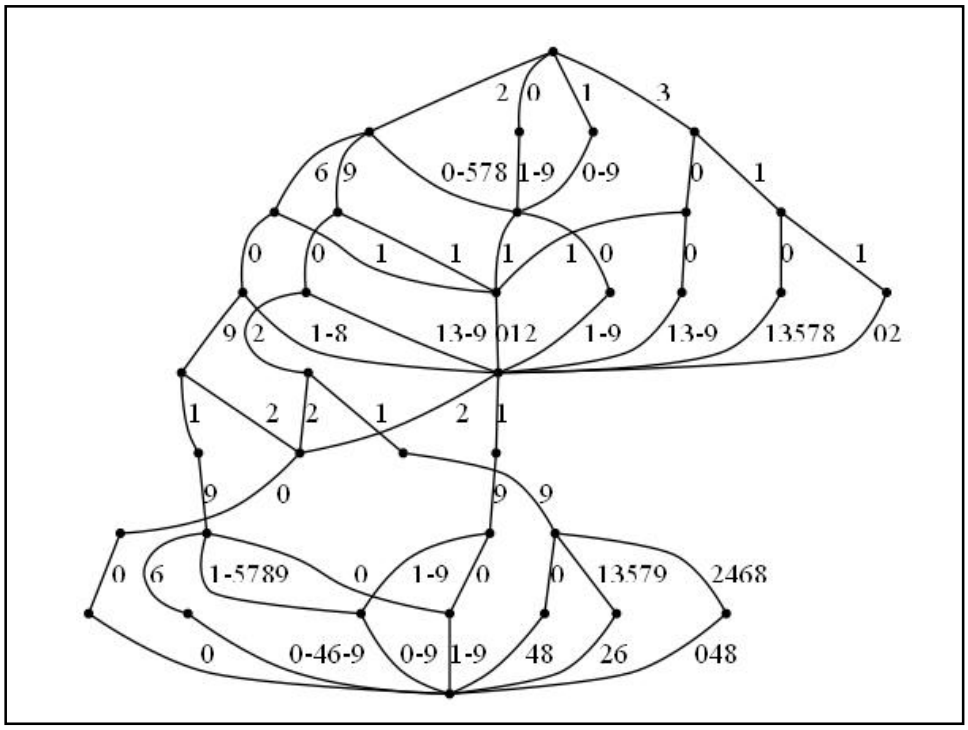
Theoretical reversibility does not imply equivalent performances

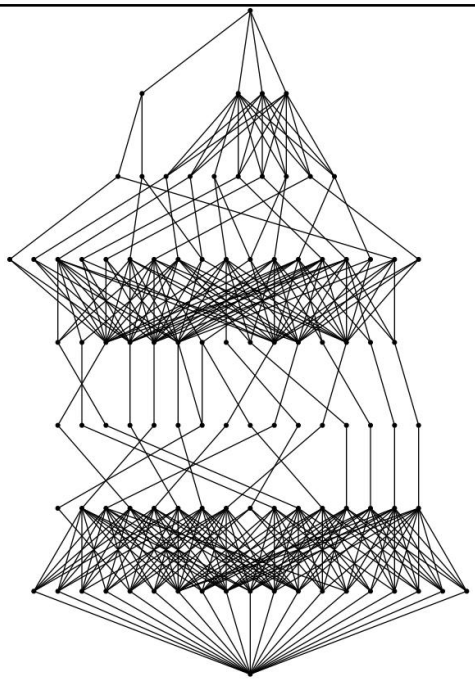
Quiz 4:
Why are pumping lemmas important?

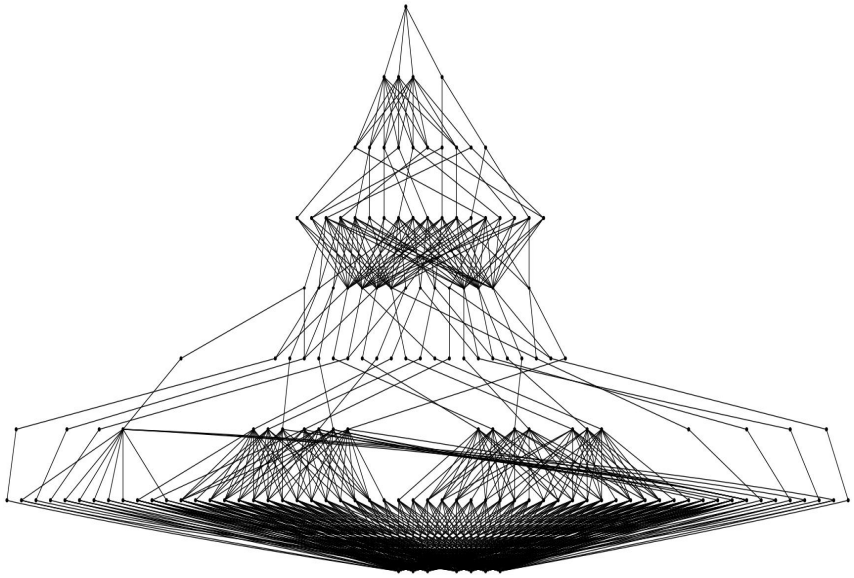
Quiz 5:
How to check for a valid date of the 20th or 21st century?




2 0 1 3
9 0-8 1-9 0-9 0 1
0 1 1 0 1 0 0 1
2 13-9 012 1-9 13-9 13578 02
1 2 2 1
1-9 0 0 1-9 0
02468 13579 2468 0 13579 0 0-9 1-9 0
048 48 26 0 0-9 1-9


2 0 1 3
6 9 0-578 1-9 0-9 0 1
0 0 1 1 1 1 0 0 0 1
9 2 1-8 13-9 012 1-9 13-9 13578 02
1 2 2 1 2 1
9 0 9 9
0 6 1-5789 0 1-9 0 0 13579 2468
0 0-46-9 0-9 1-9 48 26 048





(((0[1-9])|(1[0-9]))((0[1-9])|(1[012]))((19((0[1-9])|([1-9][0-9])))|(2000)))|(2((9((0((2((19((0[48])|([13579][26])|([2468][048])))|(2000)))|([13-9]((19((0[1-9])|([1-9][0-9])))|(2000)))))|(1[012]((19((0[1-9])|([1-9][0-9])))|(2000)))))|([0-8]((0[1-9])|(1[012]))((19((0[1-9])|([1-9][0-9])))|(2000)))))|(3((0((0[13-9])|(1[012])))|(1((0[13578])|(1[02])))))((19((0[1-9])|([1-9][0-9])))|(2000)))

Theoretical equivalence
is just that.

Theoretical.

Contribution:
lexical conjunction

(abc?)+&.{35-40}


Thank
You!