.

From natural language processing to Prolog

Alain Colmerauer

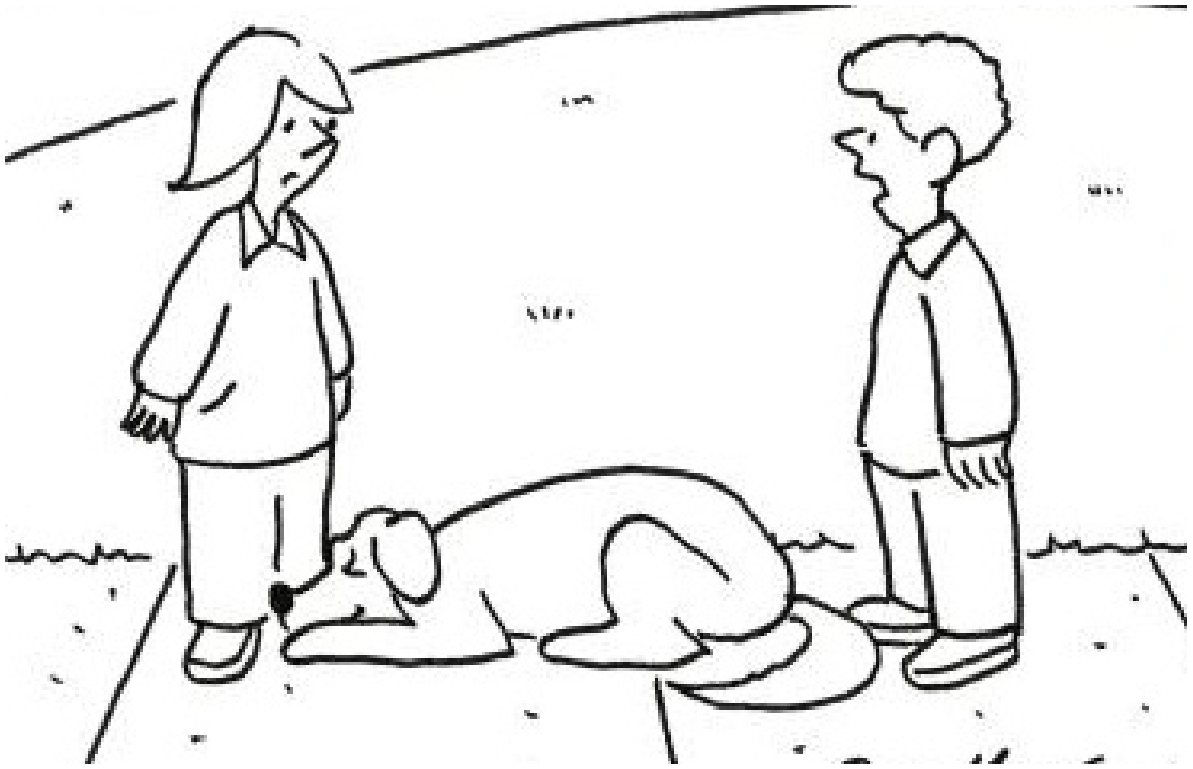
Beijing, April 8, 2011

Marseille, France, http://alain.colmerauer.free.fr

Contents

.

1st experience of automatic translation, 45 years ago: W-Grammar, 1967–1968, University of Montreal, Canada



He can ignore you in seven different languages!

W-grammar of $a^n b^n c^n$ genererated from n

• **Grammar**

```
N --> N a , N b , N c

one SYMBOL --> SYMBOL

N plus one SYMBOL --> N SYMBOL , SYMBOL
```

• **Meta-grammar**

```
N -> one
N -> N plus one

SYMBOL -> a
SYMBOL -> b
SYMBOL -> c
```

Van Wijngaarden, *Final Draft Report on the Algorithm Language Algol 68,* Amsterdam Mathematish Centrum, December 1968.

From English to French

- **Input**

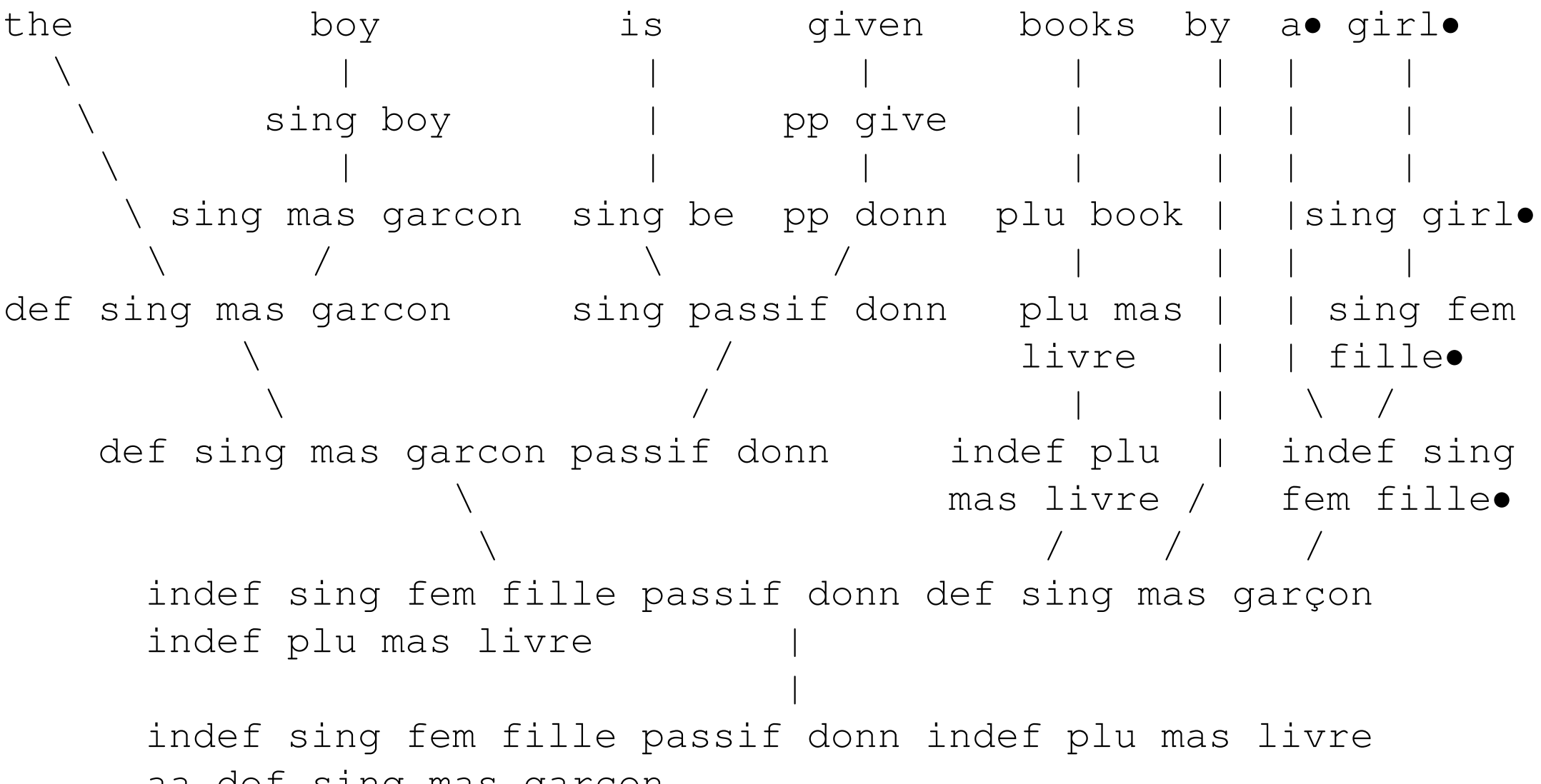
```
the , boy , is , given , books , by , a , girl
```

- **Output**

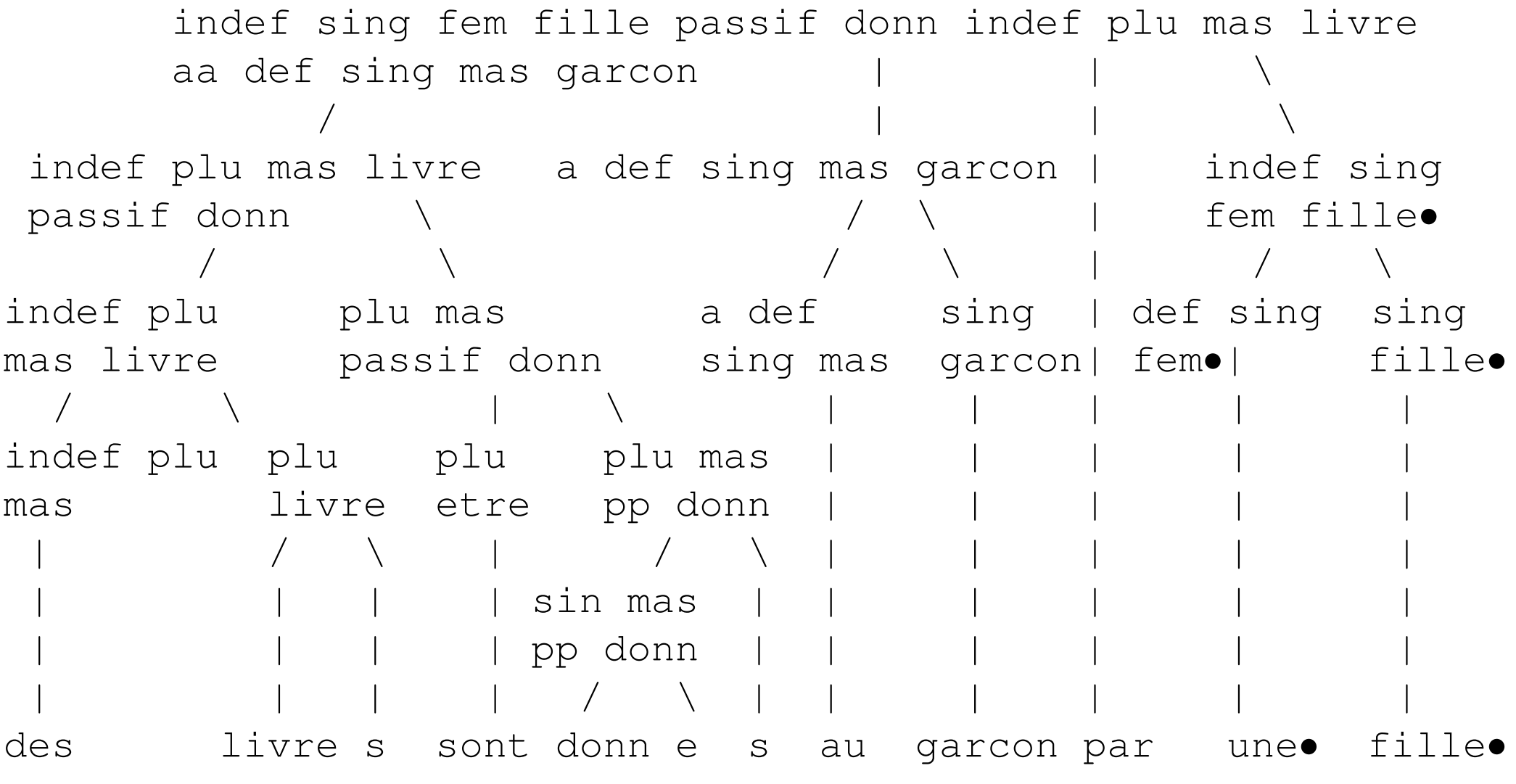
```
des , livre s , sont , donn e s , au , garcon , par , une , fille
```

Guy de Chastellier and Alain Colmerauer, W-Grammar, *Proceedings of the 24th national conference,* August, ACM, New York, page 511-518, 1969.

From English to the pivot langage



From the pivot langage to the French



English Grammar

```
SN MODE V SN1 aa SN2 -->
   SN MODE V SN2 SN1

SN1 actif V SN2 CO -->
   SN1 actif V , SN2 , CO

SN1 passif V SN2 CO -->
   SN2 passif V , CO , by , SN1

ART NOMBRE NOM MODE V -->
   ART NOMBRE NOM ,
   NOMBRE MODE V

aa SN --> to , SN

NOMBRE passif V -->
   NOMBRE be , pp V
FORME donn --> FORME give
```

```
sing actif give --> gives
plu actif give --> give
pp give --> given
sing be --> is
plu be --> are

def NOMBRE NOM -->
   the , NOMBRE NOM
indef sing NOM --> a , sing NOM •
indef plu NOM --> plu NOM

NOMBRE mas livre --> NOMBRE book
NOMBRE mas garcon --> NOMBRE boy
NOMBRE fem fille --> NOMBRE girl •
sing book --> book
plu book --> books
sing boy --> boy
plu boy --> boys
sing girl --> girl •
plu girl --> girls
```

English Meta-Grammar

```
ART -> def
ART -> indef

CO -> SN
CO -> aa SN

FORME -> pp
FORME -> NOMBRE actif

GENRE -> mas
GENRE -> fem •

MODE -> actif
MODE -> passif

N -> garcon
N -> fille •
N -> livre

NOM -> GENRE N •

NOMBRE -> sing •
NOMBRE -> plu

P -> SN SV

SN -> ART NOMBRE NOM

SN1 -> SN

SN2 -> SN

V -> donn
```

French Grammar

```
SN actif V SN1 aa SN2 -->
   SN actif V ,
   SN1 ,
   aa SN2
SN passif V SN1 aa SN2 -->
   SN1 passif V ,
   aa SN2 ,
   par ,
   SN
ART NOMBRE GENRE N MODE V -->
   ART NOMBRE GENRE N ,
   NOMBRE GENRE MODE V
AART NOMBRE GENRE N -->
   AART NOMBRE GENRE ,
   NOMBRE N •
def sing mas --> le
def sing fem --> la
def plu GENRE --> les
aa def sing mas --> au
aa BONART --> a , BONART
indef sing mas --> un
indef sing fem --> une •
indef plu GENRE --> des
aa def plu GENRE --> aux
NOMBRE GENRE passif V -->
   NOMBRE etre ,
   NOMBRE GENRE pp V
plu GENRE pp V -->
   sing GENRE pp V , s
sing fem pp V -->
   sing mas pp V , e
sing mas pp V --> V , e
sing GENRE actif V --> V e
plu GENRE actif V --> V ent
sing etre --> est
plu etre --> sont
sing N --> N
plu N --> N s
```

French Meta-Grammar

```
AART -> ART •
AART -> aa ART

ART -> def
ART -> indef •

BONART -> def sing fem
BONART -> indef NOMBRE GENRE

GENRE -> mas
GENRE -> fem

N -> garcon
N -> fille •
N -> livre

NOM -> GENRE N

MODE -> actif
MODE -> passif

NOMBRE -> sing •
NOMBRE -> plu

P -> SN SV

SV -> MODE V SN aa SN

SN -> ART NOMBRE NOM
SN1 -> SN

SN2 -> SN

V -> donn
```

.

2nd experience of automatic translation: Q-System, 1968–1970, University of Montreal, Canada

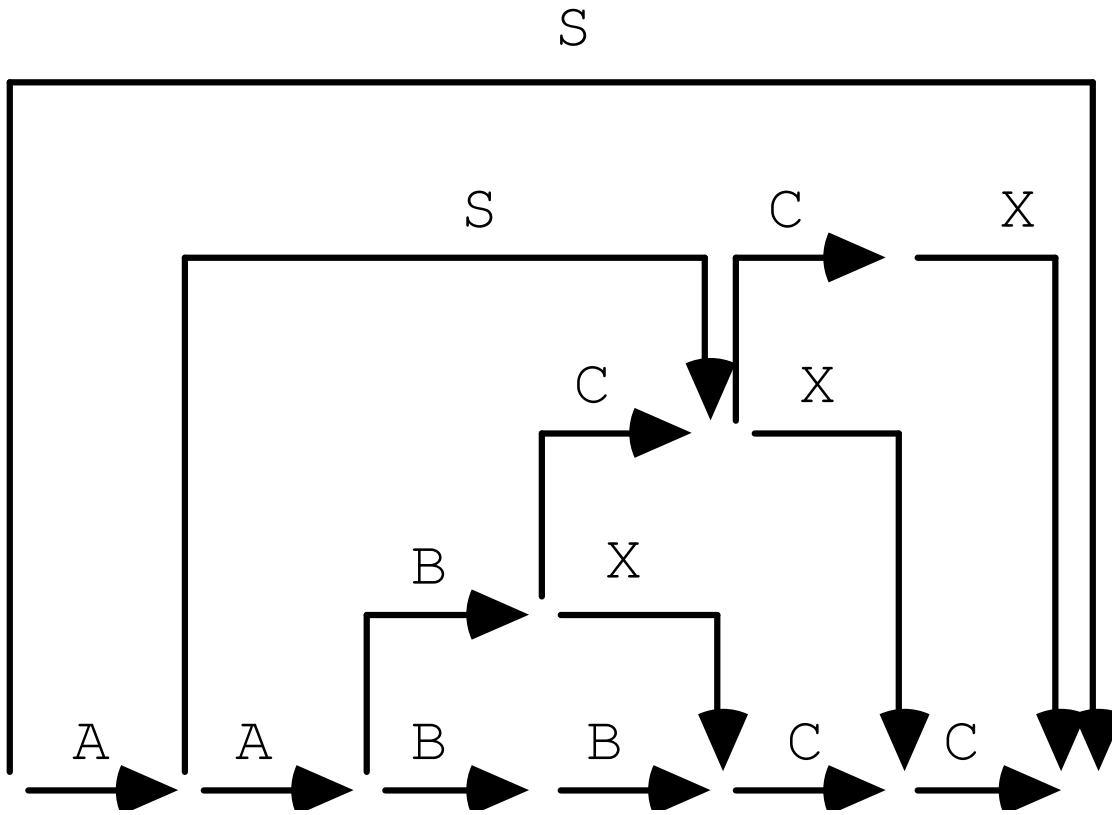
Q-System for $A^n B^n C^n \Rightarrow S(n \text{ times } 1)$

```
A(U*) + B(U*) + C(U*) == S(1,U*).

A + A(U*) == A(1,U*).

B + B(U*) == B(1,U*).

C + C(U*) == C(1,U*).
```

Alain Colmerauer, *Les systèmes-q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur,* publication interne no 43, septembre 1970, Département d'informatique, Université de Montréal, republié dans *T.A.L,* 1992, no 1-2, p. 105-148.

Q-system for $A^n B^n C^n \Rightarrow S$, variant

```
A + B + C == S.

A + S + X + C == S.

X + C == C + X.

B + B == B + X.
```


S
S
C
X
C
X
B
X
A
A
B
B
C
C

Automatic Translation, University of Montreal, example, 1973

Input

```
THE EXPANSION OF GOVERNEMENT ACTIVITIES IN CANADA AS IN MANY
OTHER COUNTRIES IS NOT SOMETHING NEW.
```

Output

```
L'EXPANSION DES ACTIVITES GOUVERNEMENTALES AU CANADA COMME DANS
PLUSIEURS AUTRES PAYS N'EST PAS QUELQUE CHOSE DE NOUVEAU.
```

Richard Kittredge, linguist for the English part. **Gilles Stewart**, computer scientist for the transfer part. **Jules Dansereau**, linguist for the French part.

In 16 steps

Step 0

```
-1- $(1) + THE + EXPANSION + OF + GOVERNEMENT + ACTIVITIES +
IN + CANADA + AS + IN + MANY + OTHER + COUNTRIES + IS + NOT
+ SOMETHING + NEW + . -2-
```

Step 1

```
-1- $(1) + ART(DEF) + EXPANSION + P(OF) + GOVERNMENT +
ACTIVITIES + P(IN) + NP(N(CANADA),/,*C,*PROP,*NT) -2-
-2- SCON(AS) -3-
-2- P(AS) -3-
-3- P(IN) + QUANT(MANY) + OTHER + COUNTRIES + T(PRS3S) + *(BE)
+ *(NOT) + NP(N(SOMETHING),/,*C,*AB,*INDF) + NEW + *(.) -4-
```

Step 2

Step 3

Step 4

Step 5

```
-1- $(1) + ART(DEF) + N(EXPANSION,/,*AB,*DV) + P(OF) +
N(GOVERNMENT,/,*H,*AB,*GP) + N(ACTIVITY,/,*AB) + S +
P(IN) + NP(N(CANADA),/,*C,*PROP,*NT) -2-
-2- SCON(AS) -3-
-2- P(AS) -3-
-3- P(IN) + QUANT(MANY) + ADJ(OTHER,/) +
N(COUNTRIES,/,*H,*C,*GP) + S + T(PRS3S) + *(BE) + *(NOT) +
NP(N(SOMETHING),/,*C,*AB,*INDF) + ADJ(NEW,/,) + *(.) -4-
```

Step 6

```
-1- $(1) -2-
-2- NP(N(EXPANSION),DET(ART(DEF)),GV(P(OF),
NP(N(GOVERNMENT,/,*H,*AB,*GP)),/,*AB,*DV) +
NP(N(ACTIVITY),/,*AB,*PL) -3-
-2- NP(N(EXPANSION),DET(ART(DEF)),GV(P(OF),NP(N(ACTIVITY),
 ADJ(GOVERNMENT3,/),/,*AB,*PL)),/,*AB,*DV) -3-
-3- P(IN) + NP(N(CANADA),/,*C,*PROP,*NT) -4-
-4- SCON(AS) -5-
-4- P(AS) -5-
-5- P(IN) +
NP(N(COUNTRY),DET(CARD(QUANT(MANY))),ADJ(OTHER,/)/,*H,*C,*GP) +
T(PRS3S) + BE + NOT +
NP(N(SOMETHING),ADJ(NEW,/,)/,*C,*AB,*INDF) + . -6-
```

Step 7

```
-1- SENTENCE(PH(GOV(T(PRS3S),OPS(INV(NOT)),NP(N(SOMETHING),
ADJ(NEW,/),/,*C,*AB,*INDF)),NP1(N(EXPANSION),DET(AET(DEF)),
GP(P(OF),NP(N(ACTIVITY),ADJ(GOVERNMENT3,/),GP(P(IN),
NP(CONJ(LIKE),NP(N(CANADA),/,*C,*PROP,*NI,*LOC),NP(N(COUNTRY),
DET(CARD(QUANT(MANY))),ADJ(OTHER,/),/,*H,*C,*GP,*PL,*LOC),/,
*C,*LOC,*LOC)),/,*AB,*PL)),/,*AB,*DV),/)) -2-
```

Step 8

```
-1- [(PH) + [(GOV) + [(T) + *IPR + [(OPS) + [(INV) + NE + PAS +
 ] + * + ] +      [(SN) + [(N) + /N(*C,*AB,*INDF) + QUELQUE +
CHOSE + ] + [ (ADJ) + /A(NP1(*C,*AB,*INDF)) + NOUVEAU(*ANTE) +
/ +] + ! + / + *C + *AB + INDF + ] + ! ] + ! + P(SUJ) + [(SN) +
[(N) + /N(*AB,*DV) + EXPANSION(*F) + ] + [(DET) + ART(DEF) + ]
+ [(GP) + [(P) +  /P(*AB,*PL) + DE + ] + ACTIVITE(*F) + ] +
[(ADJ) + /A(NP1(*AB,*PL)) + GOUVERNEMENTAL + / + ] + ! + [(GP)
+ [(P) + /P(*C,*LOC,*LOB) + DANS + ] + [(SN) + [(CONJ) + COMME
+ ] + [(SN) + [(N) + /N(*C,*PROP,*NT,*LOC) + CANADA(DEF) + ] +
/ + *C + *PROP + *NT + *LOC +  + ! + [(SN) + (N) + /N(*H,*C,*GP,
*PL,*LOC) PAYS + ] + [(DET) + [(CARD) + [(QUANT) + PLUSIEURS + ]
+ ] + ] + [(ADJ) + /A(NP1(*H,*C,*GP, + *PL + LOC )) + AUTRE(ANTE)
+ ] + ! + / + *H + *C + *GP, + *PL + LOC + ] + ! + / + *C + *LOC
+ *LOC + ] + ! +/ + *AB + *PL + ] + ! + ] + / + *AB + * + ] + !
+ NO(1) + ! + / + ] + . -2-
```

But this is fantastic! It’s like no language I’ve ever seen before!

Step 9

```
-1- PH(GOV(T(*IPR),OPS(INV(NE,PAS)),SN(N(QUELQUE,CHOSE),
ADJ(NOUVEAU,/,*ANTE),/,*C,*AB,*INDF,*M,*S,3)),P(SUJ),
SN(N(EXPANSION),DET(ART(DEF)),GP(P(DE),SN(N(ACTIVITE),
ADJ(GOUVERNEMENTAL,/),GP(P(DANS),SN(CONJ(COMME),SN(N(CANADA),
DET(ART(DEF)),/,*C,PROP,*NT,*LOC,*M,*S,3),SN(N(PAYS),
DET(CARD(QUANT(PLUSIEURS))),ADJ(AUTRE,/,*ANTE),/,*H,*C,*GP,*PL,
*LOC,*M,3),/,*H,*C,*GP,*PL,*LOC,*M,3)),/,*F,*AB,*PL,3)),/,*F,
*AB,*S,3),/) + . -2-
```

Step 10

```
-1- ELI(LA) + NOM(EXPANSION,/,*F,*AB,*S,3) + ELI(DE) + LEX(LES)
+ NOM(ACTIVITE,/,*F,*AB,*PL,3) +
ADJ5(GOUVERNEMENTLAL,/,$,*F,*AB,*PL,3)
+ LEX(A) + ELI(LE) + LEX(CANADA) + LEX(COMME) + LEX(DANS) +
LEX(PLUSIEURS) + ADJ5(AUTRE,/,*ANTE,$,*H,*C,*GP,*PL,*LOC,*M,3) +
NOM(PAYS,/,*H,*C,*GP,*PL,*LOC,*M,3) + ELI(NE) + RAD(F,T,R,E) +
P(3,*S) + DEC(*F,*S) + T(*IPR) + LEX(PAS) + LEX(QUELQUE) +
LEX(CHOSE) + ELI(DE) + TOF(NOUVEAU) + LEX(.) -2-
```

Step 11

Step 12

Step 13

Step 14

Step 15

Step 16

-1- L + ’ + EXPANSION + DES + ACTIVITES + GOUVERNEMENTALES + AU + CANADA + COMME + DANS + PLUSIEURS + AUTRES + PAYS + N + ’ + EST + PAS + QUELQUE + CHOSE + DE + NOUVEAU + . -2-

Some Q-Systems rules

```
AIRPORT == N(AIRPORT,/,*C,*LOC).


NP(U*,/,V*) + REL(W*) ==
   RNP(U*,/,V*) + NP(U*,/,V*,REL) /
   -NON- V* -HORS- W* -ET- GP,PH -HORS- U*.


ARRIVE == ARRIVER(*E,*PA).


A + V + - + A* == A + I + - + A* / A* -DANS- S,T,X.
```

.

Inference in French, 1971–1973, University of Marseille, France

Example 1, translated from French

```
INPUT TEXT:

EVERY PSYCHIATRIST IS A PERSON.
EVERY PERSON HE ANALYSES IS SICK.
*JACQUES IS A PSYCHIATRIST IN MARSEILLE.
IS *JACQUES A PERSON?
WHERE IS JACQUES?
IS JACQUES SICK?

ANSWER:

+SORT(YES) ;.
+SORT(IN MARSEILLE) ;.
+SORT(I DON'T KNOW) ;.
```

Alain Colmerauer, Henry Kanoui, Robert Pasero and Philippe Roussel. *Un système de communication en français*, preliminary report, Groupe de Recherche en Intelligence Artificielle, University II Aix-Marseille, October 1972.

Example 2, translated from French

```
INPUT TEXT:

*KAYSER WORKS IN *PARIS.
EVERY PERSONNE WHO WORKS IN PARIS, TAKES THE UNDERGROUND.
*KAYSER IS A PERSON.
WHAT DOES *KAYSER TAKE?

ANSWER:

+SORT(THE UNDERGROUND) ;.
```

Example 3, translated from French

```
INPUT TEXT:

THE ICE IS A SOLID.
EVERY SOLID WHICH MELTS, IS A LIQUID.
*KAYSER DRINKS ICE WHICH MELTS.
WHO DRINKS A LIQUID?
DOES *KAYSER DRINKS A LIQUID?

ANSWER:

+SORT(*KAYSER) ;.
+SORT(YES) ;.
```

Example 4, translated from French

```
INPUT TEXT:

THE *MONDE INFORMS EVERY FELLOW WHO READS IT.
THE *PRESIDENT IS A FELLOW.
WHO READS THE *MONDE?
WHOM DOES INFORM THE *MONDE?

ANSWER:

+SORT(THE*PRESIDENT) ;.
+SORT(THE*PRESIDENT) ;.
```

Example 5, translated from French

```
INPUT TEXT:

EVERY MAN WHO IS A BACHELOR, CAN ASK FOR  MARRIAGE EVERY
WOMAN WHO IS A BACHELOR.
EVERY MAN WHO IS RICH, CAN MARY EVERY WOMAN WHOM HE
CAN ASK FOR MARRIAGE.
EVERY MAN WHO CAN MARRY THE WOMAN WHOM HE WANTS TO MARRY
MARRIES HER.
*MARTIN, WHO IS RICH, IS A MAN WHO IS A BACHELOR.
*LEONIE IS A WOMAN WHO IS A BACHELOR.
*MARTIN WANTS TO MARRY *LEONIE.
DOES *MARTIN MARRY *LEONIE ?

ANSWER:

+SORT(YES) ;.
```

French subset

- A text made from several affirmative and interrogative sentences.
- The pronouns are allowed.
- All the common and proper nouns are allowed.
- All the verbs at the third person of present, singular and plural, are allowed.
- A lot of prepositions.
- A lot of articles.
- The negation is allowed.
- The relative clauses are allowed.

Types of inference

- Link between the singular and plural common nouns.

- Link between the same verb at the 3rd person singular and the 3rd person plural.

- The verb “to be” is treated as equality.

- Semantics of each article.

The Prolog program

• **Parser, $\approx$ 250 clauses**
Parsing a text T_0 and generating a deep structure T_1.

• **Logical form generator, $\approx$ 80 clauses**
To find the antecedent of pronoun in T_1 and to generate a logical form T_2.

• **Elementary statements generator, $\approx$ 130 clauses**
Decompose the logical formula T_2 into a set of elementary statements T_3.

• **Inference engine, $\approx$ 90 clauses**
Infer from T_3 the answers T_4.

Open a parenthesis : Context-free with attributes

• To replace the Q-Sytems for parsing the French, I used a technique which shall be the bases of the implementation of the Metamorphis Grammars, see later.

• For parsing $b_1 b_2 ... b_p$ and obtain S, if the initial symbol is s, we execute:

```
:- s([b1,b2,...,bp],[],S).
```

• For each context-free rule $a_0 \rightarrow a_1 a_2 \ldots a_n$ we generate the Prolog program:

```
a0(X0,Xn,S0) :- a1(X0,X1,S1),
                a2(X1,X2,S2),
                ...
                an(Xn-1,Xn,Sn),
                compose(S0,S1,S2,...,Sn).
```

• For each symbol ai which is terminal we add:

```
ai([ai|X],X,Si) :- attribut(Si).
```

Close the parenthesis

Extracts of Prolog progams

Parser

```
** SYNTAGME PREPOSITIONNEL ..
+SP(*X.*A,(*X.NIL).Y,*B) -PREP(*X) -SN(*A,*Y,*B) .;
** MORPHOLOGIE DU NOM ..
+PLUR(*X-E-U,*X-E-U-X) ..
** MORPHOLOGIE DU VERBE ..
+COUPLE(*X-Q-U-I-E-R-T,X-Q-U-I-E-R-E-N-T) ..
```

Logical form generator

```
+DANS(*O,MILIEU,*X,DIESE.*Y) -DANS(MILIEU,MILIEU,*X,*Y) ..
```

Elementary statements generator

```
+TPHRASE(*I,*J,*X,*L,
   P(*Q,*L,OUI((NIL-E-S-T.NIL-S-O-N-T).NIL)).*Z/*W,*K)
   -TSP(NIL,*L,P(*Q,*L,OUI((NIL-E-S-T.NIL-S-O-N-T).NIL)),*N)
   -CONC(*N,*Z,*M)
   -TPARAGRAPHE(*I,*J,*X,*M/*W,*K) .;
```

Inference engine

```
+Q(*X,*Y,*I) -DAF(*I,3) -DAF(*I,3) -DAF(*I,3) -EG(*X,*Y,2) ;.
```

.

First Prolog, 1973–1976, University of Marseille, France

Prolog 0

• Philippe Roussel found the name of Prolog and implemented it in Algol-W on an IBM 360-67, 1972. He used a teletypewriter connected by a telephone line from Marseille to the computer center of the University Grenoble.

• Syntax example

```
+APPEND(NIL,*Y,*Y);
+APPEND(dot(*A,*X),*Y,dot(*A,*Z) -APPEND(*X,*Y,*Z);
```

• No "cut", but at the end of the clause

- `..` performs a cut after the head of the clause,
- `.;` performs a cut after the execution of all the clause,
- `;;` is for a normal clause.

• Evaluable predicates limited.

Prolog I

• A little later Philipe and myself improved Prolog and assisted Gérard Battani, Henry Meloni and René Bazzoli in implementing it in Fortran and Prolog.

• Introduction of the cut.

• The evaluable predicates are chosen in a way to write a supervizer in Prolog. To change the syntax it is sufficient to change the supervizer.

• The "Grammaires de métamorphose", "Definite clause grammar", as David Warren would call them, are included.

• A lot of people visited us and took a copy with them: among them was David Warren. This Fortran version was higly distributed around the world.

Gérard Battani et Henry Meloni, *Interpréteur du langage de programmation,* Rapport de DEA, Groupe d'Intelligence Artificielle, Université de Marseille, 1973.

Philippe Roussel, *Prolog Manuel de référence et d'utilisation,* Groupe de recherche en Intelligence Artificielle, Marseille, 1975.

Alain Colmerauer, Metamorphosis grammars, in *Natural Language Communication with Computers,* Lectures Notes in Computer Science n 63, édité par L. Bolc, Springer Verlag, 1978.

.

For those who do not know Prolog very well

The progam about brother and sister

- **Progam**

```
brothersister(X,Z) :- issonof(X,Y), isdaughterof(Z,Y).

isdaughterof(helen,judith).
isdaughterof(nicole,mary).
isdaughterof(rachel,mary).

issonof(jules,judith).
issonof(paul,judith).
issonof(robert,mary).
```

- **First execution**

```
> brothersister(paul,X)?
X=helen.
```

- **Second execution**

```
> brothersister(X,helen)?
X=jules.
X=paul.
```

First execution, details

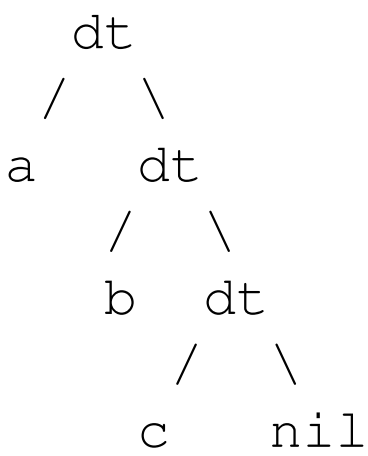
```
> brothersister(paul,X)?
brothersister(X1,Z1) :- issonof(X1,Y1), isdaughterof(Z1,Y1).
X1=paul, X=Z1

    > issonof(paul,Y1), isdaughterof(Z1,Y1)?
    issonof(paul,judith).
    Y1=judith.

        > isdaughterof(Z1,judith)?
        isdaughterof(helen,judith).
        Z1=helen
        X=helen
```

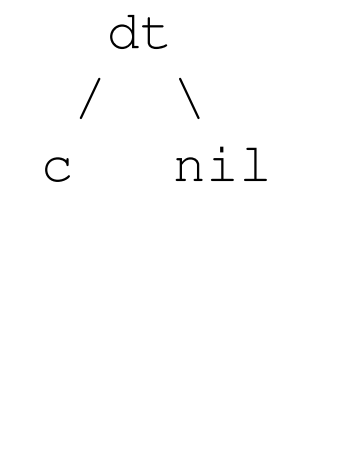
Second execution, details

```
> brothersister(X,helen)?
brothersister(X1,Z1) :- issonof(X1,Y1), isdaughterof(Z1,Y1).
X=X1, Z1=helen
   > issonof(X1,Y1), isdaughterof(helen,Y1)?
   issonof(jule,judith).
   X1=jule, Y1=judith
      > isdaughterof(helen,judith)?
      isdaughterof(helen,judith).
      X=jule
   issonof(paul,judith).
   X1=paul, Y1=judith
      > isdaughterof(helen,judith)?
      isdaughterof(helen,judith).
      X=paul
   issonof(robert,mary).
   X1=robert, Y1=mary
      > isdaughterof(helen,judith)?
      isdaughterof(helen,mary).
```

"What I can’t understand is why my brother has two sisters but I only have one..."

Example of lists



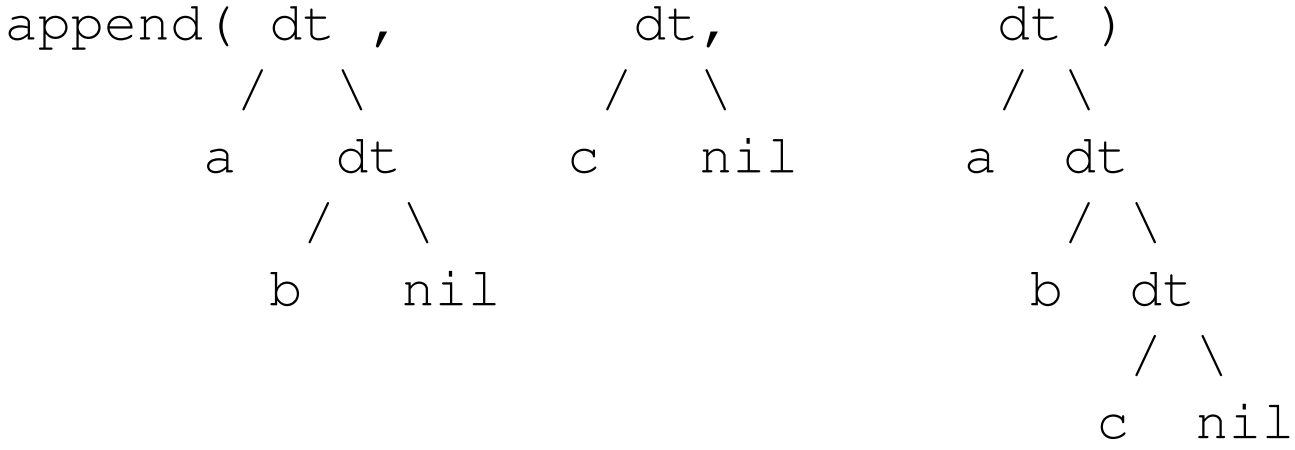


dt(a,dt(b,dt(c,nil)))

dt(c,nil)

nil

The relation append



append(dt(a,dt(b,nil)), dt(c,nil), dt(a,dt(b,dt(c,nil))))

Programming append

- **Progam**

```
append(nil,Y,Y).
append(dt(A,X),Y,dt(A,Z)) :- append(X,Y,Z).
```

- **First execution**

```
> append(dt(a,dt(b,nil)), dt(c,nil), Z)?

Z=dt(a,dt(b,dt(c,nil))).
```

- **Second execution**

```
> append(X, Y, dt(a,dt(b,nil)))?

X=nil, Y=dt(a,dt(b,nil)).
X=dt(a,nil), Y=dt(b,nil).
X=dt(a,dt(b,nil)), Y=nil.
```

First execution, details

```
> append(dt(a,dt(b,nil)), dt(c,nil), Z)?
append(dt(A1,X1), Y1, dt(A1,Z1)) :- append(X1, Y1, Z1).
A1=a, X1=dt(b,nil), Y1=dt(c,nil), Z=dt(A1,Z1)

> append(dt(b,nil), dt(c,nil), Z1)?
append(dt(A2,X2), Y2, dt(A2,Z2)) :- append(X2, Y2, Z2).
A2=b, X2=nil, Y2=dt(c,nil), Z1=dt(A2,Z2)

> append(nil, dt(c,nil), Z2)?
append(nil, Y3, Y3).
Y3=dt(c,nil), Z2=Y3
Z=dt(a,dt(b,dt(c,nil)))
```

Second execution, details

```
> append(X, Y, dt(a,dt(b,nil)))?
append(nil, Y1, Y1).
X=nil, Y=Y1, Y1=dt(a,dt(b,nil))
X=nil, Y=dt(a,dt(b,nil))
append(dt(A1,X1), Y1, dt(A1,Z1)) :- append(X1, Y1, Z1).
X=dt(A1,X1), Y=Y1, A1=a, Z1=dt(b,nil)

> append(X1, Y1, dt(b,nil))?
append(nil, Y2, Y2).
X1=nil, Y1=Y2, Y2=dt(b,nil)
X=dt(a,nil), Y=dt(b,nil)
append(dt(A2,X2), Y2, dt(A2,Z2)) :- append(X2, Y2, Z2).
X1=dt(A2,X2), Y1=Y2, A2=b, Z2=nil

> append(X2, Y2, nil)?
append(nil, Y3, Y3).
X2=nil, Y2=Y3, Y3=nil
X=dt(a,dt(b,nil)), Y=nil
```

.

Prolog II, 1977–1982, University of Marseille, France

Be honest – is he the one or just another beta version?

What is Prolog II?

• Unification is replaced by constraint solving in my explanations.

• Introduction of "different"

```
outlist(X,nil) -> ;
outlist(X,dt(Y,L)) -> dif(X,Y) outlist(X,L);
```

• Allowing infinite trees. "Occur check" is unnecessary.

```
itis(ok) -> equal(X,f(X)) equal(Y,f(Y)) equal(X,Y);

equal(X,X) :- ;


>> itis(A)?
A=ok.
```

Prolog II on Apple II

Let's get back to work!

Henry Kanoui, Michel Van Caneghem and myself implemented Prolog II on an Exorciser Motorola 6800 and on an Apple II computer. It had a virtual memory on a floppy disk addressed by words of three bytes! We received an Apple France award in 1982.

A book in several languages

• **Prolog** Francis Giannesini, Henry Kanoui, Robert Pasero, Michel van Caneghem, in French, Inter Edition 1985.

• **Prolog** Francis Giannesini, Henry Kanoui, Robert Pasero, Michel van Caneghem, in English, Addison Wesley Publishing Company, 1986.

• **Prolog** Francis Giannesini, Henry Kanoui, Robert Pasero, Michel van Caneghem, in German, Addison-Wesley Verlag 1986.

• **Prolog** Francis Giannesini, Henry Kanoui, Robert Pasero, Michel van Caneghem, in Spanish, Addison-Wesley Iberoamericana 1989.

• **Prolog** Francis Giannesini, Henry Kanoui, Robert Pasero, Michel van Caneghem, in Italian, Addison-Wesley.

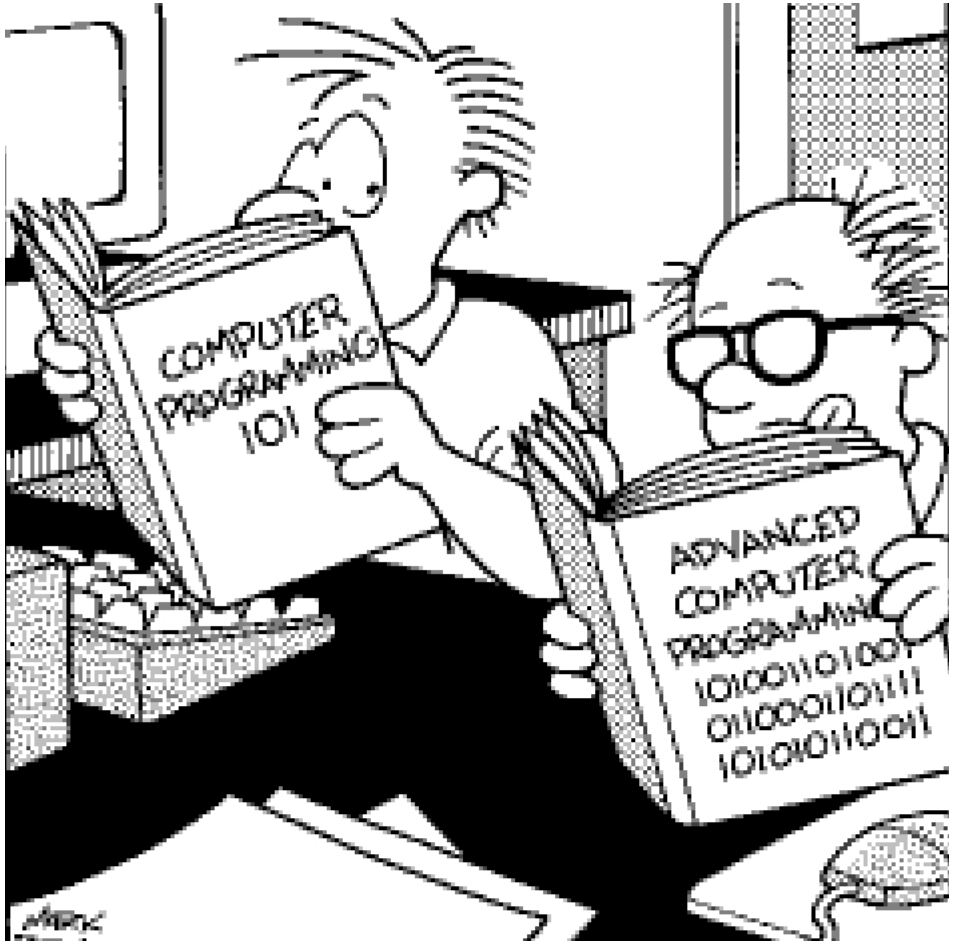
Parallel works

• David H. D. Warren. An abstract Prolog instruction set. Menlo Park, SRI International, October 1983.

• Quintus Computer Systems was founded by him with William Kornfeld, Lawrence Byrd, Fernando Perreira and Cuthbert Hurd to commercialize the Prolog compiler, in 1983.

.

Prolog III, 1983–1990, PrologIA - University of Marseille, France


COMPUTER PROGRAMMING 101
ADVANCED COMPUTER
1010011010
0110001101
1010101001

What is Prolog III?

• The leaves of a tree can be rationnal numbers and Boolean values.

• The addition, all the Boolean operations and also, among other, the dot concatenation operator "•" are allowed.

• Constraint made with $<, \leq, =, \neq, \geq, >, \Rightarrow$ are allowed. Example

```
distinct(dote(b,C)) -> distinct(C) out(b,C), {b>0};
```

• It works using a simplex algorithm with infinite precision, a complete Boolean solver, a incomplete concatenation solver and the algorithm from Prolog II.

How Prolog III works

$$(W,\ t_0 t_1 \ldots t_n,\ S)$$

$$s_0 \rightarrow s_1 \ldots s_m,\ R$$

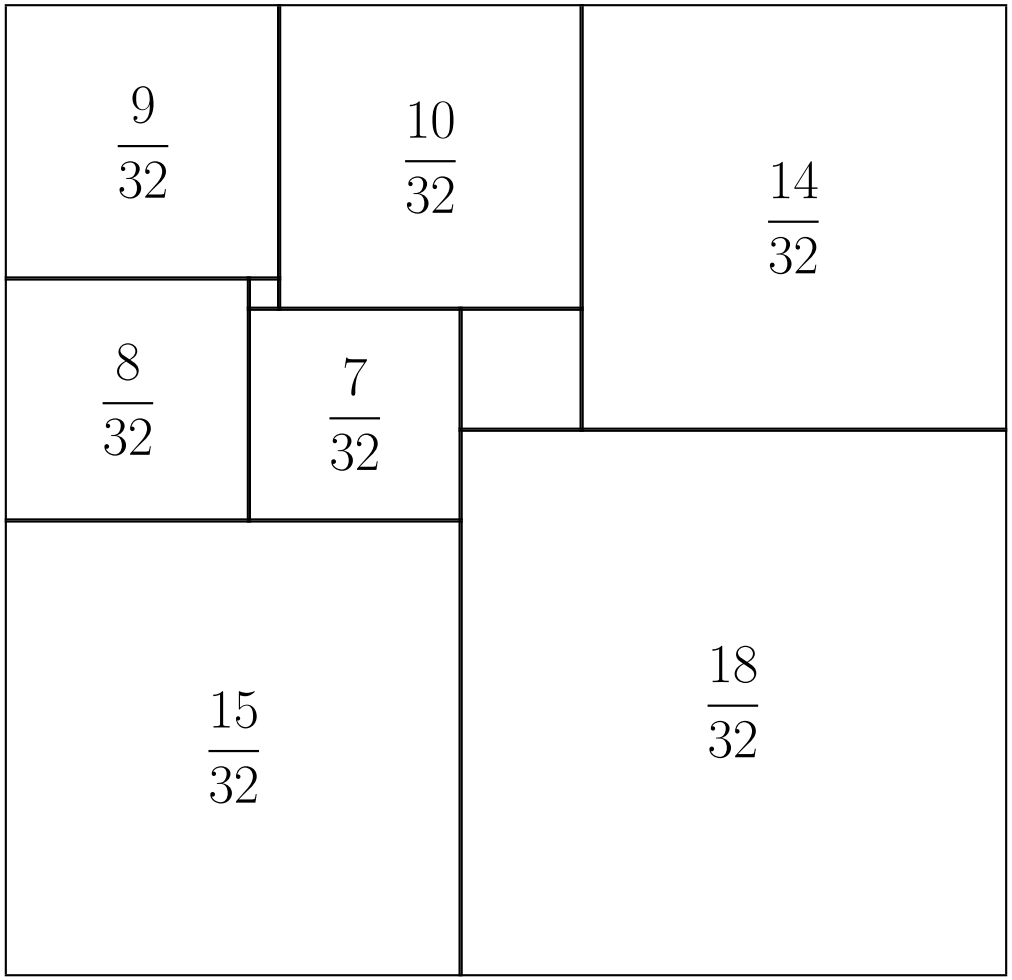
$$(W,\ s_1 \ldots s_m t_1 \ldots t_n,\ S \cup R \cup \{t_0 = s_0\})$$

Decomposition of a rectangle of unknown size $1 \times a$, $a \geq 1$, into 9 unknown squares C

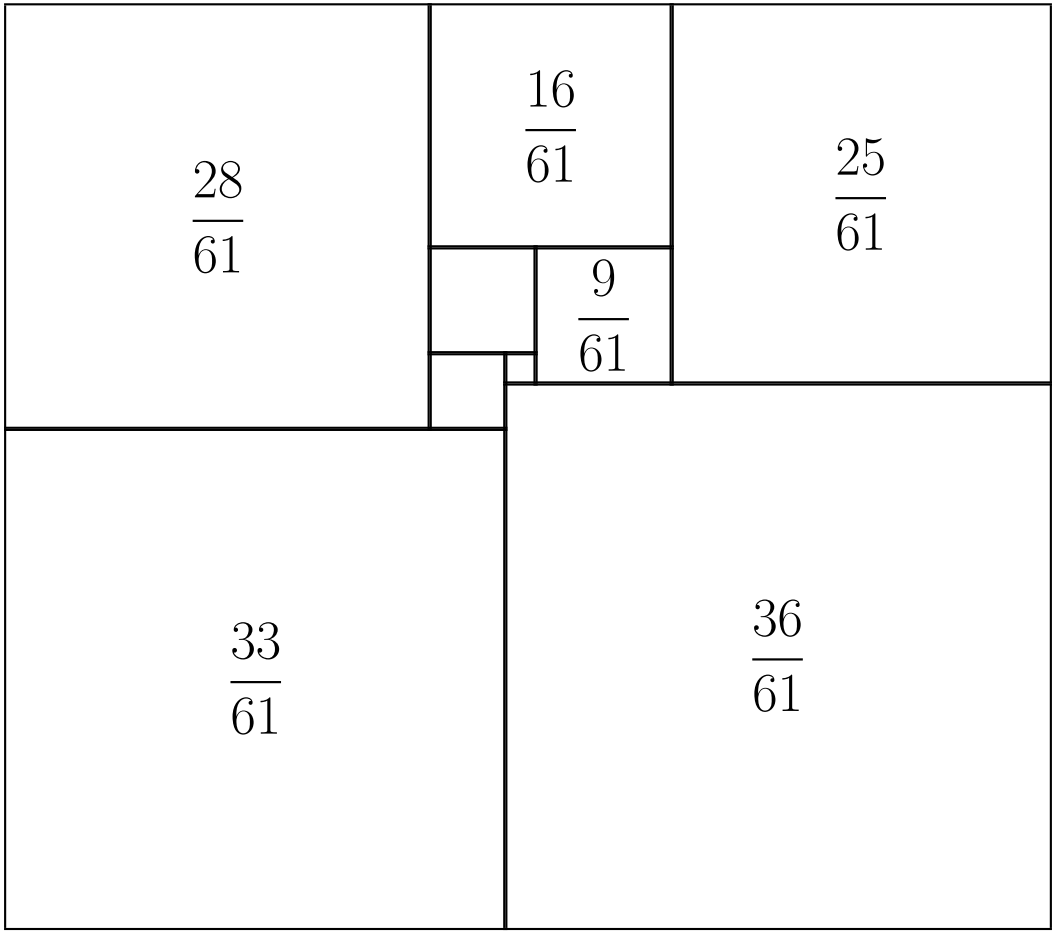
```
>> rectangle(a,C), {|C|=9}?

a = 33/32,
C = <15/32,9/16,1/4,7/32,1/8,7/16,1/32,5/16,9/32>;
a = 69/61,
C = <33/61,36/61,28/61,5/61,2/61,9/61,25/61,7/61,16/61>;
...
```

Alain Colmerauer, An Introduction to Prolog III, *Communications of the ACM*, 33(7): 68-90, 1990.

First solution



Second solution



Program

```
rectangle(a,C) -> distinct(C) area(<-1,a,1>,L,C,<>), {a>=1};

distinct(<>) -> ;
distinct(<b>•C>) -> distinct(C) out(b,C), {b>0};

out(b,<>) -> ;
out(b,<b'>•C>) ->  out(b,C), {b#b'};

area(<v>•L,<v>•L,C,C) -> , {v>=0};
area(<v>•L>,L'',<b>•C,C'') ->
   square(b,L,L')
   area(L',L'',C,C')
   area(<v+b,b>•L'',L''',C',C''),
   {V<0};

square(b,<h,0,h'>•L,L') -> square(b,<h+h'>•L,L'), {b>h};
square(b,<h,v>•L,<-b+v>•L) -> , {b=h};
square(b,<h>•L,<-b,h-b>•L>) -> , {b<h};
```

Boolean Algebra, statement of the problem

Let the 18 statements:

1. Any one, fit to be a Member of Parliament, who is not always speaking, is a public benefactor.
2. Clear-headed people, who express themselves well, have a good education.
3. A woman, who deserves praise, is one who can keep a secret.
4. People, who benefit the public, but do not use their influence for good purpose, are not fit to go into Parliament.
5. People, who are worth their weight in gold and who deserve praise, are always unassuming.
6. Public benefactors, who use their influence for good objects, deserve praise.
7. People, who are unpopular and not worth their weight in gold, never can keep a secret.
8. People, who can talk for ever and are fit to be Members of Parliament, deserve praise.
9. Any one, who can keep a secret and who is unassuming, is a never-to-be-forgotten public benefactor.

10. A woman, who benefits the public, is always popular.
11. People, who are worth their weight in gold, who never leave off talking, and whom it is impossible to forget, are just the people whose photographs are in all the shop-windows.
12. An ill-educated woman, who is not clear-headed, is not fit to go to Parliament.
13. Any one, who can keep a secret and is not for ever talking, is sure to be unpopular.
14. A clear-headed person, who has influence and uses it for good objects, is a public benefactor.
15. A public benefactor, who is unassuming, is not the sort of person whose photograph is in every shop-window.
16. People, who can keep a secret and who who use their influence for good purposes, are worth their weight in gold.
17. A person, who has no power of expression and who cannot influence others, is certainly not a woman.
18. People, who are popular and worthy of praise, either are public benefactors or else are unassuming.

• What is the connection between "able to keep a secret", "fit to be a Member of Parliament" and "worth one's weight in gold"?

• What is the connection between "clear-headed", "popular" and "fit to be a Member of Parliament"?

Boolean Algebra, solution

```
PossibleCase(<
<a,"clear-headed">,
<b,"well-educated">,
 <c,"constantly talking">,
<d,"using one's influence for good objects">,
<e,"exhibited in shop-windows">,
<f,"fit to be a Member of Parliament">,
<g,"public benefactors">,
<h,"deserving praise">,
<i,"popular">,
<j,"unassuming">,
<k,"women">,
<l,"never-to-be-forgotten">,
<m,"influential">,
<n,"able to keep a secret">,
<o,"expressing oneself well">,
<p,"worth one's weight in gold">>) -> ,
```

```
(f & ~c) ==> g,
(a & o) ==> b,
(k & h) ==> n,
(g & ~d) ==> ~f,
(p & h) ==> j,
(g & d) ==> h,
(~i & ~p) ==> ~n,
(c & f) ==> h,
(n & j) ==> (g & l),
(k & g) ==> i,
(p & c & l) ==> e,
(k & ~a & ~ b) ==> ~f,
(n & ~c) ==> ~i,
(a & m & d) ==> g,
(g & j) ==> ~e,
(n & d) ==> p,
(~o & ~m) ==> ~k,
(i & h) ==> (g | j);
```

```
PossibleSubCase(x) -> PossibleCase(y) SubSet(x,y);

SubSet(<>,y) ->;
SubSet(<e>•x,y) -> ElementOf(e,y) SubSet(x,y);

ElementOf(e,<e>•y) ->;
ElementOf(e,<e'>•y) -> ElementOf(e, y), {e#e'};
```

Boolean Algebra, solution, execution

- Answering the query:

What is the connection between "able to keep a secret", "fit to be a Member of Parliament" and "worth one's weight in gold"?

is to execute:

```
PossibleSubCase(<
<p,"able to keep a secret">,
<q,"fit to be a Member of Parliament">,
<r,"worth one's weight in gold">>) ?
```

- The answer is:

```
{p&q => r},
```

which means that:

The persons who can keep a secret and are fit to be a Member of Parliament are worth their weight in gold.

• Answering the query:

What is the connection between "clear-headed", "popular" and "fit to be a Member of Parliament"?

is to execute:

```
PossibleSubCase(<
<p,"clear-headed">,
<q,"popular">,
<r,"fit to be a Member of Parliament">>)?
```

• The answer is the set of constraints:

```
{p : bool, q : bool, r : bool},
```

which means that:

There is no connection between "clear-headed", "popular" and "fit to be a Member of Parliament".

Parallel work to Prolog III, 1988–1992

• **CHIP** M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier. The Constraint Logic Programming Language CHIP. International Conference on FGCS 1988, Tokyo, November 1988.

• **BNR-Prolog** William J. Older and André Vellino. Extending Prolog with Constraint Arithmetic on Real Intervals. Proceedings of the Canadian Conference on Electric and Computer Engineering 1990. Will be important for Prolog IV.

• **CLP(R)** J. Jaffar, S. Michaylov, P. J. Stuckey, H. C. Yap. The CLP(R) language and system, ACM transactions on programming languages and systems, 1992, vol. 14, no3, pp. 339-395.

Creation of "constraint programming" companies in France 1984–1990

• **Prologia** Created in 1984 by Alain Colmerauer, Henry Kanoui, Henry Garetta, Geneviève Guérideau, Robert Pasero, Jean-François Pique, Michel Van Caneghem, from the University of Marseille. Purpose: development of Prolog and marketing.

• **Ilog** Created in 1987 by Pierre Haren, Marc Fourrier and Jérôme Chailloux, with the support of INRIA (Institut National de Recherche en Informatique et en Automatique). Purpose: development of Le-Lisp and marketing ... Pecos then Ilog solver in 1991.

• **Cosytec** Created in 1990 by Mehmet Dincbas, Abderrahmane Aggoun, Helmut Simonis . . . , coming from the CHIP team at ECRC (European Computer-Industry Research Centre). Purpose: development CHIP and marketing.

.

Prolog IV, 1990–1996, PrologIA, France


I CAN BE UPGRADED. CAN YOU?
A. BACALL

What is Prolog IV?

• The leaves of a tree can be real numbers. The Boolean values are as simple as 0 and 1, which are integers, which are rational numbers, which are real numbers.

• 120 evaluable predicates which generate constraints.

• ISO-standard syntax. P. Deransart, A Ed-Dbali and L. Cervoni, *Prolog: The Standard*, Springer, 1996.

• It utilizes mainly an algorithm of approximate resolution by computing a fixed point. This was the basis of BNR-Prolog.

F. Benhamou, P. Bouvier, A. Colmerauer, H. Garetta, B. Giletta, J.L. Massat, G.A. Narboni, S. N'Dong, R. Pasero, J.F. Pique, Touraîvane, M. Van Caneghem et E. Vétillard. *Le manuel de Prolog IV*. PrologIA, Marseille, June 1996.

Approximate resolution by computing a fixed point

- Let us find the x's such that

$$x = \cos x.$$

- We assume that we have at our disposal only the real numbers:

$$-\infty, -15/10, -14/10, -13/10, \ldots, 13/10, 14/10, 15/10, +\infty$$

- Let us introduce the relations

$$\begin{aligned} \textbf{cos} &:= \{(x,y) \in \mathbb{R}^2 \,|\, y = \cos x\}, \\ \textbf{egal} &:= \{(x,y) \in \mathbb{R}^2 \,|\, y = x\}. \end{aligned}$$

- It is sufficient to solve

$$(x,y) \in \textbf{cos} \ \wedge \ (x,y) \in \textbf{egal} \ \wedge \ x \in \mathbb{R} \ \wedge \ y \in \mathbb{R}$$

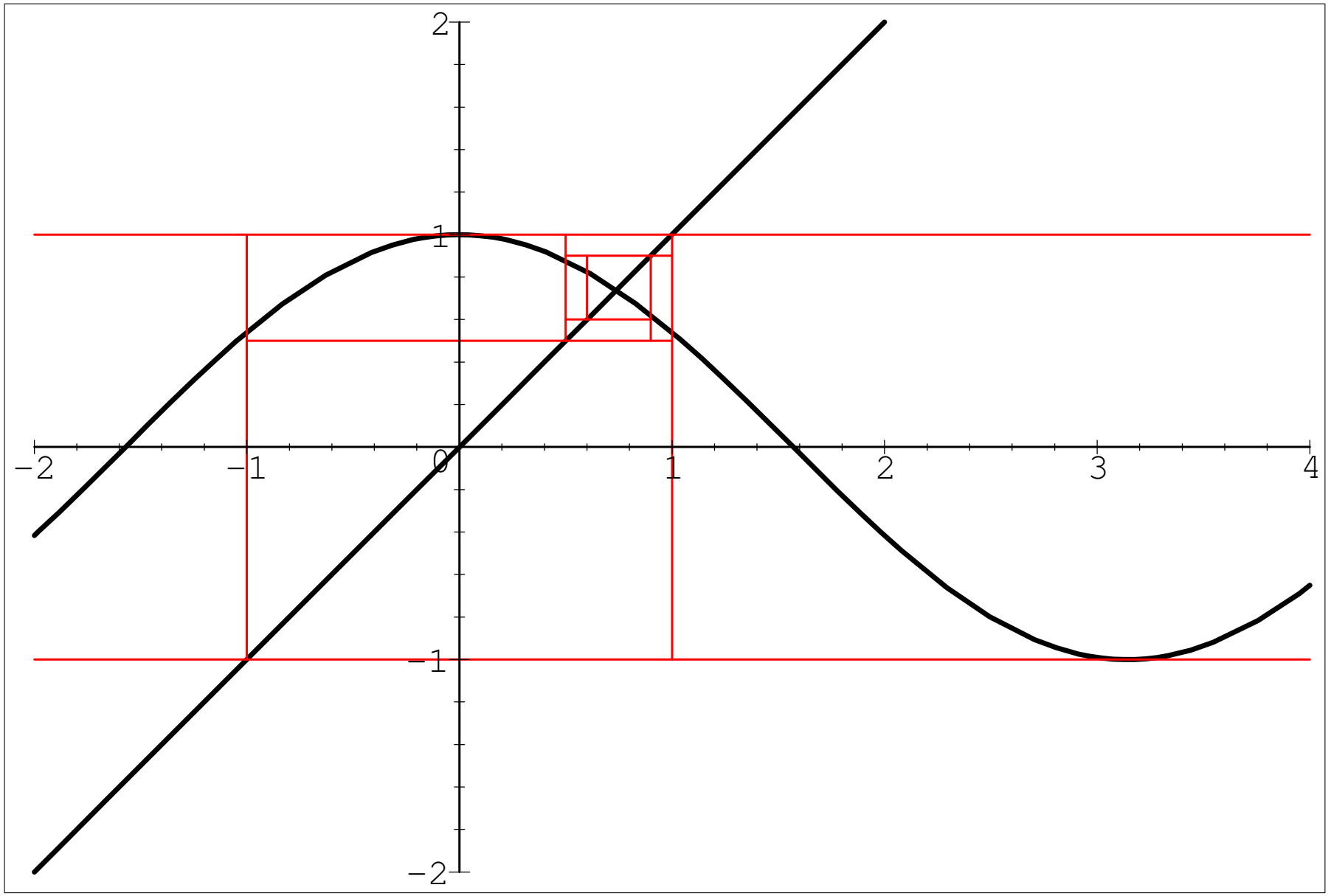
- By narrowing the cosinus and the equality one after another, we obtain

$$
\begin{array}{llllll}
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in \mathbb{R} & \wedge\ y \in \mathbb{R} & \Longrightarrow \\
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in \mathbb{R} & \wedge\ y \in [-1,\ 1] & \Longrightarrow \\
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in [-1,\ 1] & \wedge\ y \in [-1,\ 1] & \Longrightarrow \\
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in [-1,\ 1] & \wedge\ y \in [5/10,\ 1] & \Longrightarrow \\
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in [5/10, 1] & \wedge\ y \in [5/10,\ 1] & \Longrightarrow \\
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in [5/10, 1] & \wedge\ y \in [5/10,\ 9/10] & \Longrightarrow \\
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in [5/10, 9/10] & \wedge\ y \in [5/10,\ 9/10] & \Longrightarrow \\
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in [5/10, 9/10] & \wedge\ y \in [6/10,\ 9/10] & \Longrightarrow \\
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in [6/10, 9/10] & \wedge\ y \in [6/10,\ 9/10] & \Longrightarrow \\
(x,y) \in \textbf{cos} & \wedge\ (x,y) \in \textbf{egal} & \wedge\ x \in [6/10, 9/10] & \wedge\ y \in [6/10,\ 9/10]. &
\end{array}
$$

$$\cos 1 \in [5/10, 6/10],\ \cos 9/10 \in [6/10, 7/10],\ \cos 6/10 \in [8/10, 9/10],\ \cos 5/10 \in [8/10, 9/10]$$

- Thus

$$x \in [6/10,\ 9/10]$$


2
1
−1
−2
−2
−1
0
1
2
3
4

dot of all the constraints of Prolog IV

abs/2
and/2
arccos/2
arcsin/2
arctan/2
band/3
bcc/4
bco/4
bdif/3
beq/3
bequiv/3
bfinite/2
bge/3
bgt/3
bidentifier/2
bimpl/3
binfinite/2
bindot/3
bint/2
ble/3
bleaf/1
bdot/1
blt/3
bnidentifier/2
bnint/2
bnleaf/1
bndot/1
bnot/2
• bnprime/2
bnreal/2
boc/4
boo/4
bor/3
boutcc/4
boutco/4
boutdot/3
boutoc/4
boutoo/4
• bprime/2
breal/2
bxor/3
cc/3
ceil/2
co/3
conc/3
cos/2
cosh/2
cot/2
coth/2
dif/2
div/3
divlin/3
eq/2
equiv/2
exp/2
finite/1
floor/2
• gcd/3
ge/2
gelin/2
gt/2
gtlin/2
identifier/1
if/4
impl/2
index/3
infinite/1
indot/2
• intdiv/3
int/1
• lcm/3
le/2
leaf/1
lelin/2
dot/1
ln/2
log/2
lt/2
ltlin/2
max/3
min/3
minus/3
minuslin/3
• mod/3
n/3
nidentifier/1
nint/1
ndot/1
nleaf/1
not/1
• nprime/1
ntree/1
nreal/2
oc/3
oo/3
or/2
outcc/3
outco/3
outdot/2
outoc/3
outoo/3
pi/1
plus/3
pluslin/3
power/3
• prime/1
real/1
root/3
sin/2
sinh/2
size/2
sqrt/2
square/2
tan/2
tanh/2
times/3
timeslin/3
tree/1
u/3
uminus/2
uminuslin/2
uplus/2
upluslin/2
xor/2

Some constraints

1 bgt/3	$\mapsto$	$\{(a,b,c) \in \mathbf{B} \times \mathbb{R}^2 \mid a = 1 \text{ ssi } b > c\}$
2 conc/3	$\mapsto$	$\{(a,b,c) \in \mathbf{L}^3 \mid a = b \bullet c\}$
3 cos/2	$\mapsto$	$\{(a,b) \in \mathbb{R}^2 \mid a = \cos b\}$
4 dif/2	$\mapsto$	$\{(a,b) \in \mathbf{A}^2 \mid a \neq b\}$
5 eq/2	$\mapsto$	$\{(a,b) \in \mathbf{A}^2 \mid a = b\}$
6 floor/2	$\mapsto$	$\{(a,b) \in \mathbf{Z} \times \mathbb{R} \mid a = \lfloor b \rfloor\}$
7 ge/2	$\mapsto$	$\{(a,b) \in \mathbb{R}^2 \mid a \geq b\}$
8 gelin/2	$\mapsto$	$\{(a,b) \in \mathbb{R}^2 \mid a \geq b\}$
9 le/2	$\mapsto$	$\{(a,b) \in \mathbb{R}^2 \mid a \leq b\}$
10 plus/3	$\mapsto$	$\{(a,b,c) \in \mathbb{R}^3 \mid a = b + c\}$
11 pluslin/3	$\mapsto$	$\{(a,b,c) \in \mathbb{R}^3 \mid a = b + c\}$
12 index/3	$\mapsto$	$\{(a,b,c) \in \mathbf{A} \times \mathbf{L} \times \mathbf{Z} \mid 1 \leq c \leq \lvert b \rvert$ et a est le $c^{\text{ième}}$ élément de $b\}$
13 size/2	$\mapsto$	$\{(a,b) \in \mathbf{Z} \times \mathbf{L} \mid a = \lvert b \rvert\}$
14 times/3	$\mapsto$	$\{(a,b,c) \in \mathbb{R}^3 \mid a = b \times c\}$

A : set of the trees, **B** : $\{0,1\}$ set,
L : set of the dots, **Z** : integer numbers set, $\mathbb{R}$: real number set.

A complex constraint for Prolog IV

$$\exists u \exists v \exists w \exists x \left(\begin{array}{ll} & y \leq 5 \\ \wedge & v_1 = \cos v_4 \\ \wedge & \mathbf{size}(u) = 3 \\ \wedge & \mathbf{size}(v) = 10 \\ \wedge & u \bullet v = v \bullet w \\ \wedge & y \geq 2 + (3 \times x) \\ \wedge & x = (74 > \lfloor 100 \times v_1 \rfloor) \end{array} \right)$$

$y = 5$

```
>> U ex V ex W ex X ex
le(Y,5),
V:1 = cos(V:4),
size(U) = 3,
size(V) = 10,
U o V = V o W,
ge(Y,2.+.(3.*.X)),
X = bgt(74,floor(100.*.V:1)).

Y = 5.
```

Decomposition of a rectangle of unknown size $1 \times A$, $A \geq 1$, into 9 unknown squares C

```
> size(C)=9, rectangle(A,C)?

A = 33/32,
C = [15/32,9/16,1/4,7/32,1/8,7/16,1/32,5/16,9/32];
A = 69/61,
C = [33/61,36/61,28/61,5/61,2/61,9/61,25/61,7/61,16/61];
...
```

Program

```
rectangle(A,C) :- gelin(A,1), distinct(C), area([-1,A,1],L,C,[]).

distinct([]).
distinct([B|C]) :- gtlin(B,0), distinct(C), out(B,C).

out(B,[]).
out(B,[Bp|C]) :- dif(B,Bp), out(B,C).

area([V|L],[V|L],C,C) :- gelin(V,0).
area([V|L],Lppp,[B|C],Cpp) :-
   lt(V,0),
   square(B,L,Lp),
   area(Lp,Lpp,C,Cp),
   area([V+B,B|Lpp],Lppp,Cp,Cpp).

square(B,[H,0,Hp|L],Lp) :-gtlin(B,H), square(B,[H+Hp|L],Lp).
square(B,[H,V|L],[-B+V|L]) :-B=H.
square(B,[H|L],[-B,H-B|L]) :-ltlin(B,H).
```

.

Today, 2011

Prologs actually available?

• **Prolog IV** is difficult to get. The company PrologIA was sold in 2000.See http://prolog-heritage.org

• **SWI-Prolog,** 2008, Jan Wielemaker, Human Computer-Studies Laboratory, University Amsterdam. I recommend it.

• **Gnu Prolog,** 2009, Daniel Diaz, Université Paris 1.

• **XSB Prolog,** 2009, Computer Science Department of Stony Brook University, Universidade Nova de Lisboa, XSB, Inc, Katholieke Universiteit Leuven, and Uppsala Universitet.

• **SICStus** 2010, Mats Carlson, SICS, Sweden.

Where to find this presentation, for those who are interested

http://alain.colmerauer.free.fr/alcol/ArchivesPublications/Transparents/ChinaApril2011/NatLanguageProlog.pdf