.

My Happy Contribution to Constraint Programming

Alain Colmerauer

Award for Research Excellence in Constraint Programming, 2008, Sydney

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

Contents

.

W-grammar

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

• 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**

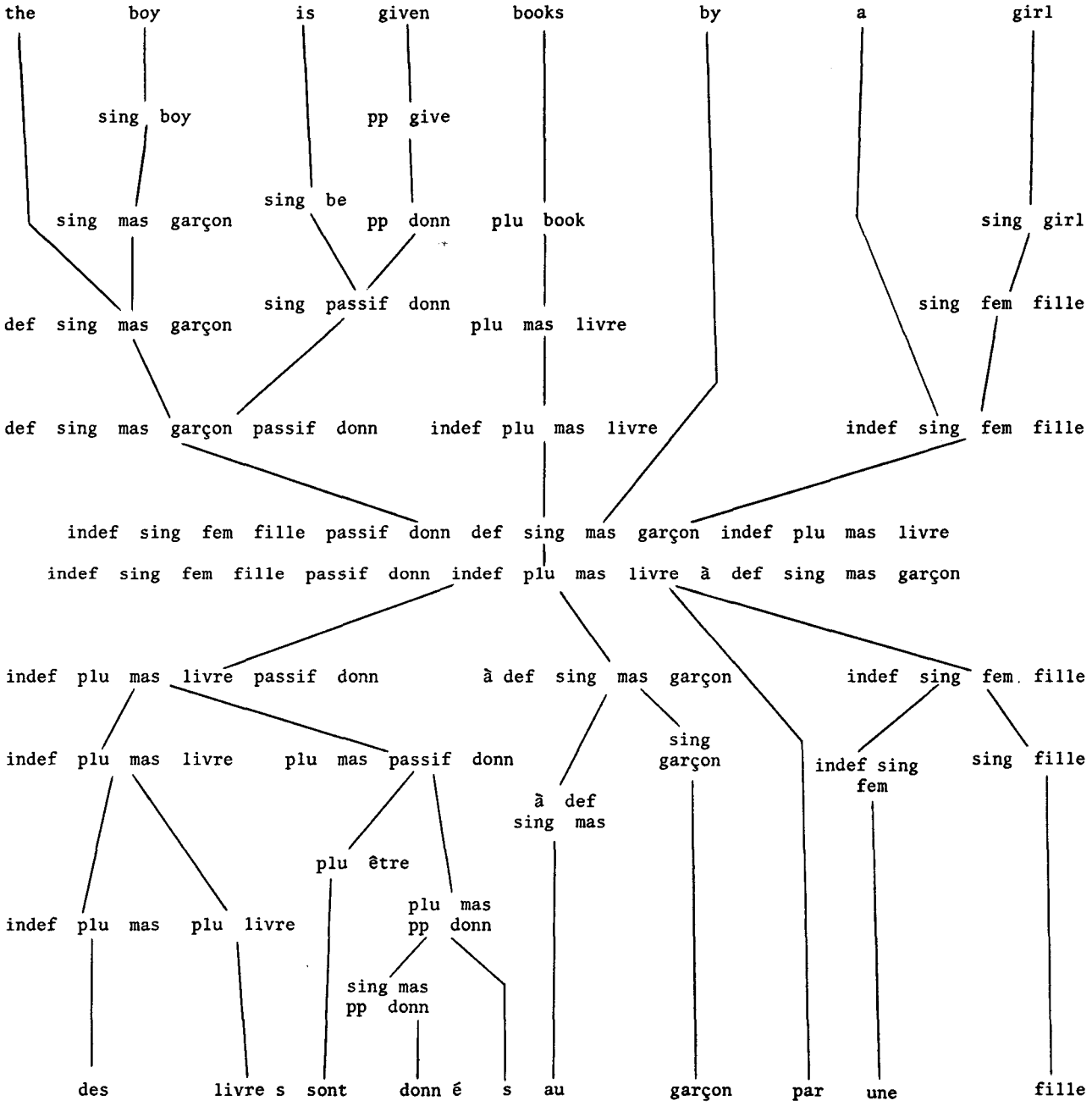
```
the , boy , is , given , books , by , a , girl
```

- **Output**

```
des , livre s , sont , donn é s , au , garçon , 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 French, next


the
boy
is
given
books
by
a
girl
sing boy
pp give
sing be
sing mas garçon
pp donn
plu book
sing girl
sing passif donn
sing fem fille
def sing mas garçon
plu mas livre
def sing mas garçon passif donn
indef plu mas livre
indef sing fem fille
indef sing fem fille passif donn def sing mas garçon indef plu mas livre
indef sing fem fille passif donn indef plu mas livre à def sing mas garçon
indef plu mas livre passif donn
à def sing mas garçon
indef sing fem fille
indef plu mas livre
plu mas passif donn
sing garçon
indef sing fem
sing fille
à def sing mas
plu être
plu mas pp donn
indef plu mas plu livre
sing mas pp donn
des
livre s
sont
donn é
s
au
garçon
par
une
fille

English grammar

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

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

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

à 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 garçon --> 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

```
P -> SN SV

SV -> MODE V SN à SN

MODE -> actif
MODE -> passif

SN -> ART NOMBRE NOM

ART -> def
ART -> indef

NOMBRE -> sing
NOMBRE -> plu

NOM -> GENRE N

GENRE -> mas
GENRE -> fem

N -> garçon
N -> fille
N -> livre

V -> donn

SN1 -> SN

SN2 -> SN

FORME -> pp
FORME -> NOMBRE actif

CO -> SN
CO -> à SN
```

French grammar

```
SN actif V SN1 à SN2 -->
   SN actif V ,
   SN1 ,
   à SN2
SN1 passif V SN2 à SN2 -->
   SN1 passif V ,
   à 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
à def sing mas --> au
à BONART --> à , BON-ART
indef sing mas --> un
indef sing fem --> une
indef plu GENRE --> des
à def plu GENRE --> aux
NOMBRE GENRE passif V -->
   NOMBRE être ,
   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 , é
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

```
P -> SN SV

SV -> MODE V SN à SN

MODE -> actif
MODE -> passif

SN -> ART NOMBRE NOM

ART -> def
ART -> indef

NOMBRE -> sing
NOMBRE -> plu

NOM -> GENRE N

GENRE -> mas
GENRE -> fem

N -> garçon
N -> fille
N -> livre

V -> donn

SN1 -> SN

SN2 -> SN

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

AART -> ART
AART -> à ART
```

.

Q-system

Q-system for $A^n B^n C^n$

Grammar

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

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

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

C(1,U*) == C + C(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 $A^n B^n C^n$, variant

- **Grammar**

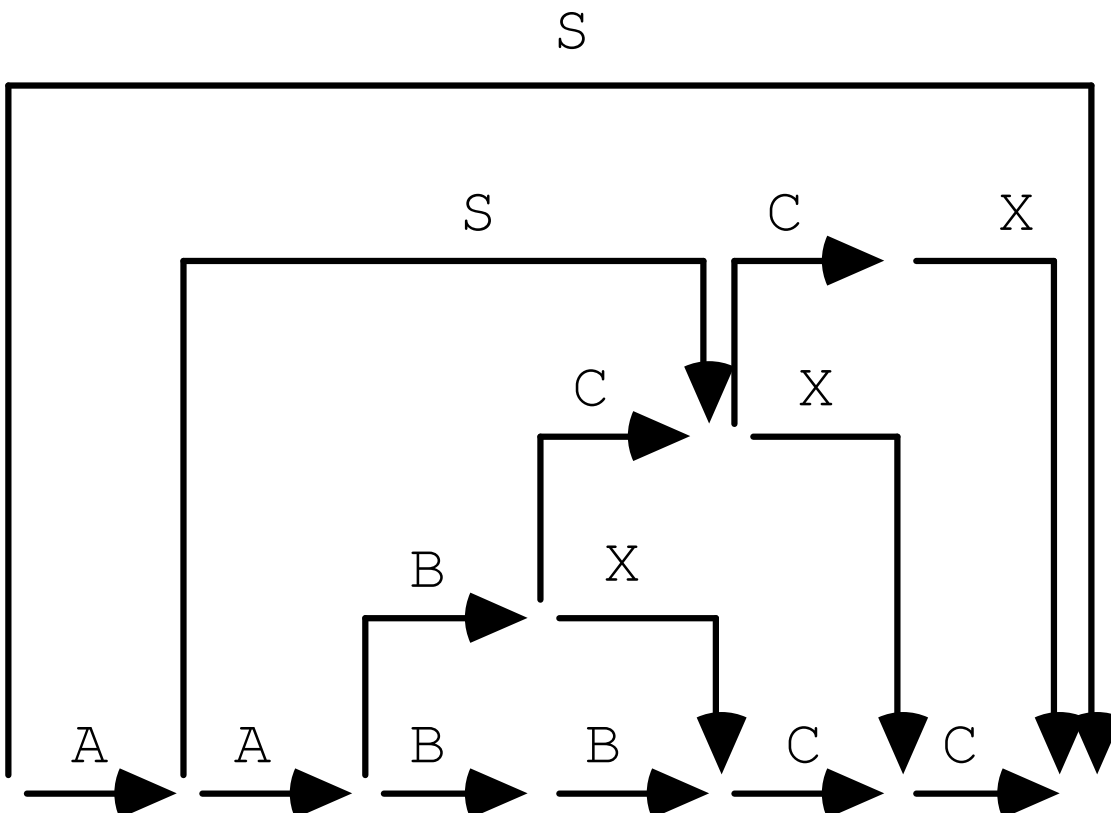
```
S == A + B + C.

S == A + S + X + C.

C + X == X + C.

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

Q-system for $A^nB^nC^n$, variant, next


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

Automatic Translation, TAUM, Example, 1973

Input

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

Output

```
L’EXPANSION DES ACTIVITES GOUVERNEMENTLALES 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.

TAUM, Example, phases 0-1

Phase 0

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

Phase 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-
```

TAUM, Example, phases 2-5

Phase 2

Phase 3

Phase 4

Phase 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-
```

TAUM, Example, phase 6

Phase 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-
```

TAUM, Example, phase 7

Phase 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-
```

TAUM, Example, phase 8

Phase 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-
```

TAUM, Example, phase 9

Phase 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-
```

TAUM, Example, phase 10

Phase 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-
```

TAUM, Example, phases 11-16

Phase 11

Phase 12

Phase 13

Phase 14

Phase 15

Phase 16

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

TAUM, Example, grammar

Q-system 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.
...
```

.

Prolog

Prolog, an example

• Program

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

• Execution

```
append(X,Y,list(a,list(b,list(c,nil))))?

X=nil, Y=list(a,list(b,list(c,nil))).
X=list(a,nil), Y=list(b,list(c,nil)).
X=list(a,list(b,nil)), Y=list(c,nil).
X=list(a,list(b,list(c,nil))), Y=nil.
```

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.

Parsing of $a^n b^n c^n$ in Prolog

- **Grammar**

```
goodstring(X0,X1,N) :-
   sequense(X0,X1,N,a),
   sequense(X1,X2,N,b),
   sequense(X2,X3,N,c).

sequence(X0,X0,zero,S) :- .
sequence(list(S,X0),X1,plus(one,N),S) :-
   sequence(X0,X1,N,S).
```

- **Execution**

```
goodstring(list(a,list(a,list(b,list(b,list(c,list(c,nil)))))),
                                                    nil,N)?
N = plus(one,plus(one,zero)).
```

Inference in natural language

• Original

TOUT PSYCHIATRE EST UNE PERSONNE.
CHAQUE PERSONNE QU'IL ANALYSE, EST MALADE.
*JACQUES EST UN PSYCHIATRE A *MARSEILLE.

EST-CE QUE *JACQUES EST UNE PERSONNE? OU EST *JACQUES? EST-CE QUE *JACQUES EST MALADE?

> OUI. A MARSEILLE. JE NE SAIS PAS.

• English version

Every psychiatrist is a person.
Every person he analyzes is sick.
Jacques is a psychiatrist in Marseille.

Is Jacques a person? Where is Jacques? Is Jacques sick?

> Yes. In Marseille. I don't know.

.

Prolog II

Prolog II, a step toward constraint programming

- **Unification is replaced by constraint solving**

- **Introduction of "different"**

```
outlist(X,nil) :- .
outlist(X,list(Y,L)) :- outlist(X,L), {X#Y}.
```

- **Allowing infinite trees, no more "occur check"**

```
itis(ok) :- infinitetree(X) infinitetree(Y) equal(X,Y).
infinitetree(X) :- equal(X,f(X)).
equal(X,X) :- .
```

```
itis(A)?

A=ok.
```

Prolog II on Apple II

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 for that.

.

Prolog III

What is Prolog III?

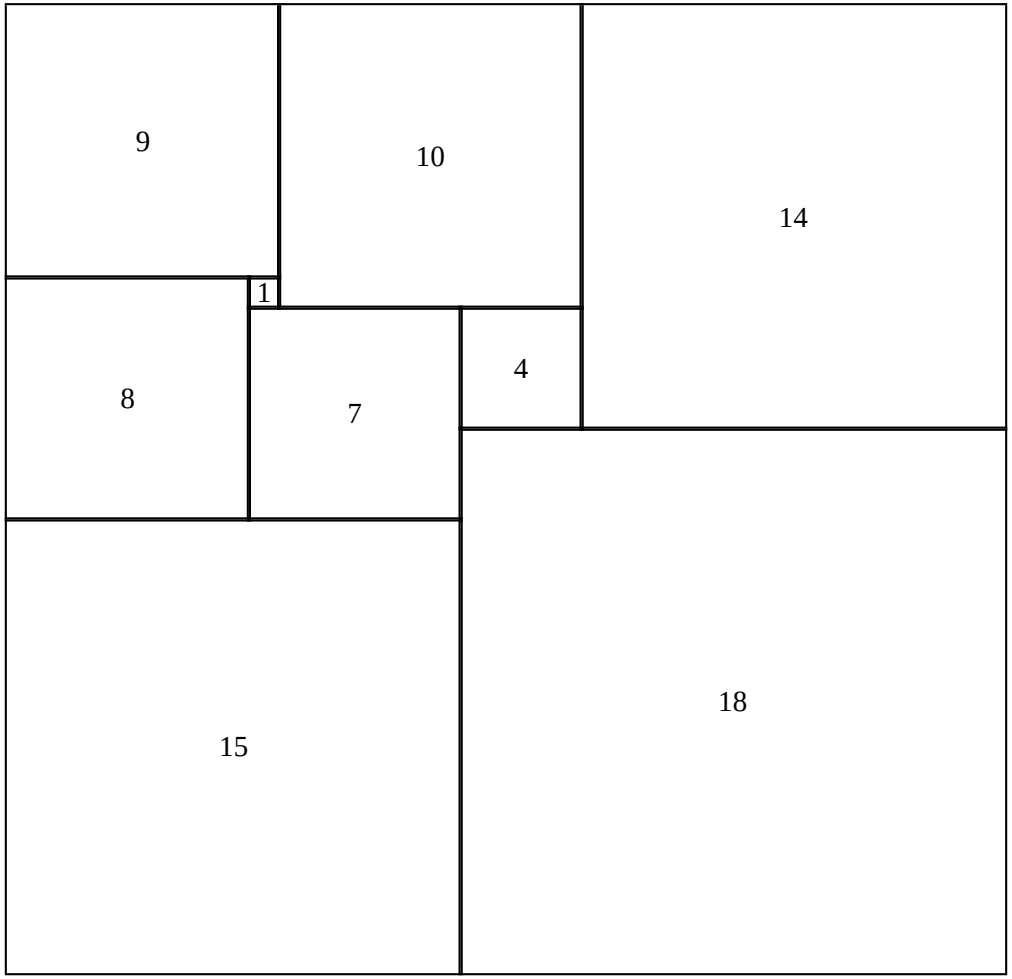
- The leaves of a tree can be rationnal numbers, that is fractions p/q.

- The addition $+$ and the substraction $-$ are allowed.

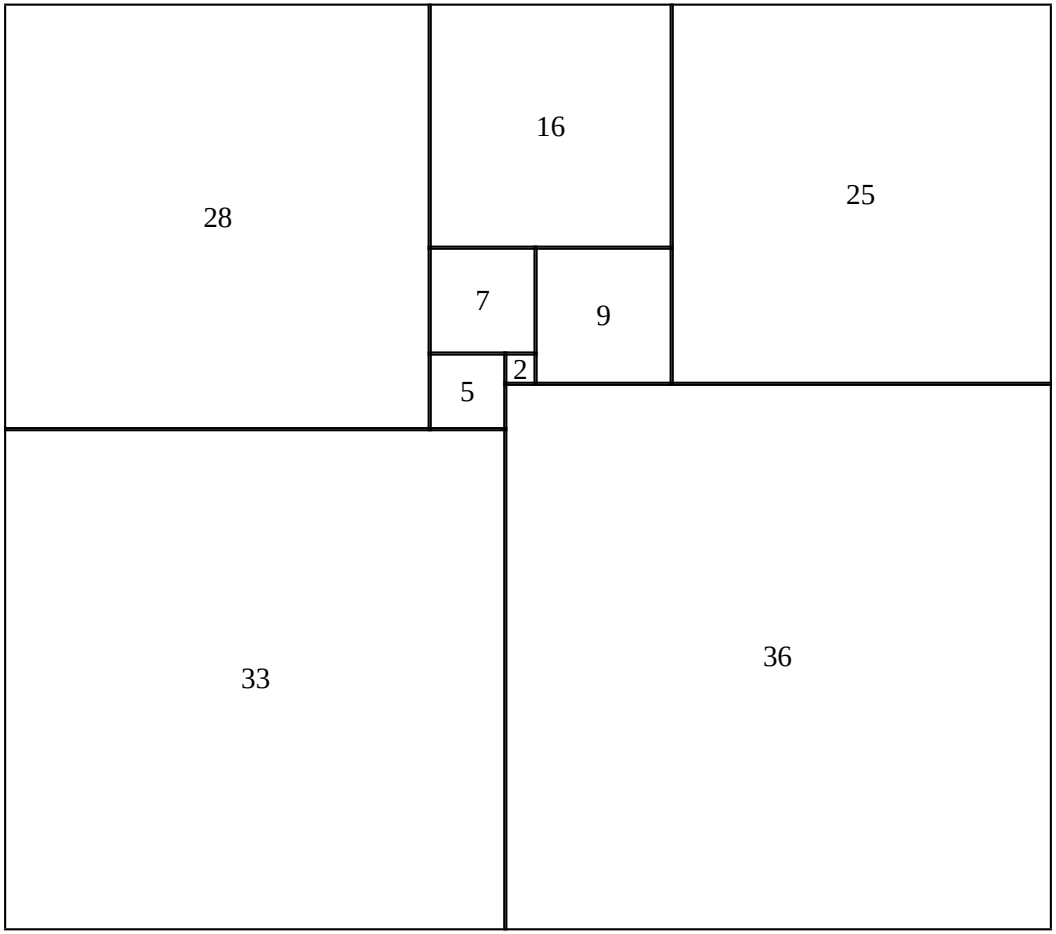
- Constraint with $<, \leq, =, \neq, \geq, >$ are allowed.

- It works using a simplex algorithm and the algorithm from Prolog II.

Decomposition of a rectangle into 9 squares



Decomposition of a rectangle into 9 squares, next



Alain Colmerauer, An Introduction to Prolog III, *Communications of the ACM*, 33(7): 68-90, 1990. A preliminary version is available in `http://alain.colmerauer@free.fr`.

Decompostion into n squares with Prolog III

```
rectangle(A,C) :-
   distinctSizes(C)
   area([-1,A,1],L,C,[]),
   A>=1.

distinctSizes([]).
distinctSizes([B|C]) :-
   distinctSizes(C)
   out(B,C),
   B>0.

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

```
area([V|L],[V|L],C,C) :- ,
   V>=0.
area([V|L],Lppp,[B|C],Cpp) :-
   square(B,L,Lp)
   area(Lp,Lpp,C,Cp)
   area([V+B,B|Lpp],Lppp,Cp,Cpp),
   V<0.

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

Decomposition into 9 squares with Prolog III, next

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

.

Around the world 1990

Around the world

• 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.

• Prolog III

Alain Colmerauer. Final Specifications for Prolog III. Esprit project P1219 (1106), Further Development of Prolog and its Validation by KBS in Technical Areas. Milestone II, February 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.

• 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 three companies in France

• 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 and ?, comming from the CHIP team at ECRC (European Computer-Industry Research Centre). Purpose: development CHIP and marketing.

.

Prolog IV

A complex constraint for Prolog IV

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

$y = 5$

```
>> 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.
```

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

Alain Colmerauer, *Prolog IV*, 1995, `htp://alain.colmerauer@free.fr`, in French.

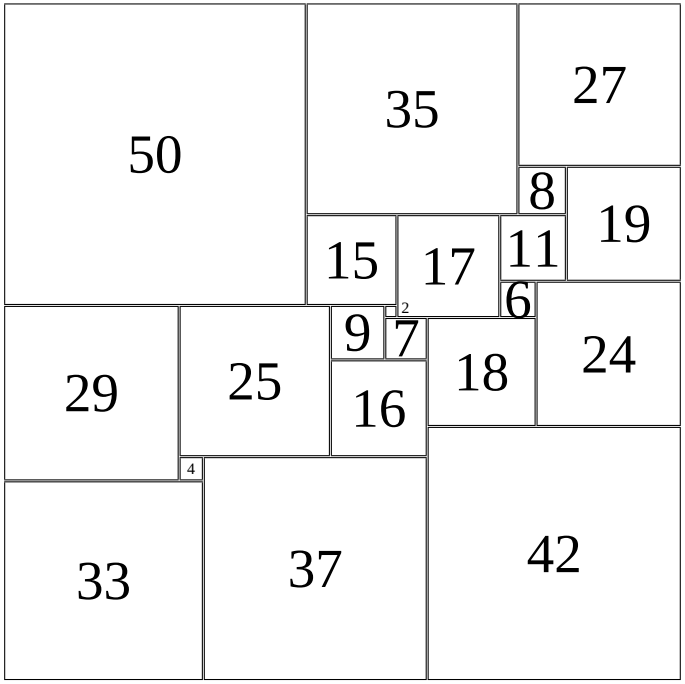
List of all the constraints of Prolog IV

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
binlist/3
bint/2
ble/3
bleaf/1
blist/1
blt/3
bnidentifier/2
bnint/2
bnleaf/1
bnlist/1
bnot/2
• bnprime/2
bnreal/2
boc/4
boo/4
bor/3
boutcc/4
boutco/4
boutlist/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
inlist/2
• intdiv/3
int/1
• lcm/3
le/2
leaf/1
lelin/2
list/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
nlist/1
nleaf/1
not/1
• nprime/1
ntree/1
nreal/2
oc/3
oo/3
or/2
outcc/3
outco/3
outlist/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

.

Decomposition of an integer square into integer squares

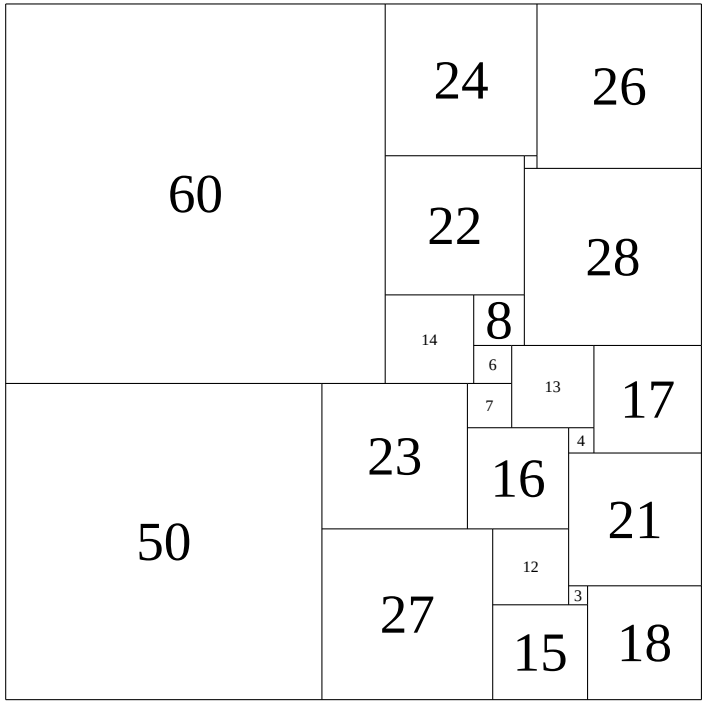
Decomposition of a square into 21 different squares

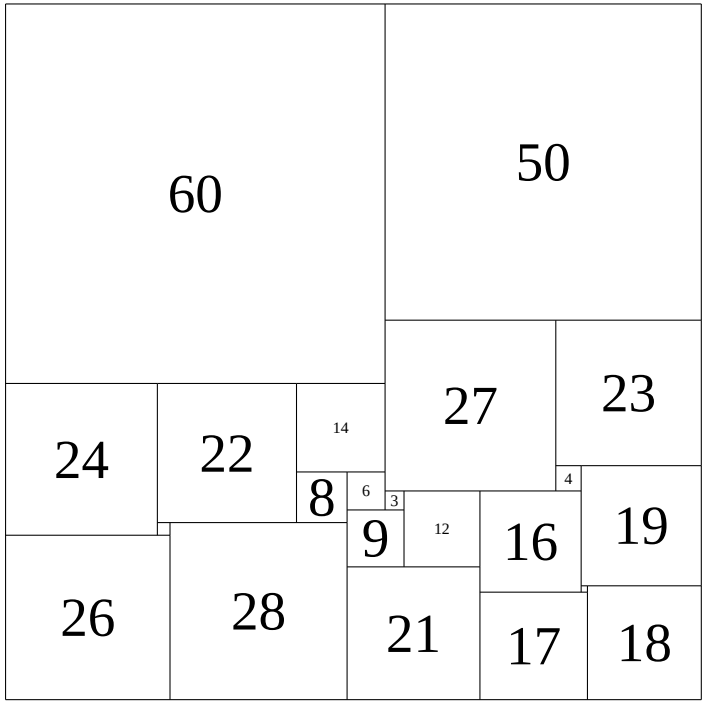


We find that the size of the initial square is $\mathbf{112 \times 112}$.

A.J.W. Duijvestijn, Simple Perfect Squared Square of Lowest Order, *Journal of Combinatorial Theory*, Series B 25, pages 240-243, 1978.

Decomposition of a square into 22 different squares





But the size of the initial square is only 110×110.

Ian Gambini, A method for squaring the square, *Discrete Applied Mathematics 98*, pages 65-80, 1999.

.

Narrowing

Optimal interval narrowing of the sortedness constraint

• **The sortedness constraint**

$$\textit{sort}(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{2n}) \quad \equiv \quad \begin{cases} (x_{n+1}, \ldots, x_{2n}) \\ \text{is equal to} \\ (x_1, \ldots, x_n) \text{ sorted} \\ \text{in non-decreasing order.} \end{cases}$$

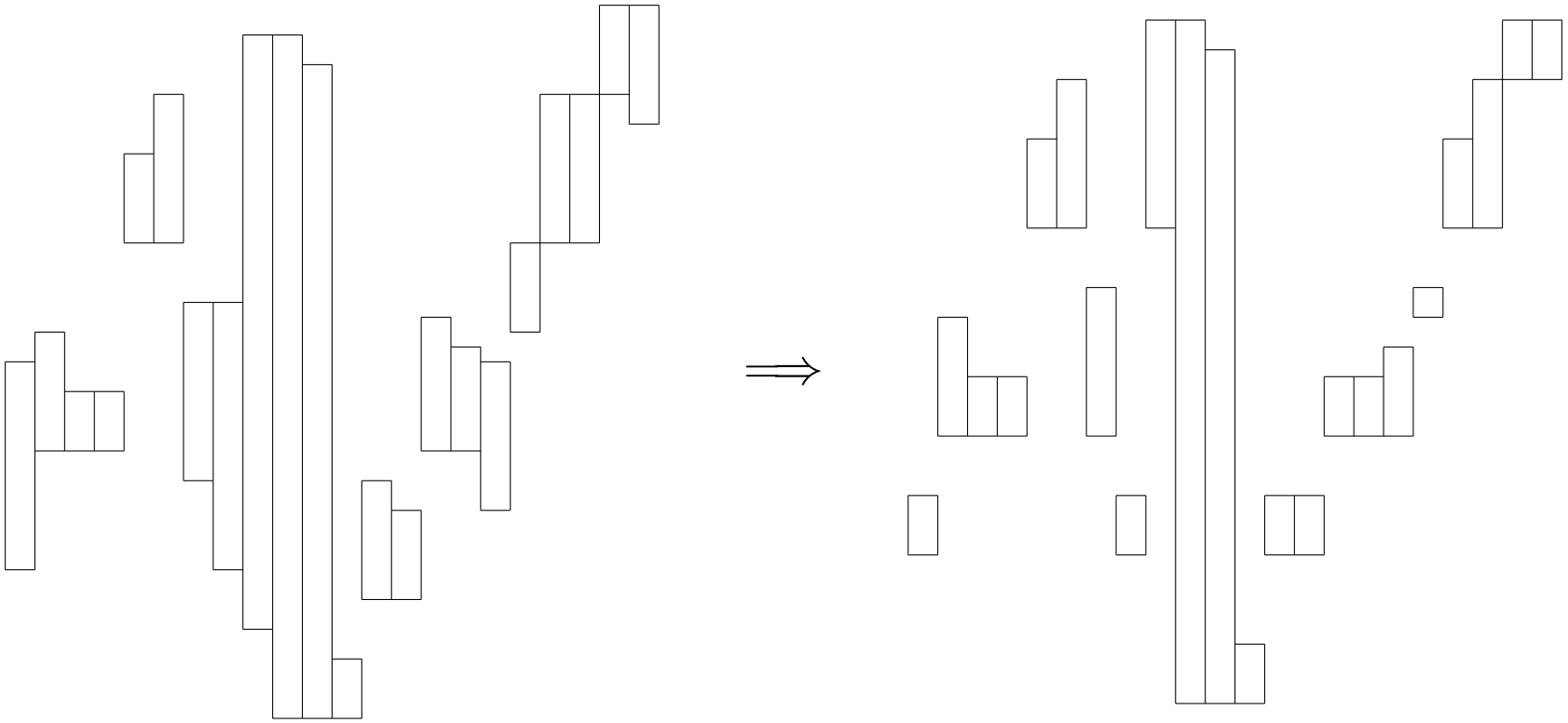
• **Problem** Given the intervals a_i, find the intervals a'_i, as small as possible, such that the two following constraints have the same solutions:

$$\textit{sort}(x_1, \ldots, x_{2n}) \ \wedge \ x_1 \in a_1 \ \wedge \ \cdots \ \wedge \ x_{2n} \in a_{2n}$$

$$\Leftrightarrow$$

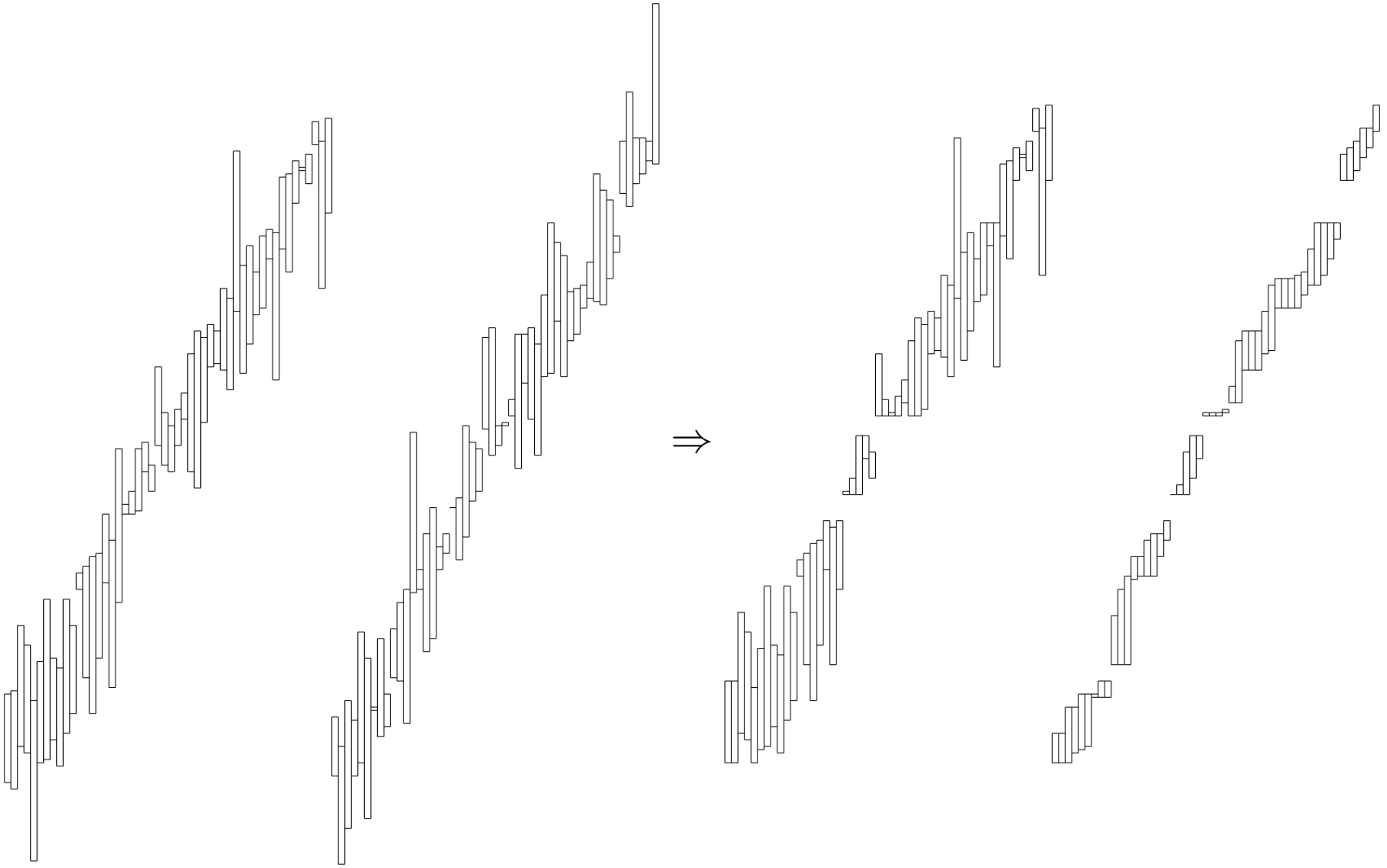
$$\textit{sort}(x_1, \ldots, x_{2n}) \ \wedge \ x_1 \in a'_1 \ \wedge \ \cdots \ \wedge \ x_{2n} \in a'_{2n}$$

Optimal interval narrowing of the sortedness constraint $2n = 22$



Noëlle Bleuzen and Alain Colmerauer, Optimal Narrowing of a Block of Sortings in Optimal time, *Constraints*, 5(1-2), pp 85-118, 2000. A preliminary version is available in `http://alain.colmerauer@free.fr`.

Optimal interval narrowing of the sortedness constraint $2n = 100$



.

Complete theories

Definition

• **Definition** A theory T, that is a set of propositions, is *complete*, if for every propostion p exactly one of the two properties holds: $T \models p$ or $T \models \neg p$.

• **How to check if a theory T is complete.** To develop an algorithm which decide whether a proposition p is such that either $T \models p$ or $T \models \neg p$.

So we have: neither $T \not\models p$ and $T \not\models \neg p$, nor $T \models p$ and $T \models \neg p$.

Extension of trees to additive ordered rationnal number

The theory of the extension into trees of ordered additive rational numbers consists in the set of the 21 following propositions:

1 $\forall \bar{x} \forall \bar{y} \, (\neg num \, f\bar{x} \wedge \neg num \, f\bar{y} \wedge f\bar{x} = f\bar{y}) \rightarrow \wedge_i \, x_i = y_i,$
2 $\forall \bar{x} \forall \bar{y} \, f\bar{x} = g\bar{y} \rightarrow num \, f\bar{x} \wedge num \, g\bar{y},$
3 $\forall \bar{x} \forall \bar{y} \, (\wedge_{i \in I} \, num \, x_i) \wedge (\wedge_{j \in J} \neg num \, y_j) \rightarrow (\exists ! \bar{z} \, \wedge_{k \in K} (\neg num \, z_k \wedge z_k = t_k(\bar{x}, \bar{y}, \bar{z}))),$
4 $\forall x \forall y \, x < y \rightarrow (num \, x \wedge num \, y),$
5 $\forall x \forall y \, num \, x + y \leftrightarrow num \, x \wedge num \, y,$
6 $\forall x \, num \, -x \leftrightarrow num \, x,$
7 $\forall \bar{x} \, tree \; h\bar{x},$
8 $\forall x \forall y \, (num \, x \wedge num \, y) \rightarrow x + y = y + x,$
9 $\forall x \forall y \forall z \, (num \, x \wedge num \, y \wedge num \, z) \rightarrow x + (y + z) = (x + y) + z,$
10 $\forall x \, num \, x \rightarrow x + 0 = x,$
11 $\forall x \, num \, x \rightarrow x + (-x) = 0,$

where f and g are two distinct function symbols of F, $h \in F - \{+, -, 0, 1\}$, x, y, z are variables, $\bar{x}$ is a vector of variables x_i, $\bar{y}$ is a vector of variables y_i, $\bar{z}$ is a vector of distinct variables z_i and where $t_k(\bar{x}, \bar{y}, \bar{z})$ is a term which begins by a function symbol $f_k \in F - \{0, 1\}$ followed by variables taken from $\bar{x}$ or $\bar{y}$ or $\bar{z}$, moreover, if $f_k \in \{+, -\}$ then $t_k(\bar{x}, \bar{y}, \bar{z})$ contains at least a variable of $\bar{y}$ or $\bar{z}$.

Extension of trees to additive ordered rationnal number

$$
\begin{array}{ll}
12_n & \forall x\, num\, x \rightarrow (nx = 0 \rightarrow x = 0), \\
13_n & \forall x\, num\, x \rightarrow \exists! y\, num\, y \wedge ny = x, \\
14 & \forall x\, num\, x \rightarrow \neg x < x, \\
15 & \forall x \forall y \forall z\, num\, x \wedge num\, y \wedge num\, z \rightarrow ((x < y \wedge y < z) \rightarrow x < z), \\
16 & \forall x \forall y\, (num\, x \wedge num\, y) \rightarrow (x < y \vee x = y \vee y < x), \\
17 & \forall x \forall y\, (num\, x \wedge num\, y) \rightarrow (x < y \rightarrow (\exists z\, num\, z \wedge x < z \wedge z < y)), \\
18 & \forall x\, num\, x\ \rightarrow (\exists y\, num\, y \wedge x < y), \\
19 & \forall x\, num\, x\ \rightarrow (\exists y\, num\, y \wedge y < x), \\
20 & \forall x\, \forall y\, \forall z\, (num\, x \wedge num\, y \wedge\ num\ z) \rightarrow (x < y \rightarrow (x + z < y + z)), \\
21 & 0 < 1.
\end{array}
$$

where n is a positive integer.

Khalil Djelloul, *Complete theories around trees.* Thèse de doctorat, Université de la Méditerranée, juin 2006.

Complexity

Theorem The time complexity of an algorithm, which decides whether a first order tree constraint (with $\exists, \forall, \neg, \vee, \wedge$), without free variables, is true , is in

$$O(\underbrace{2^{\cdots^{\left(2^{\left(2^2\right)}\right)}}}_{n}).$$

Dao Thi-Bich-Hanh and Alain Colmerauer. Expressiveness of full first order constraints in the algebra of finite or infinite trees, *Constraints*, Volume 8 Issue 3, Kluwer Academic Publishers, July 2003.

Note

For $n = 1, 2, 3, 4, 5$ we have

$$
\begin{aligned}
&2 \\
&2^2 = 4 \\
&2^{\left(2^2\right)} = 16 \\
&2^{\left(2^{\left(2^2\right)}\right)} = 65536 \\
&2^{\left(2^{\left(2^{\left(2^2\right)}\right)}\right)} = 2^{65536} > 10^{20000}
\end{aligned}
$$

The number 10^{20000} is probably much greater than the number of atoms of the universe and the number of nanoseconds which elapsed since its creation!