Global constraints and decompositions

Christian Bessiere

Univ Montpellier, CNRS

(joined work with E. Hebrard, B. Hnich, G. Katsirelos, Z. Kiziltan, N. Narodytska C.G. Quimper, T. Walsh)

Outline

- Background (CSP, propagation)
- Global constraints
- Decompositions
- Decomposability wrt AC or BC
- Non decomposability result

Constraint network

- A set of variables
  - $X = \{x_1,..,x_n\}$
- Their domains
  - $D(x_i)$: **finite** set of values for x_i
- Constraints
  - $C = \{c_1,\ldots,c_i,\ldots\}$

c_i specifies the combinations of values allowed on the sequence of variables $X(c_i)=(x_{i1},\ldots,x_{iq})$

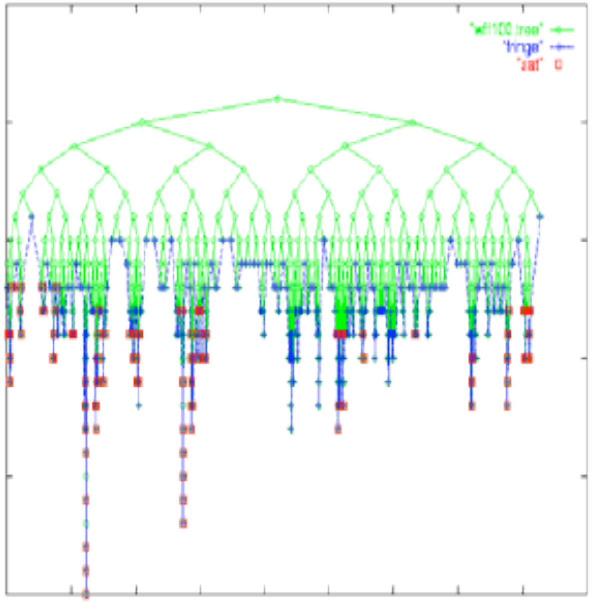
$c_i \subseteq Z^{|X(ci)|}$

$c_i = \{\text{allowed tuples on } X(c_i)\}$

So, a constraint c_i is defined by any Boolean function with domain $Z^{|X(ci)|}$

Solving a constraint problem

```
Function Solve(P)
  propagate(P)
  if empty domain then return 0
  if P fully instantiated then return 1
  select variable Xi and value v
  Xi:=v
  if Solve(P + {Xi=v}) then return 1
  return Solve(P + {Xi≠v})
```



Efficient when **propagate** reduces the search space a lot

Propagate

D(x)={0,2,4}, D(y)={1,2,3},

D(z)={~~0~~,1,2,3,4,5,6,7,~~8~~,~~9~~}

x + y = z

x+y=z
011
022
033
213
224
235
415
426
437

Propagate: arc consistency

D(x)={0,2,4}, D(y)={1,~~2~~,3},

D(z)={~~0~~,1,~~2~~,3,~~4~~,5,~~6~~,7,~~8~~,~~9~~}

x + y = z

y≠2

x+y=z
011
~~022~~
033
213
~~224~~
235
415
~~426~~
437

Optimal algorithms for arc consistency: complexity in ***O(d^r)***, where r is the number of variables of the constraint and d is the size of the domains

Global constraints

- Constraints that can involve an arbitrary number of variables
  - `Alldifferent`$(x_1,..,x_n) \Leftrightarrow x_i \neq x_j \quad \forall i,j$
  - `sum`$(x_1,..,x_n,K) \Leftrightarrow \Sigma x_i = K$

- Frequent pattern in applications: useful to express complex relations between variables
  - `Alldifferent` : two courses cannot occur simultaneously
  - `Atleast`$_{k,v}$: at least two hostesses must speak japanese
  - `Stretch` : no more than 5 working days; not morning after night --> N N R R M M A A A A R N M M R R (nurse rostering)

Why global constraints?

- Beyond their expressivity, they allow extensive propagation

  ➔ Global constraints have helped in solving open problems

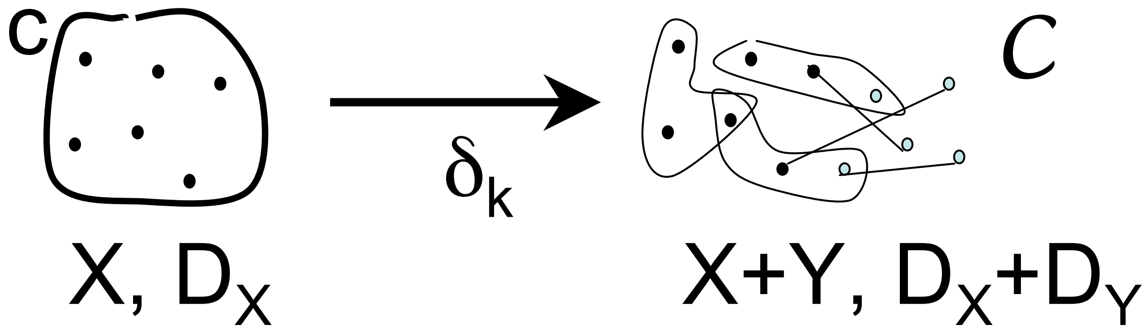
  – Sport league scheduling, etc.

- Global constraints are a specificity of CP
- Most (all?) CP solvers contain global constraints
- More than 300 global constraints in Beldiceanu's catalog
- But generic arc consistency algorithms are in $O(d^r)$…

➔ we have to implement an **ad hoc** propagator for every constraint in the solver!

Do we need 300 global constraints?

- No!
- We can rewrite them in CNF (SAT solveurs)
- We can ***decompose*** them in ‘simpler’ constraints (e.g., fixed arity)



$sol(P)=sol(\delta_k(P))[X]$

$|X(c_i)| \leq k, \forall c_i \in \mathcal{C}$

$|\delta_k(c)|$ is polynomial

Why decompositions?

- Save the time of the designer of a solver
- SMT solvers:
  - The SAT solver receives explanations from the global constraints

  ➔ it is critical to have short explanations (see yesterday's invited talk)
- Inherently incremental

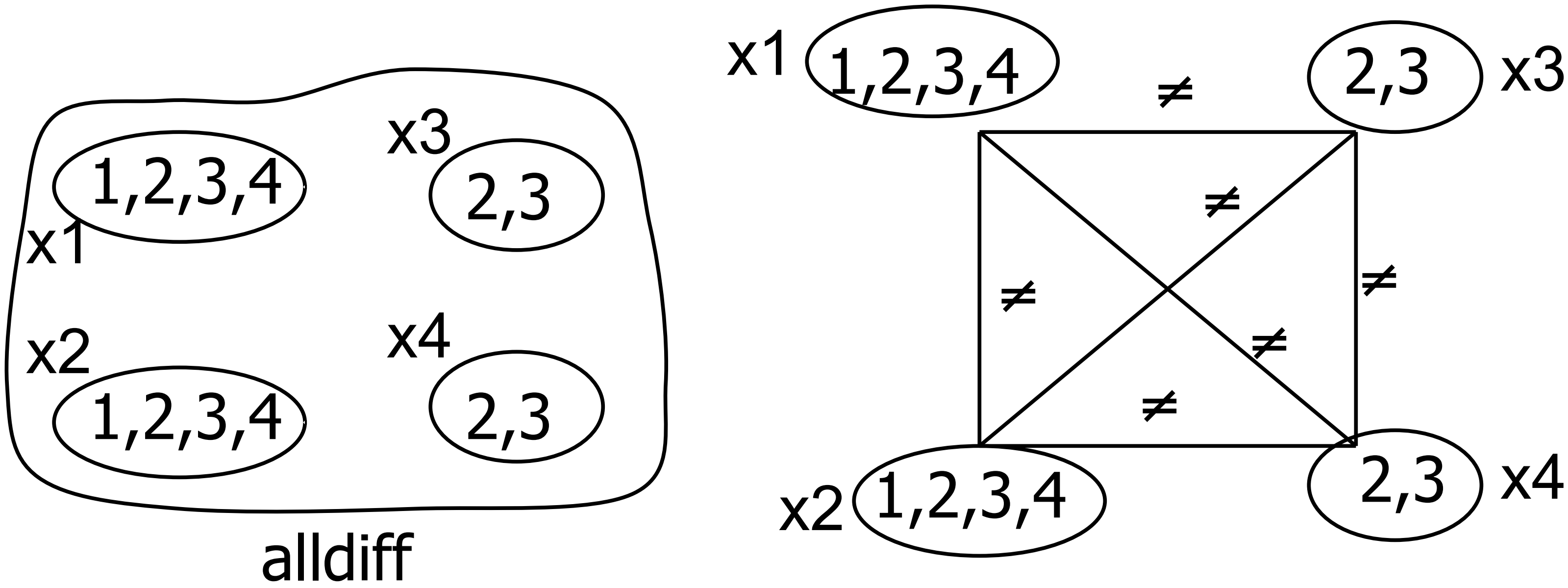
Decompositions

- What can be expected from a decomposition?
- To express the same thing
  ➔ Semantic decomposition
- To allow the same propagation (e.g., arc consistency)
  ➔ Operational decomposition

[Bessiere & Van Hentenryck 2002]

Semantic decomposability (no extra variables)

- `Alldiff`



Solutions of the CSP on the right are the same as the allowed tuples of the `Alldiff` on the left

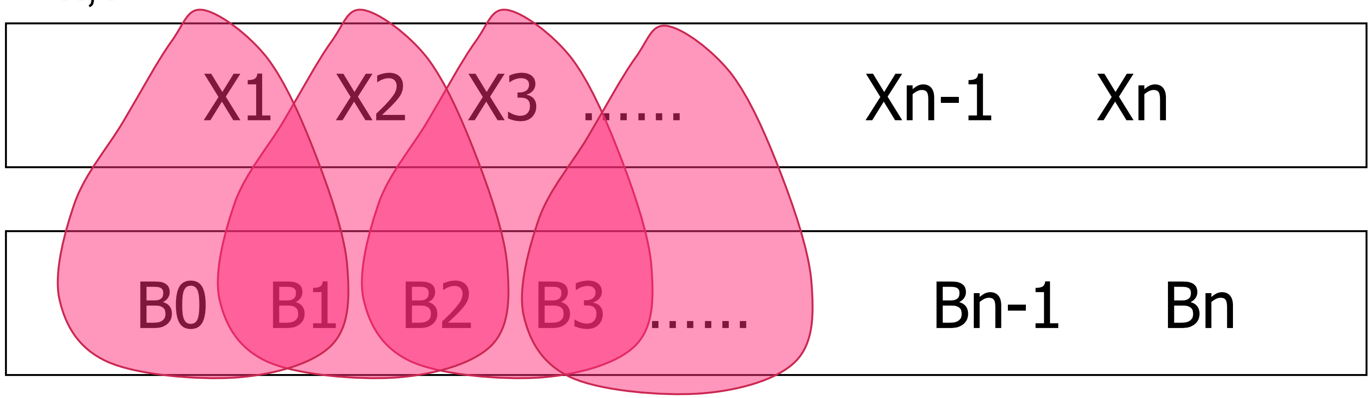
Semantic decomposability (extra variables)

- $\texttt{Atleast}_{k,v}$

X1	X2	X3	……	Xn-1	Xn

Semantic decomposability (extra variables)

- `Atleast`$_{k,v}$



- $B_0 \ldots B_n$, $D(B_i)=\{0,\ldots,n\}$
- $(x_i=v \;\&\; B_i=B_{i-1}+1) \vee (x_i \neq v \;\&\; B_i = B_{i-1})$, $\forall i$
- $B_0 = 0$, $B_n \geq k$

Solutions of this CSP projected on the Xi's
are the same as the tuples allowed by `Atleast`

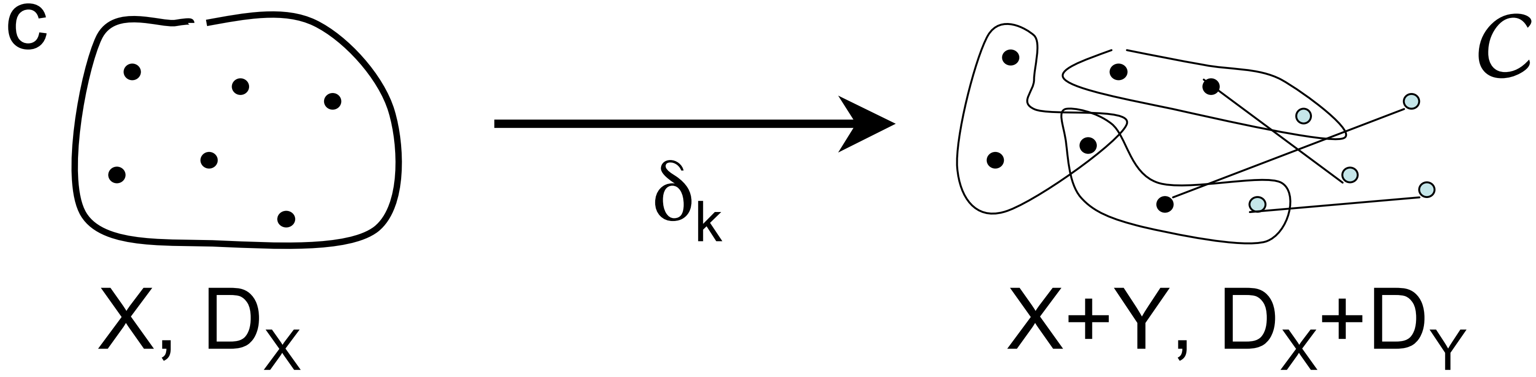
Semantic decomposability today

- Not discriminant:
  - Any polynomial Boolean function can be decided by unit propagation on a poly size CNF decomposition [Jones&Laaser74]
  - Any CNF can be expressed by constraints with fixed arity (because UP on CNF ⇔ UP on 3CNF)

➔ Any global constraint is semantically decomposable (though we don't necessarily know the decomposition --see Tuesday's best paper talk)

Operational decomposability (AC-decomposition)

- *AC-decomposition* not only preserves the semantics of the global constraint, but also the level of propagation (i.e., arc consistency)

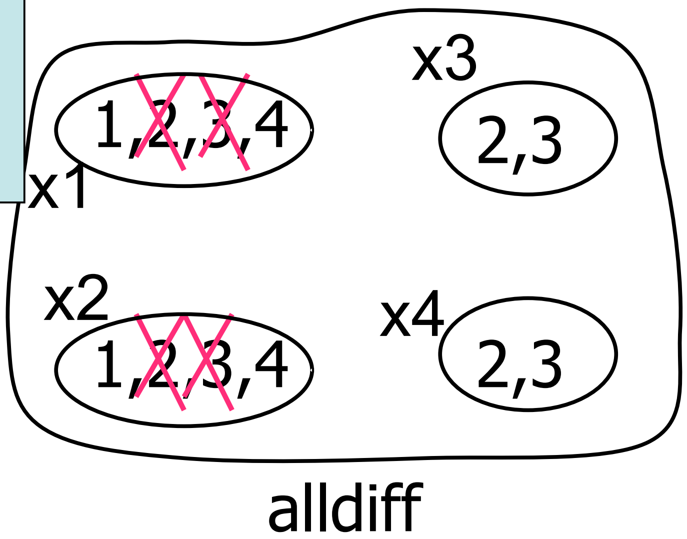


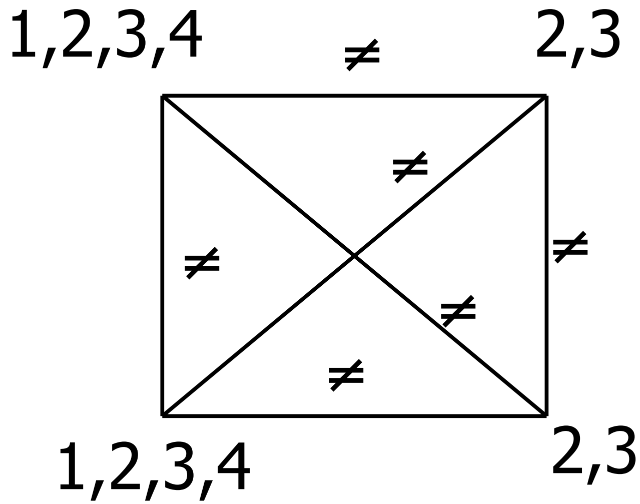
For any $D'_X \subseteq D_X$: $AC(\{c\}) = AC(\mathcal{C})|_X$

Example 1

1423
1432
4123
4132

`Alldiff`

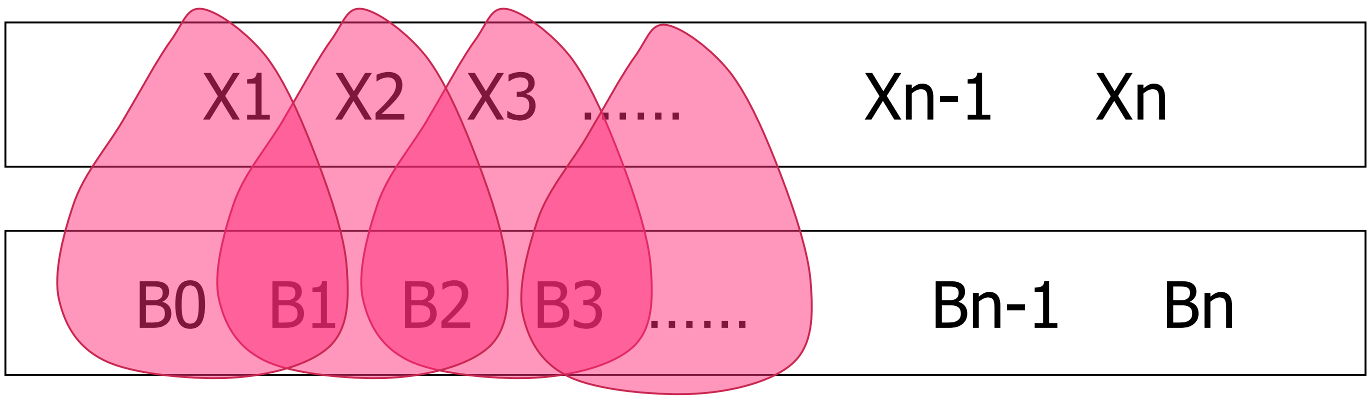




This decomposition hinders propagation

Example 2

- Atleast



- $B_0 \ldots B_n$, $D(B_i)=\{0,\ldots,n\}$
- $(x_i=v \ \& \ B_i=B_{i-1}+1) \vee (x_i \neq v \ \& \ B_i = B_{i-1})$, $\forall i$
- $B_0 = 0$, $B_n \geq k$

Acyclic hypergraph → This decomposition preserves propagation

‘Chain-like’ AC-decomposition

- Many constraints can be decomposed as a chain of ternary constraints that form a *Berge-acyclic* hypergraph (➔AC preserved)

- E.g., `Atmost`, `consecutive-1`, `lex`, `stretch`, `regular`

Taxonomy?

- Tools of computational complexity can help us

- c a global constraint on $X(c)=(x_1 \dots x_n)$

  – checker(c) ⇔ « is there a tuple in $D(x_1) \times \dots \times D(x_n)$ satisfying c ? »

- **If** checker(c) is NP-complete
  then propagate c is NP-hard
  then there is no AC-decomposition for c

NP-hard constraints

- They can be detected by polynomial reductions… and there are a lot!

- Examples:
  – `Nvalue`(N,x_1,..,x_n) (N = number of values used by x_1,..,x_n)
  – `Sum`(x_1,..,x_n,K)

- This allowed to discover that some propagators are not complete (they don't prune all arc inconsistent values)

Relax propagation: bound consistency (BC)

D(x)={0,2,4}, D(y)={1,~~2~~,3},

D(z)={~~0~~,1,2,3,4,5,6,7,~~8~~,~~9~~}

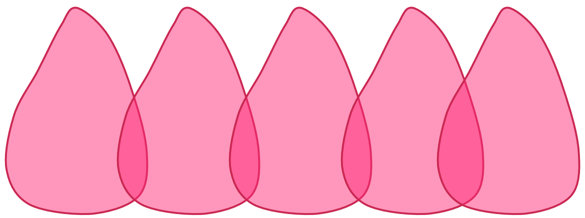
x + y = z

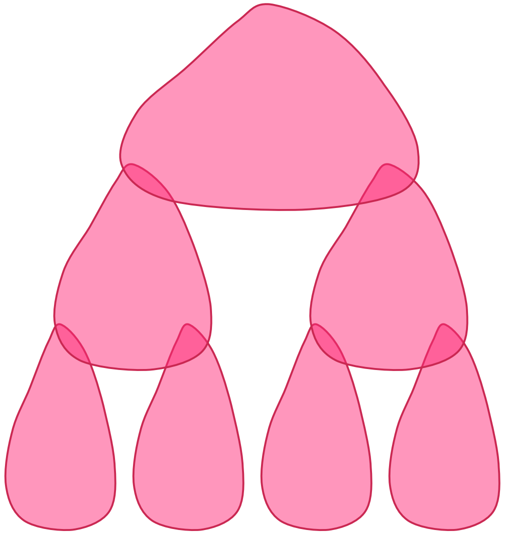
Suppose 2 removed from D(y)

x+y=z
011
022
033
213
224
235
415
426
437

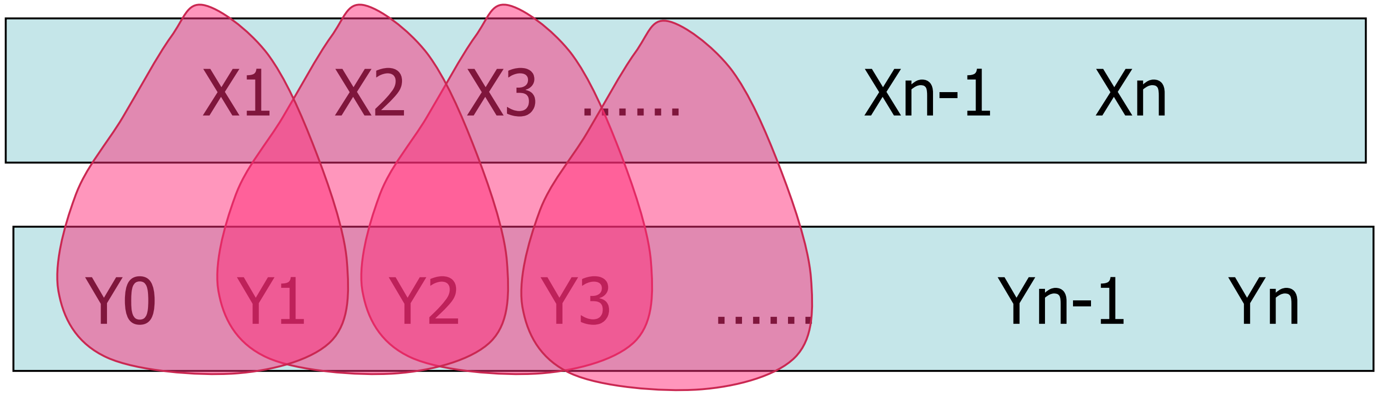
BC-decompositions

- Several common constraints for which AC is NP-hard allow BC-decompositions in ternary constraints arranged as a chain (`sum`) or a pyramid (`Nvalue`, see tomorrow's talk)





Warning: size of the 'gadget' Example: sum(x_1,..,x_n,K)



- $Y_i = Y_{i-1} + X_i$, $\forall i$
- $Y_0 = 0$, $Y_n = K$
- $D(Y_i)$= ???

$D(X_1)=\{0,1,\ldots 9\}$; $D(X_2)=\{0,10,\ldots 90\}$; $D(X_3)=\{0,100,\ldots 900\}$; $D(X_4)=\{0,1000,\ldots 9000\}$ . . .

➔ For AC, $D(Y_4)$ must contain 10^4 values ➔ exponential size

➔ BC can use the interval domain [0,..,999]

Until now we have:

- Constraints NP-hard to propagate

  ➔ no AC-decomposition

  (sometimes a BC-decomposition)

- Constraints polynomial to propagate

  ➔ AC-decomposition when we find one
  (`atleast`, `stretch`, etc.)

  ➔ And the others???

Non AC-decomposability result

- AC-decomposition for c
  ⇔ decomposition into CNF which computes AC
  [Bessiere, Hebrard, Walsh 2003]
  ⇔ CNF checker (= decides if the constraint has a solution tuple)

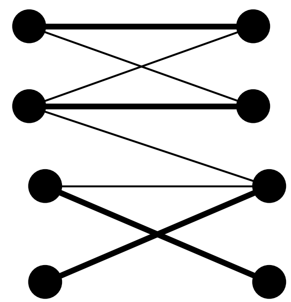
- CNF checker
  ⇒ monotone circuit of polynomial size

Theorem no poly-size monotone circuit ⇒ no AC-decomposition (and no CNF computes AC)

Circuit complexity

- Classes of functions that cannot be computed by monotone circuits of poly size [Rasborov 85, Tardos 88]

- Example:
  - perfect matching
  - Subsumed by checker(`alldiff`) [Knuth92, Regin94]



$x_1 \in \mathbf{1},2$
$x_2 \in 1,\mathbf{2},3$
$x_3 \in 3,\mathbf{4}$
$x_4 \in \mathbf{3}$

➔ `alldiff` has no AC-decomposition

- Other examples: `gcc`, `Nvalue`, etc.

So what?

- Constraint programming ***cannot*** be reduced to CNF (i.e., to SAT)

- Constraint programming ***cannot*** be reduced to constraints with fixed arity

Summary

- NP-hard constraints
  – Use a lower level of consistency
- AC-decomposable constraints
  – Use the decomposition (when we know it!)
- Constraints that are poly but non AC-decomposable
  – ***You must implement the poly algorithm :(***
  – …or use a lower level of consistency

Canonical language?

- Idea: provide solvers with a set $\mathcal{L}$ of a few (a dozen?) of global constraints that would encode all others
  - AC($\mathcal{L}$)-decomposability of c:
    - ➔ c can be decomposed into constraints of $\mathcal{L}$
    - ➔ No new propagator to implement!

- Examples:
  - `range` + `roots` can easily express around 70 constraints in the catalog (version with 214 constraints)
  - `slide` (or Beldiceanu's counter constraint [Beldiceanu et al. 2004]) expresses many others
  - Extending the result in CP'10 best paper would help!







Some references

Deriving Filtering Algorithms from Constraint Checkers
N.Beldiceanu, M. Carlsson, T. Petit.
Proceedings CP'04, Toronto CA, pages 107-122.

Global Constraint Catalogue: Past, Present and Future
N. Beldiceanu, M. Carlsson, S. Demassey, T. Petit.
Constraints 12 (1): 21-62 (2007).

The Problem of Compatible Representatives
D.E. Knuth, A. Raghunathan.
SIAM J. Discrete Math. 5(3): 422-427 (1992).

A Filtering Algorithm for Constraints of Difference in CSPs
J.C. Régin.
Proceedings AAAI'94, Seattle WA, pages 362-367.

To be or not to be...a global constraint
C. Bessiere and P. Van Hentenryck.
Proceedings CP'03, Kinsale, Ireland, pages 789-794.

Local consistencies in SAT
C. Bessiere, E. Hebrard and T. Walsh.
Proceedings SAT'03, LNCS 2919, pages 299-314.

The Complexity of Reasoning with Global Constraints
C. Bessiere, E. Hebrard, B. Hnich, T. Walsh.
Constraints, volume 12(2), pages 239-259 (2007).

Range and Roots: Two Common Patterns for Specifying and Propagating Counting and Occurrence Constraints
C. Bessiere, E. Hebrard, B. Hnich, Z. Kiziltan, T. Walsh.
Artificial Intelligence, volume 173(11), pages 1054-1078 (2009).

Circuit Complexity and Decompositions of Global Constraints
C. Bessiere, G. Katsirelos, N. Narodytska, T. Walsh.
Proceedings IJCAI'09, Pasadena CA, pages 412-418.

Decompositions of All Different, Global Cardinality and Related Constraints
C. Bessiere, G. Katsirelos, N. Narodytska, C.G. Quimper, T. Walsh.
Proceedings IJCAI'09, Pasadena CA, pages 419-424.

SLIDE: A Useful Special Case of the CARDPATH Constraint
C. Bessiere, E. Hebrard, B. Hnich, Z. Kiziltan, T. Walsh.
Proceedings ECAI'08, Patras, Greece, pages 475-479.