La mémoire retrouvée: promenade en PPC liée à mon temps passé à Uppsala

Nicolas Beldiceanu

`nicolas.beldiceanu@imt-atlantique.fr`

IMT Atlantique, LS2N (CNRS)

Nantes, France

20 August 2025

NordConsNet, Uppsala

Ary Murnu

"*Cock-a-doodle-doo! My Lord Boyar, Give Me Back My Two-Penny Purse!!*"
– Ion Creangă, *Tale*

Purpose

Illustrate:

- My collaboration with Mats through an **intertwined chain of events** from 1986 to 2025.
- The pleasure of **shared moments** of discovery.
- The need to **dismantle** what was taken for granted (to build something simpler).

Purpose

Illustrate:

- My collaboration with Mats through an **intertwined chain of events** from 1986 to 2025.
- The pleasure of **shared moments** of discovery.
- The need to **dismantle** what was taken for granted (to build something simpler).

Dwelling only on the past is boring,
focusing only on the present is arrogant,
only by **linking** the two do we find clarity.

Interlude

"*A research result leading to a communication or publication in a given year is often the product of many years of research, trial and error, progress and mistakes that have been corrected, but which have enabled progress to be made.*"

– Dominique Glaymann

Interlude

"*A research result leading to a communication or publication in a given year is often the product of many years of research, trial and error, progress and mistakes that have been corrected, but which have enabled progress to be made.*"

– Dominique Glaymann

- **1999**: manual map of graph invariants at SICS
  → *acquiring maps of sharp bounds* [CP 2022]
- **2000**: minor enhancement of alldifferent in SICStus
  → *scalable GAC for alldifferent* [IJCAI 2025]
- **2003**: meta-data of the global constraint catalogue
  → *Model Seeker* [CP 2012]

Circumstances that led me to Uppsala

- 198?: Visit of SICS at ECRC (*Seif and ?*)
- 1987: Contact with ECRC (*they reviewed my 1st paper*)
- 1988: I join ECRC
- 1990: Mats supported our ICLP paper (*with Abder*)
- 05/1999: Abder suggested contacting Mats wrt academy
- 06/1999: Went for an interview at SICS (*Seif, Mats, Per*)
- 07/1999: Visit Stockholm with family (*and met Mats*)
- 11/1999: Join SICS in Uppsala

Contents

Part 1: Memory regained

Two things were made explicit:

- It is useful to describe the meaning of constraints **independently** from their use.
- Constraints are not just algorithms but **modelling** tools (independent from CP, MIP, SAT).

Memory regained

- Introduce global constraints at ECRC
  (*an example of introduction of global constraints in* CHIP:
  application to block theory problems) **[TR-LP-49,1990]**

- Introduce a few global constraints before joining SICS
  [Math Comput. Modelling, 1993, 1994]
  (*but from my first day at* SICS,
  I started with my memory and a blank page)

- What should I start with ?
  - Went back to my classics, Laurière and Pitrat:
    (*toward efficiency through generality*) **[IJCAI 1979]**
  - Rediscovered by many persons:
    "*general methods that leverage computation are ultimately the most effective*" – Rich Sutton, 2019

Global constraint catalogue ancestor

To regain memory, I decided to:

- Identify useful modelling abstractions
- Define their meaning explicitly
  (*independently from* CP, MIP, SAT)
- Find ways to synthesise code from this description
  (*initially described with graph properties*)

Lead to a first textual version in 2000 of the catalogue
[SICS Technical Report, T2000:01]

Global constraint catalogue

- Mats defined the electronic version of the catalogue and wrote a program that produced the textual version of the catalogue.
- Each constraint was described by:
  - Many Prolog facts for various aspects of a constraint,
  - A textual LaTeX part consisting of several fields.
- Constraint meaning described by metadata:
  - graph properties,
  - finite automata, register automata, transducers,
  - first-order logic formulae,
  - reformulation.

Metadata used in the Model Seeker (with Helmut), [CP2012].

Current status of the catalogue

- Less effort into the catalogue.
- More effort to develop an atlas of sharp bounds over combinatorial objects:
  - digraph,
  - tree,
  - forest,
  - permutation,
  - partition,
  - sequence,
  - cyclic sequence,
  - time series.

consists of maps describing sharp bounds of a combinatorial object's characteristics along with their relations

Part 2: the automata chronicle

Three things that were made explicit:

- An automaton can **replace** a dedicated hand-crafted filtering algorithm.
- Making explicit the **meaning of the transitions** of an automaton simplifies things.
- Using libraries of algorithms on automata is useful for **checking properties** of regexp in the context of quantitative regexp and cyclic automata.

How it all began

- 2001: Visit of the York group to Uppsala, where the topic of symmetry constraints arose, *I was not there*.
- 2002: A. M. Frisch, *et al.* (global constraints for lexicographic orderings) [CP] *dedicated filtering algorithm*.

How it all began

- 2001: Visit of the York group to Uppsala, where
  the topic of symmetry constraints arose,
  I was not there.

- 2002: A. M. Frisch, *et al.*
  (global constraints for lexicographic orderings) [CP]
  dedicated filtering algorithm.

- **Question**: How can we replace an ad hoc filtering algorithm with some method derived from some first principle?

How it all began

- 2001: Visit of the York group to Uppsala, where
  the topic of symmetry constraints arose,
  I was not there.

- 2002: A. M. Frisch, *et al.*
  (global constraints for lexicographic orderings) [CP]
  dedicated filtering algorithm.

- **Question**: How can we replace an ad hoc filtering algorithm with some method derived from some first principle?

- **Answer** : Mats and myself, [ESOP 04/2004]
  (from constraints to finite automata to filtering algorithms)
  automaton based filtering for lex ordering and lex_chain.

How it continued

- Mats and myself, [CP 2004]
  (deriving filtering algorithms from constraint checkers)
  - Choose to use registers
    alternative wrt encoding registers in states
  - Reformulation rather than a dedicated algorithm
    preserves GAC when no register

How it continued

- Mats and myself, [CP 2004]
  (deriving filtering algorithms from constraint checkers)
  - Choose to use registers
    alternative wrt encoding registers in states
  - Reformulation rather than a dedicated algorithm
    preserves GAC when no register

- Pesant, [CP 2004]
  (a regular language membership constraint)
  dedicated filtering algorithm that unfolds the automaton

Development: from automata to transducers

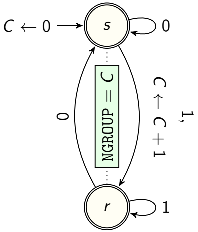
Mats encodes many constraints on sequences of the catalogue using automata with registers.

- A fun programming exercise: create **constant-size** automata with the **fewest** registers possible.

Development: from automata to transducers

Mats encodes many constraints on sequences of the catalogue using automata with registers.

- A fun programming exercise: create **constant-size** automata with the **fewest** registers possible.

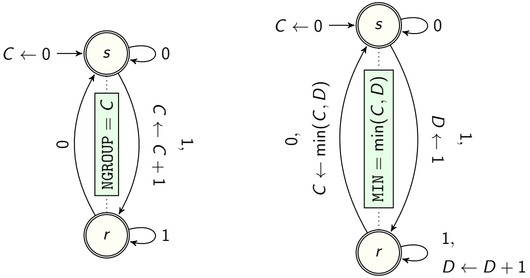


Automata for the group constraint: number of groups

Development: from automata to transducers

Mats encodes many constraints on sequences of the catalogue using automata with registers.

- A fun programming exercise: create constant-size automata with the fewest registers possible.

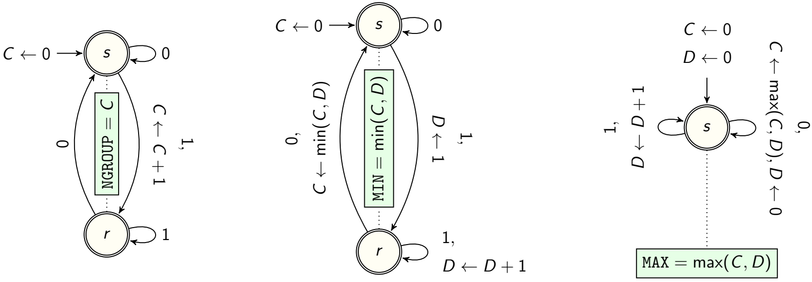


Automata for the group constraint: number of groups, smallest size

Development: from automata to transducers

Mats encodes many constraints on sequences of the catalogue using automata with registers.

- A fun programming exercise: create **constant-size** automata with the **fewest** registers possible.



Automata for the group constraint: number of groups, smallest size, biggest size

From automata to transducers (the aha moment)

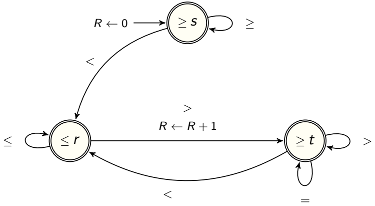
- Transitions carry an implicit meaning
- Making this meaning explicit simplifies things
  (*mentions of arcs corresponding to mismatches in few papers*)

From automata to transducers (the aha moment)

- Transitions carry an implicit meaning
- Making this meaning explicit simplifies things
  (*mentions of arcs corresponding to mismatches in few papers*)
- We realised ☺ that there was a **common pattern** in Mats manual register automata
  - Could give a semantic to the transitions: represent the **discovery phases** of a pattern.
  - Could implement this as the **output alphabet of a transducer**.

From automata to transducers (the aha moment)

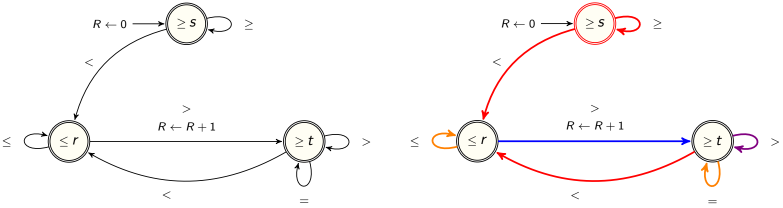
- Transitions carry an implicit meaning
- Making this meaning explicit simplifies things (*mentions of arcs corresponding to mismatches in few papers*)
- We realised ☺ that there was a **common pattern** in Mats manual register automata
  - Could give a semantic to the transitions: represent the **discovery phases** of a pattern.
  - Could implement this as the **output alphabet of a transducer**.



Automaton for counting the number of peaks, i.e. ‘ $<(<\mid=)^*(>\mid=)^*>$ ’

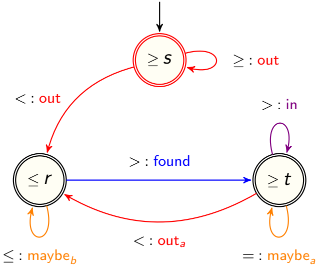
From automata to transducers (the aha moment)

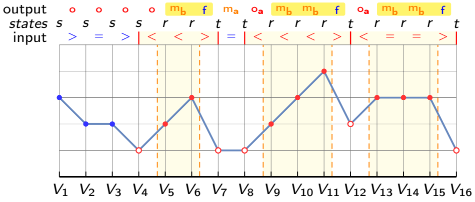
- Transitions carry an implicit meaning
- Making this meaning explicit simplifies things (*mentions of arcs corresponding to mismatches in few papers*)
- We realised ☺ that there was a **common pattern** in Mats manual register automata
  - Could give a semantic to the transitions: represent the **discovery phases** of a pattern.
  - Could implement this as the **output alphabet of a transducer**.



Automaton for counting the number of peaks, i.e. ‘ $<(<\mid=)^*(>\mid=)^*>$ ’

From automata to transducers (example of peak, i.e. $`<(<|=)^*(>|=)^*>'$)





From automata to transducers (results)

Led to the time-series catalogue

- 2016: Beldiceanu, Carlsson, *et al.*
  (Using finite transducers for describing and synthesising structural time-series constraints)
  [Constraints]
- 2016: Arafailova, *et al.*
  (Global Constraint Catalogue, Vol. II, Time-Series Constraints)
  [CoRR]

Getting bogged down with extensions (and hitting walls)

- As we could not handle all patterns we spent a lot of effort trying to extend the phase letters with some limited success.
- But we hit two walls:
  - Handling patterns containing **many disjunctions**.
  - Resynchronise the computation in **constant time** (register update) when we have a mismatch.
- In 2019, a colleague of mine, Hervé Grall, discovered a novel simple transducer model solving the problem with the first wall (*I implemented his approach in SICStus during Covid*)

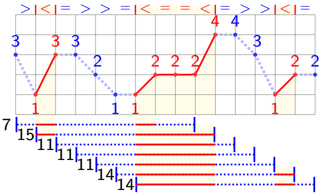
A library on regexp and automata in SICStus with two unusual applications

- 2015-2025: my course on modelling with automata at IMT
- 2017 : SICStus library on regexp/automata (by Mats)
- 2017 : quantitative regexp (*checking properties on regexp*)
- 2025 : circular automaton (*started with a question in a master's internship*)

A library on regexp and automata (checking properties)

- Question: come up with $\theta(n)$ **checkers** for sliding time series constraints.

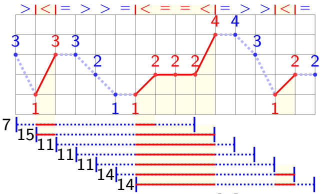
- Example : check the sum of increasing sequences, i.e. '$<(<|=)^*<|<$', in every time window of size 10.



A library on regexp and automata (checking properties)

- Question: come up with $\theta(n)$ **checkers** for sliding time series constraints.

- Example : check the sum of increasing sequences, i.e. '$<(<|=)^*<|<$', in every time window of size 10.



Many properties of regexp allow one to come up with efficient algorithms

Example: convexity of a regexp (definition)

Definition

$\mathcal{L}$ is *convex* if for any word $w = s_1 s_2 \ldots s_{n-1}$ in $\mathcal{L}$ and for any pair of factors $u = s_c s_{c+1} \ldots s_d$ and $v = s_e s_{e+1} \ldots s_f$ of w such that, both u and v are words in $\mathcal{L}$, $s_{\min(c,e)} s_{\min(c,e)+1} \ldots s_{\max(d,f)}$ is also in $\mathcal{L}$.

Example

The langage ' $< (< \mid =)^*(> \mid =)^* >$ ' describing a peak is convex.

Example: convexity of a regexp (definition)

Definition

$\mathcal{L}$ is *convex* if for any word $w = s_1 s_2 \ldots s_{n-1}$ in $\mathcal{L}$ and for any pair of factors $u = s_c s_{c+1} \ldots s_d$ and $v = s_e s_{e+1} \ldots s_f$ of w such that, both u and v are words in $\mathcal{L}$, $s_{\min(c,e)} s_{\min(c,e)+1} \ldots s_{\max(d,f)}$ is also in $\mathcal{L}$.

Example

The langage ' $< (< \mid =)^* (> \mid =)^* >$ ' describing a peak is convex.

Most patterns of the time-series catalogue are convex.

Example: convexity of a regexp (proof)

Definition

$\mathcal{L}$ is *convex* if for any word $w = s_1 s_2 \ldots s_{n-1}$ in $\mathcal{L}$ and for any pair of factors $u = s_c s_{c+1} \ldots s_d$ and $v = s_e s_{e+1} \ldots s_f$ of w such that, both u and v are words in $\mathcal{L}$, $s_{\min(c,e)} s_{\min(c,e)+1} \cdots s_{\max(d,f)}$ is also in $\mathcal{L}$.

Example: convexity of a regexp (proof)

Definition

$\mathcal{L}$ is *convex* if for any word $w = s_1 s_2 \ldots s_{n-1}$ in $\mathcal{L}$ and for any pair of factors $u = s_c s_{c+1} \ldots s_d$ and $v = s_e s_{e+1} \ldots s_f$ of w such that, both u and v are words in $\mathcal{L}$, $s_{\min(c,e)} s_{\min(c,e)+1} \cdots s_{\max(d,f)}$ is also in $\mathcal{L}$.

- Sketch for proving $\mathcal{L}$ is convex using the library on regexp, where Σ is the alphabet of $\mathcal{L}$, and s a letter not in Σ.
  (*case when* u *and* v *are disjoint: look for counter-example*)

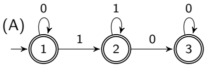
$$\bigcap \begin{pmatrix} \text{shuffle}(\text{shuffle}(\mathcal{L}, s), s) \\ \Sigma^* s \;\; \mathcal{L} \; \Sigma^* \; \mathcal{L} \;\; s \, \Sigma^* \\ \Sigma^* s \, (\Sigma^+ \setminus \mathcal{L}) \, s \, \Sigma^* \end{pmatrix} = \emptyset$$

Results

- N. Beldiceanu, M. Carlsson, *et al.*
  (classifying pattern and feature properties to get a $\theta(n) \cdots$) **[CoRR 2019]**
- A. Hien, *et al.*
  (automata based multivariate time series analysis $\cdots$) **[ITISE 2023]**

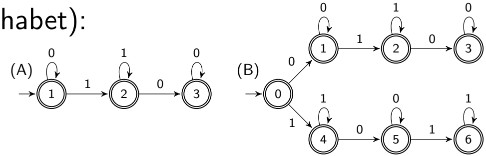
A library on regexp and automata (cyclic automata)

- Context: Cyclic constraints are often mentioned but not systematically defined or addressed.
- Problem: compute the cyclic automaton of a finite automaton (*last position of a sequence is adjacent to the first position*)
- Examples (with 0/1 alphabet):
  - global_contiguity



A library on regexp and automata (cyclic automata)

- Context: Cyclic constraints are often mentioned but not systematically defined or addressed.
- Problem: compute the cyclic automaton of a finite automaton (*last position of a sequence is adjacent to the first position*)
- Examples (with 0/1 alphabet):
  - global_contiguity

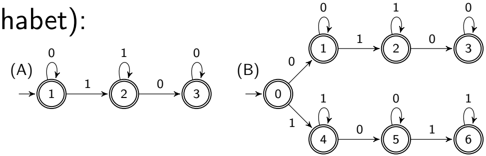


A library on regexp and automata (cyclic automata)

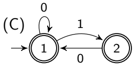
- Context: Cyclic constraints are often mentioned but not systematically defined or addressed.

- Problem: compute the cyclic automaton of a finite automaton (*last position of a sequence is adjacent to the first position*)

- Examples (with 0/1 alphabet):
  - global_contiguity



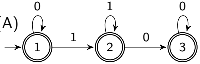
  - forbidden pattern "11"



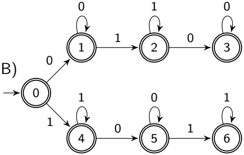
A library on regexp and automata (cyclic automata)

- Context: Cyclic constraints are often mentioned but not systematically defined or addressed.
- Problem: compute the cyclic automaton of a finite automaton (*last position of a sequence is adjacent to the first position*)
- Examples (with 0/1 alphabet):
  - global_contiguity

(A)

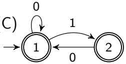


(B)

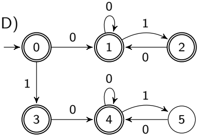


  - forbidden pattern “11”

(C)

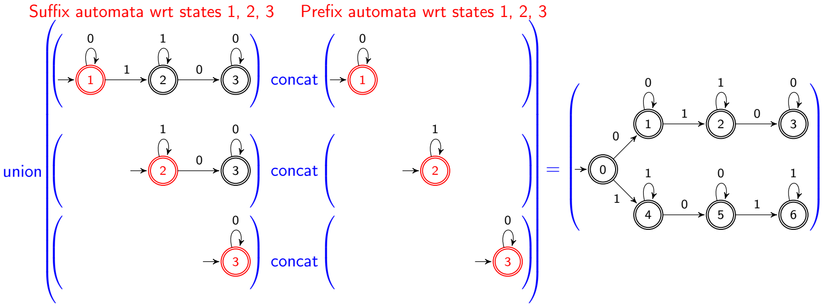


(D)



A library on regexp and automata

(e.g., computing the cyclic automata for global_contiguity)



Generating cyclic automata (remarks)

- Building the cyclic automata by hand is horrible (*as states have to record **how the sequence starts** to know **how the sequence can end***).
- If the original automaton has n states, the cyclic version may have $O(n^2)$ states.
- Use SICStus library on regexp to generate circular automata (*an hour for generating an automaton with 80000 states*).
- Automata associated with global constraints are **structured** (*exploiting this structure may very likely speed up many operations on automata*).

Part 3: the double life of alldifferent

One thing we made explicit:

- Don't be a prisoner to what you've learned during your early studies (Tarjan algorithm).

The double life of alldifferent a flood of events, whether related or unrelated, visible or not, ⋯ and the curse of DFS

- **1970**: Berge (edges belonging to a maximum matching) **[Graphes]**
- **1972**: Tarjan (DFS and linear graph algorithms) **[SIAM J. Comp]**
- **1973**: Hopcroft, Karp (an $n^{\frac{5}{2}}$ ⋯ graphs) **[SIAM J. Comp]**
- **1976**: Laurière (Alice) **[HDR]**
- **1994**: Régin (GAC for alldiff) **[AAAI]**
- **2002**: Dahlhaus, *et al.* (Partially complemented representations of digraphs) **[Discrete Math. Theor. Comput. Sci.]**
- **2008**: Gent, *et al.* (GAC for alldiff: empirical survey) **[Constraints]**
- **2018-2023**: (4 papers for enhancing GAC for alldiff) **[IJCAI]**
- **2023**: Tardivo, *et al.* (GPU for GAC alldiff) **[J. Log. Computing]**

The double life of alldifferent
a stream of widely spaced events

- **2000**: Carlsson, myself (hook in the SICStus GAC alldiff) **SICStus**
- **2025**: Le Bozec-Chiffoleau, *et al.* (Scalable GAC for alldiff) **[IJCAI]**

GAC for alldifferent: hook for computing scc in SICStus in 2000

- In some cases no need to explore all arcs:
  if a DFS builds a single path visiting all nodes and comes back to the initial node, find one scc in $O(n)$ rather than in $O(m)$ (*save time for dense graphs*).

Scalable GAC for alldifferent in 2025

- Le Bozec-Chiffoleau, *et al.*, [IJCAI]
  (bimodal depth-first search for scalable GAC for alldifferent)

- Theoretical worst-case complexity of bimodal DFS:
  $\mathrm{O(n + \tilde{m})}$
  where $\tilde{m}$ is the sum, for each vertex v,
  of the minimum between
  - the numbers of successors and
  - the non-successors of v.

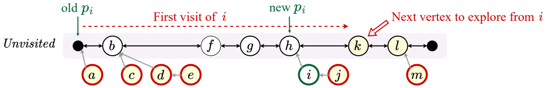
Our approach: key question and key idea

- Start from the edge classification in a DFS.
- Knew that:
  - Need to **accelerate the visit of unvisited nodes** in a DFS.
  - The graph needs to be represented **implictly** from the domains and matching.
  - Can compute the scc during the post visit of each node.
- Key question:
  how to efficiently scan over unvisited successors of a node v ?
- Key idea: for each node v, **dynamically** choose between
  - Iterating over the successors of v: explore the unvisited ones.
  - iterating over the unvisited ones : find the successors of v.
- How: use a data structure (tracking list) to handle the unvisited nodes.

Illustration of the key idea of bimodal DFS (example from Sulian)

Example: build a DFS-tree from vertex (a) (focus on vertex (i)).

First visit of (i) $\rightarrow N^+(i) = \{a, c, d, e, j, k, l, m\}$ and $Unvisited = \{b, f, g, h, k, l\}$.

> $|Unvisited| < |N^+(i)| \implies$ **Iterate over** $\boldsymbol{Unvisited}$ **instead of** $\boldsymbol{N^+(i)}$
> to find the vertices to explore from (i).
>
> **Record with** $\boldsymbol{p_i}$ **the last node traversed in** $\boldsymbol{Unvisited}$
> before finding the next unvisited successor of (i) to explore.
>
> 
>

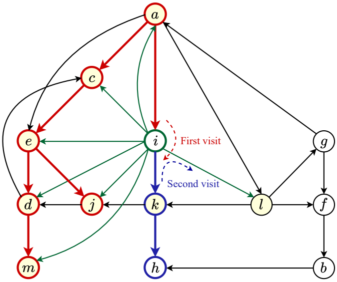
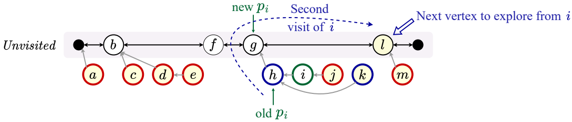
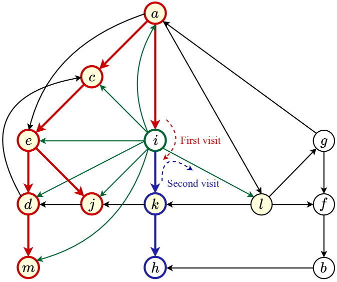


Illustration of the key idea of bimodal DFS

> to avoid iterating from scratch. (Repeat this for future visits)
>
> 
>



Experiments using CHOCO (Sulian, Charles)

Mainly carried by alldifferent for which **the size is gradually increased**.

Experiments using CHOCO (Sulian, Charles)

Mainly carried by alldifferent for which **the size is gradually increased**.

Four strategies for our bimodal approach:

Name	**Iterate over** Unvisited **when**
CLASSIC	Never
COMP	Always
PARTIAL	$\lvert Unvisited \rvert < \lvert D(x) \rvert$
TUNED	$\sqrt{\lvert Unvisited \rvert} < \lvert D(x) \rvert$

Experiments using CHOCO (Sulian, Charles)

Mainly carried by alldifferent for which **the size is gradually increased**.

Four strategies for our bimodal approach:

Name	**Iterate over** Unvisited **when**
CLASSIC	Never
COMP	Always
PARTIAL	$\|Unvisited\| < \|D(x)\|$
TUNED	$\sqrt{\|Unvisited\|} < \|D(x)\|$

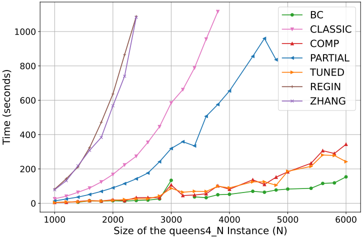
Three built-in algorithms in CHOCO:
REGIN, **ZHANG** and **BC**

Experiments using CHOCO (Sulian, Charles)

Mainly carried by alldifferent for which **the size is gradually increased**.

Four strategies for our bimodal approach:

Name	**Iterate over** Unvisited **when**
CLASSIC	Never
COMP	Always
PARTIAL	$\|Unvisited\| < \|D(x)\|$
TUNED	$\sqrt{\|Unvisited\|} < \|D(x)\|$



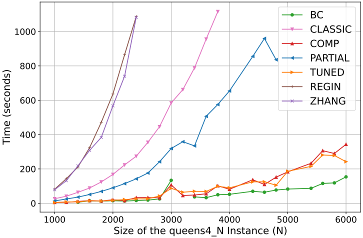
Three built-in algorithms in CHOCO:
REGIN, **ZHANG** and **BC**

Experiments using CHOCO (Sulian, Charles)

Mainly carried by alldifferent for which **the size is gradually increased**.

Four strategies for our bimodal approach:

Name	**Iterate over** Unvisited **when**
CLASSIC	Never
COMP	Always
PARTIAL	$\|Unvisited\| < \|D(x)\|$
TUNED	$\sqrt{\|Unvisited\|} < \|D(x)\|$



Three built-in algorithms in CHOCO:
REGIN, ZHANG and BC

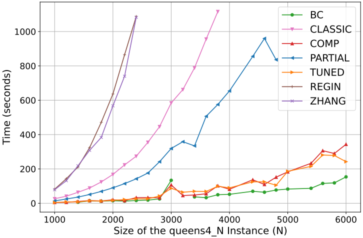
COMP and TUNED are close to the speed of BC and solve more problems!

Experiments using CHOCO (Sulian, Charles)

Mainly carried by alldifferent for which **the size is gradually increased**.

Four strategies for our bimodal approach:

Name	**Iterate over** Unvisited **when**
CLASSIC	Never
COMP	Always
PARTIAL	$\lvert Unvisited \rvert < \lvert D(x) \rvert$
TUNED	$\sqrt{\lvert Unvisited \rvert} < \lvert D(x) \rvert$



Three built-in algorithms in CHOCO:
REGIN, ZHANG and BC

COMP and TUNED are close to the speed of BC and solve more problems!

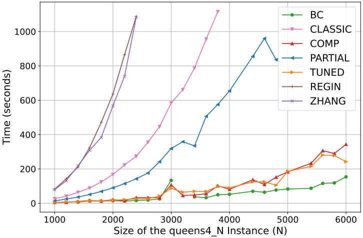
COMP and TUNED offer **growing speedups** compare to the other GAC algorithms **as the input size increases!**

Experiments using CHOCO (Sulian, Charles)

Mainly carried by alldifferent for which **the size is gradually increased**.

Four strategies for our bimodal approach:

Name	**Iterate over** Unvisited **when**
CLASSIC	Never
COMP	Always
PARTIAL	$\lvert Unvisited \rvert < \lvert D(x) \rvert$
TUNED	$\sqrt{\lvert Unvisited \rvert} < \lvert D(x) \rvert$



Three built-in algorithms in CHOCO:
REGIN, **ZHANG** and **BC**

COMP and **TUNED** are close to the speed of **BC** and solve more problems!

COMP and **TUNED** offer **growing speedups** compare to the other GAC algorithms **as the input size increases!**

GAC may now be chosen over BC as the default consistency level for propagating alldifferent!

Post-Mortem Analysis

- The bottleneck for computing the scc has been recognised inside and outside the CP community (*motivating a few GPU-based approaches for scc*).
- Scc computation usually taught using Tarjan or Kosaraju algorithms (*both implicitly assume that every arc needs to be scanned*).
- Potentially relevant work gets unnoticed, (*the partial complementary representation of Dahlhaus* et al.).

Post-Mortem Analysis (continued)

- In CP, we derive many graphs from the domain store of a subset of variables + some extra linear size information (*e.g. for alldifferent domain store* + *maximum matching*)
  - **Copying/creating explicitly these graphs kills you,** and motivated bound-consistency algorithms.
- Remark from Laurière's HDR in 1976:
  - No use of ad hoc graph algorithms for alldifferent as intelligent systems should not rely on black-box algorithms, but should rather be able to reconstruct them from some kind of first principles.

Part 4: building maps of hidden links between combinatorial objects

One thing we made explicit:

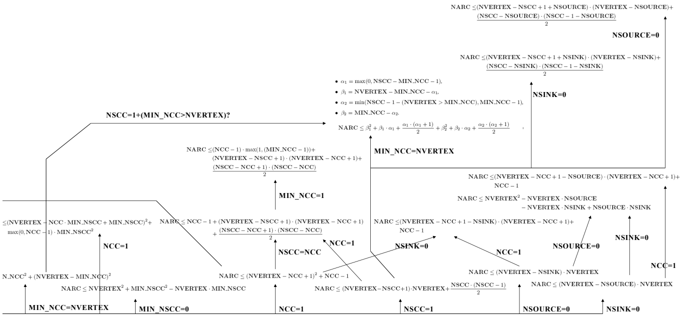
- It is very difficult to define what is a simple formula.

Genesis of the idea of a map

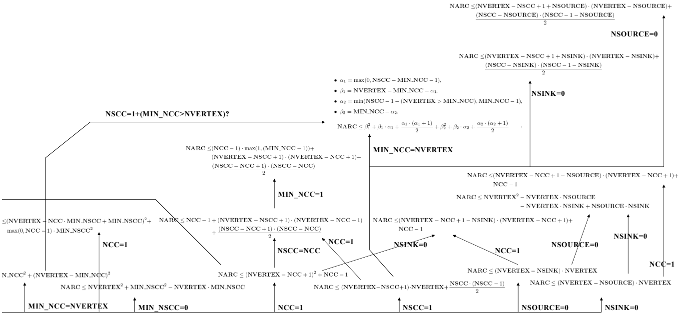
- Observations in 1999:
  - Many constraints of CHIP had a **lot** of arguments (up to 16).
  - These arguments **do not vary independently**.
  - It is impossible to **catch all their interactions** within a single filtering algorithm.

So I planned capturing these interactions in a systematic way with so-called map.

First map of sharp bounds at SICS in 2000



First map of sharp bounds at SICS in 2000



At that time, the notion of learning bounds from data was not there, so in 2019, I started a project to learn such maps (*with Claude-Guy, Jovial, Ramiz, Rémi*)

Some context

- **Combinatorial object**: mathematical structure with a finite number of elements (*permutation, partition*).
- **Characteristics**: metric characterising an instance of the combinatorial object (*number of cycles of a permutation*).

Some context

- **Conjecture**: assumed formula, linking characteristics.
- **Invariant**: conjecture having been proven
  sharp bound on number of arcs in a digraph:
  $m \leq n^2$.
- **Formula bias**: the space of formula considered by invariants.

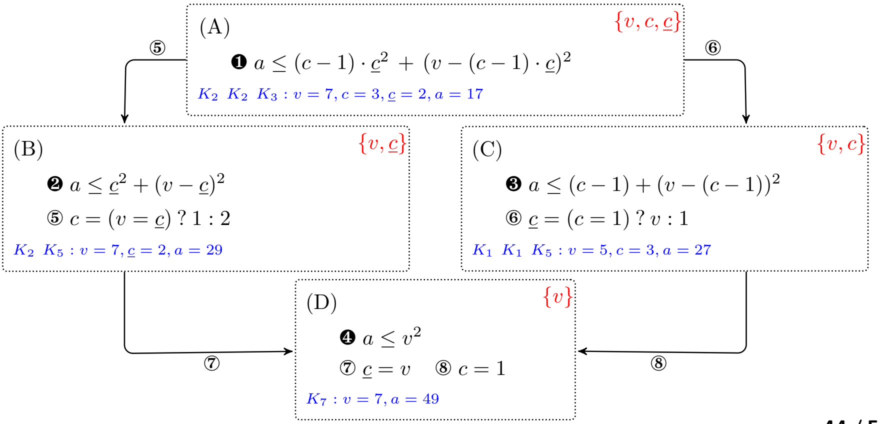
What sets conjecture acquisition apart from typical machine learning tasks

- Need to fit perfectly all positive examples.
- The formula bias is huge.
- Need to select the most significant conjectures from many and prove them.

Map example of interrelated sharp bounds

Upper bounds of number of arcs a of a digraph wrt:

- number of vertices v,
- number of connected components c,
- size of the smallest connected component $\underline{c}$.



Bound for the BACP: sharp lower bound on sum of the squares of the parts of a partition

- n is the number of elements of the partition,
- P is its number of parts,
- $\underline{M}$ is its size of the smallest part,
- $\overline{M}$ is its size of the biggest part.

Bound for the BACP: sharp lower bound on sum of the squares of the parts of a partition

- n is the number of elements of the partition,
- P is its number of parts,
- $\underline{M}$ is its size of the smallest part,
- $\overline{M}$ is its size of the biggest part.

$$S \geq 2 \cdot a \cdot nn + ss + nn - a^2 \cdot vv - a \cdot vv$$

$$\begin{aligned}
vv &= (P = 1\ ?\ 0 : P - 2) \\
nn &= \max(n - \overline{M} - \underline{M}, 0) \\
a &= \begin{cases} \left\lfloor \frac{nn}{vv} \right\rfloor & \text{if } P > 2 \\ 0 & \text{else} \end{cases} \\
ss &= \begin{cases} \underline{M}^2 + \overline{M}^2 & \text{if } P \geq 2 \\ \underline{M}^2 & \text{else} \end{cases}
\end{aligned}$$

Sharp lower bound on sum of the squares (intuition when $P \geq 2$ from Jovial)

Minimising the sum of squares means **balancing the load** as effectively as possible, while considering the **feasibility constraints**.

Sharp lower bound on sum of the squares (intuition when $P \geq 2$ from Jovial)

$$S \geq \overline{M}^2 \cdot 1 + \underline{M}^2 \cdot 1 + (a+1)^2 \cdot (nn \bmod vv) + a^2 \cdot (vv - (nn \bmod vv))$$

$$S_{min} = \sum_i Size_i{}^2 \times \textit{number of parts of size } Size_i$$

with $Size_i \in \{\overline{M}, \underline{M}, a+1, a\}$

$$1 \quad + \quad 1 \quad + \quad nn \bmod vv \quad + \quad vv - (nn \bmod vv) \quad = \quad P \text{ parts}$$

- P : number of parts,
- nn : number of elements remaining without the largest and smallest parts,
- vv : number of remaining parts without the largest and smallest parts,
- a : average size of parts excluding the largest and smallest parts.

Application to BACP [CPAIOR 2025]

(minimising the sum of squares of students' load)

	SICStus		Chuffed	
	Number of instances solved	**Average time to prove optimality**	**Number of instances solved**	**Average time to prove optimality**
Model 1 (Boolean)	29	1 min	30	0.2 min
Model 2 (Partition)	11	2.5 min	31	0.2 min
Model 3 (Cumulative)	29	1.2 min	27	1.6 min

No model proved optimality **without** the sharp bounds.

Bound Seeker on 8 combinatorial objects

Combinatorial object	# of maps	# of conjectures
Digraphs	16	2413
Rooted trees	10	189
Rooted forests with isolated vertex	20	1862
Rooted forests without isolated vertex	20	1779
Non empty partitions	10	779
Partitions with empty set	10	343
Sequences of 0/1	20	4603
Cyclic Sequences of 0/1	20	4162
Total	126	16130

Part 5: MDD and global constraints

Two observations and one question:

- The appeal of representing all solutions as memory capacity and core count grow.
- The wide gap between a general method and dedicated filtering algorithms.

Are they really irreconcilable?

MDD in CP

- Probably introduced in CP by me and Mats, see **case** constraint, [SICStus Release 3.9.0, 2002].
- Motivated:
  - originally by **configuration problems** (*putting together several element constraints*),
  - later by constraints on sequences, and
  - recently for encoding large corpus (*generating text with LLM and CP*).
- Later on, a lot of work on MDD in CP: e.g., Yap, Van Hoeve, Cire, Régin, Michel.
- Relaxation of global constraints with limited-width MDDs.

The MDD paradox

- On the one hand, MDD are general.
- On the other hand, global constraints (cumulatives, diffn, lex_chain, stable_keysort) have dedicated filtering algorithms.
- But many (configuration) problems would benefit from a tight integration of both worlds.

Question: how to do such tight integration?

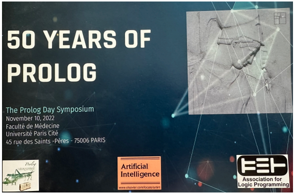
Conclusion

- After industry, I could start my academic career at SICS in Uppsala with Mats, where
  - I found support.
  - A non-competitive atmosphere (*compete just with yourself*).
  - Lagom (*got an annual review in a casual park setting during a company outing*).
  - Publications are the result of a work, not a goal per se.

There are so many potential research topics that all you have to do is look around and scratch the surface a little.

Epilogue

- Prolog Day Symposium, 10 November 2022, in Paris.
- Price won by people using SICStus, (*train traffic control, Siemens*).
- Video about Colmerauer returning to Europe and creating Prolog;



Epilogue

- Prolog Day Symposium, 10 November 2022, in Paris.
- Price won by people using SICStus, (*train traffic control, Siemens*).
- Video about Colmerauer returning to Europe and creating Prolog; in fact Colmerauer told me once:

"*Mats was a person of few words but understood Prolog more than him.*"