Complexity of enumeration: saturation problems

Arnaud Mary[1] **Yann Strozecki**[2]

[1]Baobab, Lyon

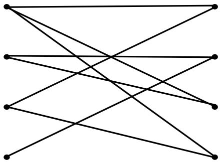
[2]Laboratoire D AVID, Versailles

Orléans, STACS 2016

Enumeration problems

- Enumeration problems: list all solutions rather than just deciding whether there is one.

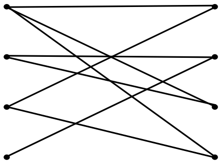
Perfect matching ?



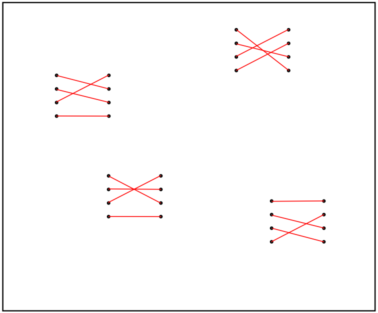
Enumeration problems

- Enumeration problems: list all solutions rather than just deciding whether there is one.

Perfect matching ?



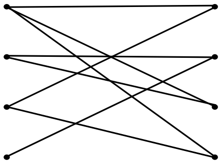
Solution space:



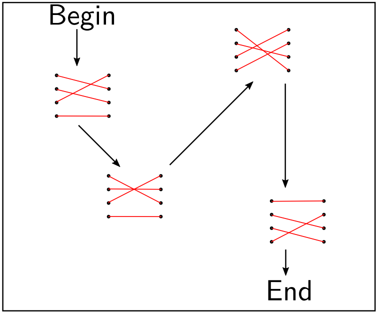
Enumeration problems

- Enumeration problems: list all solutions rather than just deciding whether there is one.

Perfect matching ?



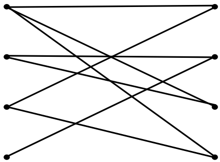
Solution space:



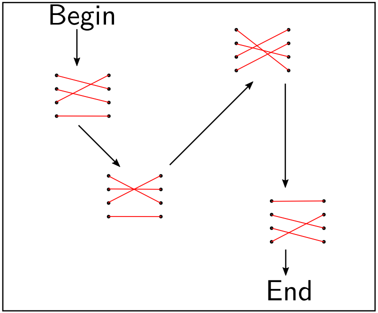
Enumeration problems

- Enumeration problems: list all solutions rather than just deciding whether there is one.
- Complexity measures: total time and delay between solutions.

Perfect matching ?



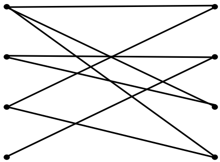
Solution space:



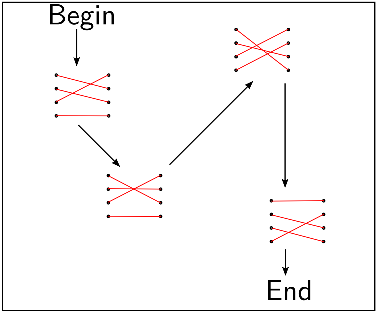
Enumeration problems

- Enumeration problems: list all solutions rather than just deciding whether there is one.
- Complexity measures: total time and delay between solutions.
- Motivations: database queries, optimization, building libraries.

Perfect matching ?



Solution space:



Framework

An enumeration problem A is a function which associates to each input a set of solutions $A(x)$.

An enumeration algorithm must generate every element of $A(x)$ one after the other without repetition.

Framework

An enumeration problem A is a function which associates to each input a set of solutions $A(x)$.

An enumeration algorithm must generate every element of $A(x)$ one after the other without repetition.

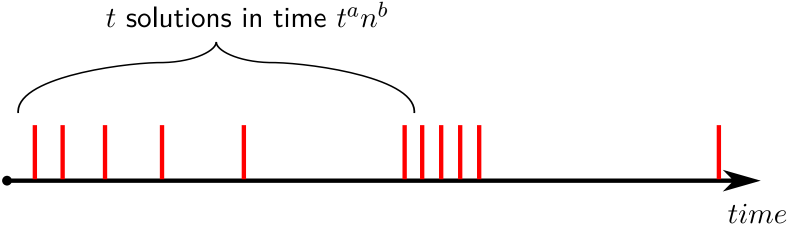
Concrete complexity classes:
A polynomial time precomputation is allowed.

1. Polynomial total time: $\textsc{TotalP}$
2. Incremental polynomial time: $\textsc{IncP}$
3. Polynomial delay: $\textsc{DelayP}$

Incremental time

Definition (Incremental polynomial time)

IncP is the set of enumeration problems such that there is an algorithm which for all t produces t solutions (if they exist) from an input of size n in time $O(t^a n^b)$ with a, b constants.



Saturation algorithm

Most algorithms with an incremental delay are saturation algorithms:

- **begin** with a polynomial number of simple solutions
- **for each** k-uple of already generated solutions apply a rule to produce a new solution
- **stop** when no new solutions are found

Saturation algorithm

Most algorithms with an incremental delay are saturation algorithms:

- **begin** with a polynomial number of simple solutions
- **for each** k-uple of already generated solutions apply a rule to produce a new solution
- **stop** when no new solutions are found

1. Accessible vertices in a graph by flooding.
2. Generate a finite group from a set of generators.
3. Generate all possible unions of sets:

Saturation algorithm

Most algorithms with an incremental delay are saturation algorithms:

- **begin** with a polynomial number of simple solutions
- **for each** k-uple of already generated solutions apply a rule to produce a new solution
- **stop** when no new solutions are found

1. Accessible vertices in a graph by flooding.
2. Generate a finite group from a set of generators.
3. Generate all possible unions of sets:
   - $\{12, 134, 23, 14\}$
   - $\{12, 134, 1234, 23, 14\}$
   - $\{12, 134, 1234, 23, 123, 14\}$
   - $\{12, 134, 1234, 23, 123, 14, 124\}$

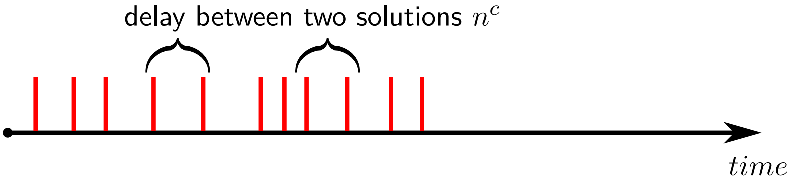
Polynomial Delay

The delay is the maximum time between the production of two consecutive solutions in an enumeration.

Definition (Polynomial delay)

$\textsc{DelayP}$ is the set of enumeration problems such that there is an algorithm whose delay is polynomial in the input.

$$\textsc{DelayP} \subseteq \textsc{IncP}$$



Unions in polynomial delay

Closure by union revisited.
Instance: a set $S = \{s_1, \dots s_m\}$ with $s_i \subseteq \{1, \dots, n\}$.
Problem: generate all unions of elements in S.

Unions in polynomial delay

Closure by union revisited.
Instance: a set $S = \{s_1, \dots s_m\}$ with $s_i \subseteq \{1, \dots, n\}$.
Problem: generate all unions of elements in S.

1. Recursive strategy, enumerate first the solutions which contains 1, then those which do not contain 1.

Unions in polynomial delay

Closure by union revisited.
Instance: a set $S = \{s_1, \dots s_m\}$ with $s_i \subseteq \{1, \dots, n\}$.
Problem: generate all unions of elements in S.

1. Recursive strategy, enumerate first the solutions which contains 1, then those which do not contain 1.
2. The algorithm should not explore a branch without solutions (flashlight search), so that we can bound the delay.

Unions in polynomial delay

Closure by union revisited.
Instance: a set $S = \{s_1, \dots s_m\}$ with $s_i \subseteq \{1, \dots, n\}$.
Problem: generate all unions of elements in S.

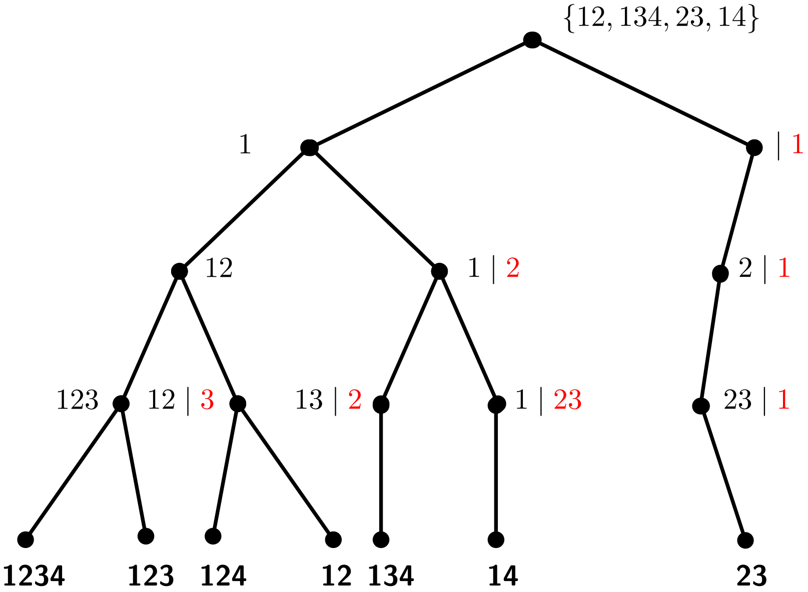
1. Recursive strategy, enumerate first the solutions which contains 1, then those which do not contain 1.
2. The algorithm should not explore a branch without solutions (flashlight search), so that we can bound the delay.
3. We must solve the extension problem: given two sets A and B is there a **solution** S **such that** $A \subseteq S$ **and** $S \cap B = \emptyset$?

Unions in polynomial delay

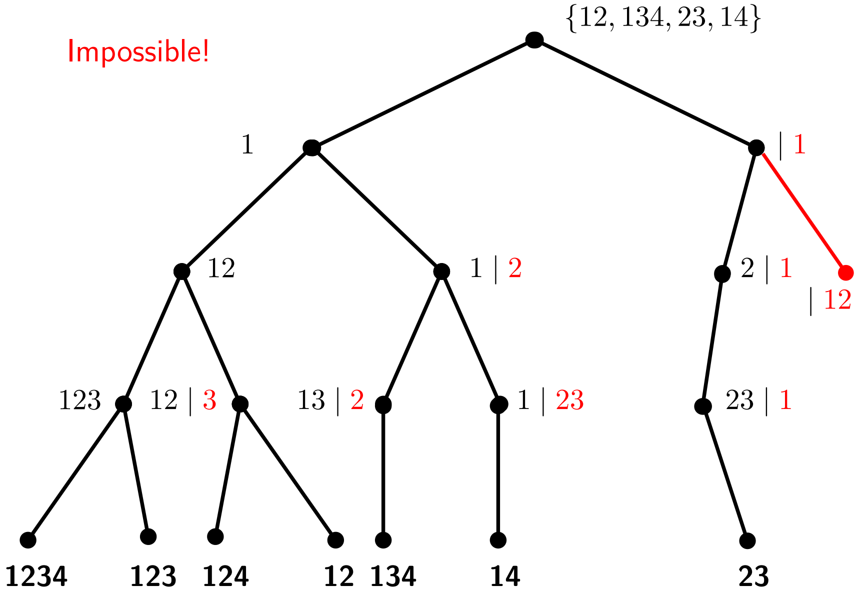
Closure by union revisited.
Instance: a set $S = \{s_1, \dots s_m\}$ with $s_i \subseteq \{1, \dots, n\}$.
Problem: generate all unions of elements in S.

1. Recursive strategy, enumerate first the solutions which contains 1, then those which do not contain 1.
2. The algorithm should not explore a branch without solutions (flashlight search), so that we can bound the delay.
3. We must solve the extension problem: given two sets A and B is there a **solution** S **such that** $A \subseteq S$ **and** $S \cap B = \emptyset$?
4. The extension problem is easy to solve in time $O(mn)$ thus the backtrack search has delay $O(mn^2)$.

Partial solution tree



Partial solution tree



From saturation to polynomial delay

Question

Can we solve saturation problems with a polynomial delay ?

From saturation to polynomial delay

Question

Can we solve saturation problems with a polynomial delay ?

No, since saturation problems are "equals" to $\textsc{IncP}$ and we have recently proved $\textsc{IncP} \neq \textsc{DelayP}$.

From saturation to polynomial delay

Question

Can we solve saturation problems with a polynomial delay ?

No, since saturation problems are “equals” to $\textsc{IncP}$ and we have recently proved $\textsc{IncP} \neq \textsc{DelayP}$.

We need to restrict the saturation rules. Since it works for the union, we will consider set operations.

Our goal is to design the largest possible toolbox of efficient enumeration algorithms.

Set operations

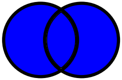
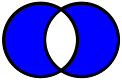
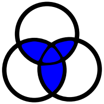
A set over $\{1, \ldots, n\}$ will be represented by its characteristic vector of size n.

A set operation is a boolean operation $\{0,1\}^k \to \{0,1\}$ applied componentwise to k boolean vectors.

Set operations

A set over $\{1,\ldots,n\}$ will be represented by its characteristic vector of size n.

A set operation is a boolean operation $\{0,1\}^k \to \{0,1\}$ applied componentwise to k boolean vectors.

$\vee$	$\begin{pmatrix}1\\0\\1\end{pmatrix} \vee \begin{pmatrix}1\\1\\0\end{pmatrix} = \begin{pmatrix}1\\1\\1\end{pmatrix}$	$\cup$	
$+$	$\begin{pmatrix}1\\0\\1\end{pmatrix} + \begin{pmatrix}1\\1\\0\end{pmatrix} = \begin{pmatrix}0\\1\\1\end{pmatrix}$	$\triangle$	
$\mathrm{maj}(x,y,z)$	$\mathrm{maj}(\begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}1\\1\\0\end{pmatrix}) = \begin{pmatrix}1\\0\\0\end{pmatrix}$	Majority	

Closure by set operation

Let $\mathcal{S}$ be a set of boolean vectors of size n and $\mathcal{F}$ be a finite set of boolean operations.

Closure:

- $\mathcal{F}^0(\mathcal{S}) = \mathcal{S}$
- $\mathcal{F}^i(\mathcal{S}) = \{f(v_1, \dots, v_t) \mid v_1, \dots, v_t \in \mathcal{F}^{i-1}(S)$ and $f \in \mathcal{F}\}$
- $Cl_{\mathcal{F}}(\mathcal{S}) = \cup_i \mathcal{F}^i(\mathcal{S})$

Closure by set operation

Let $\mathcal{S}$ be a set of boolean vectors of size n and $\mathcal{F}$ be a finite set of boolean operations.

Closure:

- $\mathcal{F}^0(\mathcal{S}) = \mathcal{S}$
- $\mathcal{F}^i(\mathcal{S}) = \{f(v_1, \ldots, v_t) \mid v_1, \ldots, v_t \in \mathcal{F}^{i-1}(S)$ and $f \in \mathcal{F}\}$
- $Cl_{\mathcal{F}}(\mathcal{S}) = \cup_i \mathcal{F}^i(\mathcal{S})$

Our enumeration problem is then to list the elements of $\mathrm{C}l_{\mathcal{F}}(\mathcal{S})$.

Extension problem

$\textsc{Closure}_{\mathcal{F}}$:

Input: $\mathcal{S}$ a set of vectors of size n, and a vector v of size n
Problem: decide whether $v \in Cl_{\mathcal{F}}(\mathcal{S})$.

$\textsc{Closure}_{\mathcal{F}}$ is the extension problem associated to the computation of $Cl_{\mathcal{F}}(\mathcal{S})$.

Extension problem

$\textsc{Closure}_{\mathcal{F}}$:

Input: $\mathcal{S}$ a set of vectors of size n, and a vector v of size n
Problem: decide whether $v \in Cl_{\mathcal{F}}(\mathcal{S})$.

$\textsc{Closure}_{\mathcal{F}}$ is the extension problem associated to the computation of $Cl_{\mathcal{F}}(\mathcal{S})$.

Goal: prove that **Closure**$_{\mathcal{F}} \in \mathsf{P}$ for as many sets $\mathcal{F}$ as possible, to use the backtrack search.

Clones and reduction

There are many finite families of boolean operations, how to reduce their number ?

Clones and reduction

There are many finite families of boolean operations, how to reduce their number ?

Definition

Let $\mathcal{F}$ be a finite set of operations, the functional clone generated by $\mathcal{F}$, denoted by $< \mathcal{F} >$, is the set of operations obtained by any composition of the operations of $\mathcal{F}$ and of the projections π_k^n defined by $\pi_k^n(x_1, \dots, x_n) = x_k$.

For instance $(x \vee y) + x + z \ \in < \vee, + >$.

Clones and reduction

There are many finite families of boolean operations, how to reduce their number ?

Definition

Let $\mathcal{F}$ be a finite set of operations, the functional clone generated by $\mathcal{F}$, denoted by $< \mathcal{F} >$, is the set of operations obtained by any composition of the operations of $\mathcal{F}$ and of the projections π_k^n defined by $\pi_k^n(x_1, \ldots, x_n) = x_k$.

For instance $(x \vee y) + x + z \in < \vee, + >$.

Lemma

For all set of operations $\mathcal{F}$ and all set of vectors $\mathcal{S}$,
$Cl_{\mathcal{F}}(\mathcal{S}) = Cl_{<\mathcal{F}>}(\mathcal{S})$.

Clones and reduction

There are many finite families of boolean operations, how to reduce their number ?

Definition

Let $\mathcal{F}$ be a finite set of operations, the functional clone generated by $\mathcal{F}$, denoted by $< \mathcal{F} >$, is the set of operations obtained by any composition of the operations of $\mathcal{F}$ and of the projections π_k^n defined by $\pi_k^n(x_1, \ldots, x_n) = x_k$.

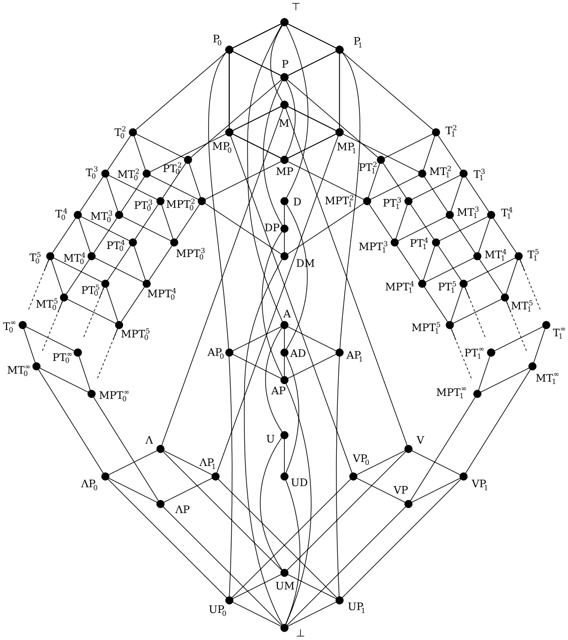
For instance $(x \vee y) + x + z \in < \vee, + >$.

Lemma

For all set of operations $\mathcal{F}$ and all set of vectors $\mathcal{S}$,
$Cl_{\mathcal{F}}(\mathcal{S}) = Cl_{<\mathcal{F}>}(\mathcal{S})$.

There are less clones than families and they are well described and organized in Post's lattice.

Post's lattice



How to reduce Post's lattice

To an operation f we can associate its dual $\overline{f}$ defined by $\overline{f}(s_1, \ldots, s_t) = \neg f(\neg s_1, \ldots, \neg s_t)$.

Proposition

The following problems can be polynomially reduced to $\textsc{Closure}_{\mathcal{F}}$*:*

1. $\textsc{Closure}_{\overline{\mathcal{F}}}$
2. $\textsc{Closure}_{\mathcal{F} \cup \{\neg\}}$ *when* $\mathcal{F} = \overline{\mathcal{F}}$
3. $\textsc{Closure}_{\mathcal{F} \cup \{\mathbf{0}\}}$, $\textsc{Closure}_{\mathcal{F} \cup \{\mathbf{1}\}}$, $\textsc{Closure}_{\mathcal{F} \cup \{\mathbf{0},\mathbf{1}\}}$

Reduced Post's lattice

Clone	Base
I_2	$\emptyset$
L_2	$x + y + z$
L_0	$+$
E_2	$\wedge$
S_{10}	$x \wedge (y \vee z)$
S_{10}^k	$Th_k^{k+1}, x \wedge (y \vee z)$
S_{12}	$x \wedge (y \rightarrow z)$
S_{12}^k	$Th_k^{k+1}, x \wedge (y \rightarrow z)$
D_2	maj
D_1	$\text{maj}, x + y + z$
M_2	$\vee, \wedge$
R_2	$x \mathbin{?} y : z$
R_0	$\vee, +$

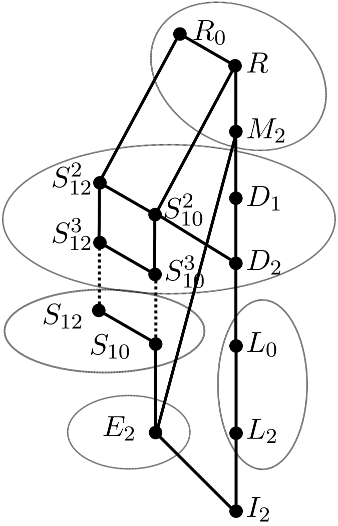


Figure: Reduced Post's lattice, the edges represent inclusions of clones

Union revisited bis

The case of $< \vee >$ is done and is equivalent to $E_2 =< \wedge >$. The delay is $O(mn^2)$, can we improve it?

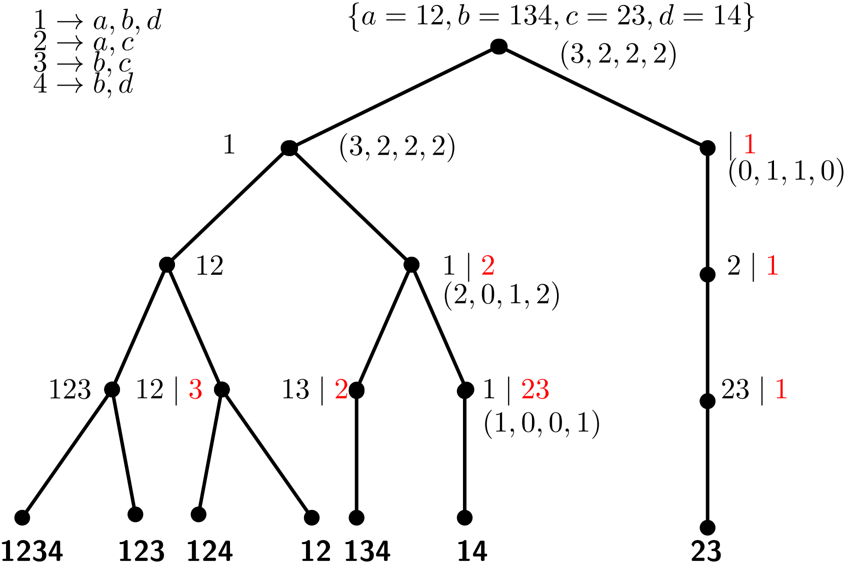
Union revisited bis

The case of $< \vee >$ is done and is equivalent to $E_2 =< \wedge >$. The delay is $O(mn^2)$, can we improve it?

- Complexity comes from solving repeatedly the extension problem.
- We can set up datastructures to solve it faster.
- During a branch of the backtrack search we go over the instance once.
- Therefore the delay is improved to $O(mn)$.

Open question: can we get rid or decrease the dependency on m?

The data structures



Algebras

- $L_0 =< x + y >$, $Cl_{L_0}(\mathcal{S})$ is the vector space generated by the vectors in $\mathcal{S}$.
- $\textsc{Closure}_{L_0} \in \mathsf{P}$ since it is equivalent to solving a linear system.

Algebras

- $L_0 =< x + y >$, $Cl_{L_0}(\mathcal{S})$ is the vector space generated by the vectors in $\mathcal{S}$.
- $\textsc{Closure}_{L_0} \in$ P since it is equivalent to solving a linear system.

- $A =< \vee, \neg >$ is the set of all boolean functions. $Cl_A(\mathcal{S})$ is the boolean algebra generated by the atoms.
- $\textsc{Closure}_A \in$ P by computing the union of the atoms corresponding to required elements.

Algebras

- $L_0 =< x + y >$, $Cl_{L_0}(\mathcal{S})$ is the vector space generated by the vectors in $\mathcal{S}$.
- $\textsc{Closure}_{L_0} \in \mathsf{P}$ since it is equivalent to solving a linear system.

- $A =< \vee, \neg >$ is the set of all boolean functions. $Cl_A(\mathcal{S})$ is the boolean algebra generated by the atoms.
- $\textsc{Closure}_A \in \mathsf{P}$ by computing the union of the atoms corresponding to required elements.

- In both cases, the base can be turned into explicit solutions by Gray code enumeration with a delay $O(n)$.

Majority

Proposition

Let $\mathcal{S}$ be a vector set, a vector v belongs to $Cl_{<maj>}(\mathcal{S})$ if and only if for all $i, j \in [n]$, $i \neq j$, there exists $x \in \mathcal{S}$ such that $x_{i,j} = v_{i,j}$.

Idea of the proof: you build incrementally the vector v by using a sequence of vectors which have the same pairs as v.

Majority

Proposition

Let $\mathcal{S}$ be a vector set, a vector v belongs to $Cl_{<maj>}(\mathcal{S})$ if and only if for all $i, j \in [n]$, $i \neq j$, there exists $x \in \mathcal{S}$ such that $x_{i,j} = v_{i,j}$.

Idea of the proof: you build incrementally the vector v by using a sequence of vectors which have the same pairs as v.

- The possible values of the pairs can be computed and stored in the precomputation step.
- At each step of the backtrack search, we fix one element therefore we need to check a linear number of pairs.
- $\textsc{Closure}_{maj} \in \mathrm{P}$ and the delay is $O(n^2)$.

Thank universal algebra

An operation f is a near unanimity of arity k if it satisfies $f(x_1, x_2, \ldots, x_k) = x$ for each k-tuple with at most one element different from x.

Thank universal algebra

An operation f is a near unanimity of arity k if it satisfies $f(x_1, x_2, \ldots, x_k) = x$ for each k-tuple with at most one element different from x.

Theorem (Baker-Pixley)

Let $\mathcal{F}$ be a clone which contains a near unanimity term of arity k, then $v \in Cl_{\mathcal{F}}(\mathcal{S})$ if and only if for all set of indices I of size $k-1$, $v_I \in Cl_{\mathcal{F}}(\mathcal{S}_I)$.

Thank universal algebra

An operation f is a near unanimity of arity k if it satisfies $f(x_1, x_2, \ldots, x_k) = x$ for each k-tuple with at most one element different from x.

Theorem (Baker-Pixley)

Let $\mathcal{F}$ be a clone which contains a near unanimity term of arity k, then $v \in Cl_{\mathcal{F}}(\mathcal{S})$ if and only if for all set of indices I of size $k-1$, $v_I \in Cl_{\mathcal{F}}(\mathcal{S}_I)$.

Corollary

For all clones $\mathcal{F}$ containing a near unanimity, $\textsc{Closure}_{\mathcal{F}} \in \mathsf{P}$.

The result

Theorem

For all sets $\mathcal{F}$ of boolean operations, $\textsc{Closure}_{\mathcal{F}} \in \mathsf{P}$.

Corollary

For all sets $\mathcal{F}$ of boolean operations, enumerating $Cl_{\mathcal{F}}$ is in $\textsc{DelayP}$.

The result

Theorem

For all sets $\mathcal{F}$ of boolean operations, $\textsc{Closure}_{\mathcal{F}} \in \mathsf{P}$.

Corollary

For all sets $\mathcal{F}$ of boolean operations, enumerating $Cl_{\mathcal{F}}$ is in $\textsc{DelayP}$.

Can we generalize this result to vectors over a finite domain D with more than two elements?

Larger domains

What does not work:

- The lattice of clones is uncountable and not well described.

Larger domains

What does not work:

- The lattice of clones is uncountable and not well described.
- Over $D = \{0, 1, 2\}$, let $f(x, y) = x + y$ when $x + y <= 2$ otherwise $f(x, y) = 2$. $\textsc{Closure}_{<f>}$ is NP-hard.

Larger domains

What does not work:

- The lattice of clones is uncountable and not well described.
- Over $D = \{0, 1, 2\}$, let $f(x, y) = x + y$ when $x + y <= 2$ otherwise $f(x, y) = 2$. $\textsc{Closure}_{<f>}$ is NP-hard.

What does work:

- Near unanimity.

Larger domains

What does not work:

- The lattice of clones is uncountable and not well described.
- Over $D = \{0, 1, 2\}$, let $f(x, y) = x + y$ when $x + y <= 2$ otherwise $f(x, y) = 2$. $\textsc{Closure}_{<f>}$ is NP-hard.

What does work:

- Near unanimity.
- Group operations.

Larger domains

What does not work:

- The lattice of clones is uncountable and not well described.
- Over $D = \{0, 1, 2\}$, let $f(x, y) = x + y$ when $x + y <= 2$ otherwise $f(x, y) = 2$. $\textsc{Closure}_{<f>}$ is NP-hard.

What does work:

- Near unanimity.
- Group operations.
- Associative operations with an alternative algorithm and exponential space.

Take away

Results:

- For all sets $\mathcal{F}$ of boolean operations, $\textsc{Closure}_{\mathcal{F}} \in$ P and we have an efficient enumeration algorithm of $Cl_{\mathcal{F}}$.
- $\textsc{Closure}_{\mathcal{F}}$ can be NP-hard for three elements domain.

Take away

Results:

- For all sets $\mathcal{F}$ of boolean operations, $\textsc{Closure}_{\mathcal{F}} \in \mathsf{P}$ and we have an efficient enumeration algorithm of $Cl_{\mathcal{F}}$.
- $\textsc{Closure}_{\mathcal{F}}$ can be NP-hard for three elements domain.

Open questions:

- Characterize the complexity of $\textsc{Closure}_{\mathcal{F}}$ for larger domains (dichotomy theorem?).
- Find another enumeration strategy in polynomial delay and polynomial space.
- Improve the delay of enumerating $Cl_{<\vee>}$.

Thanks !

Questions ?