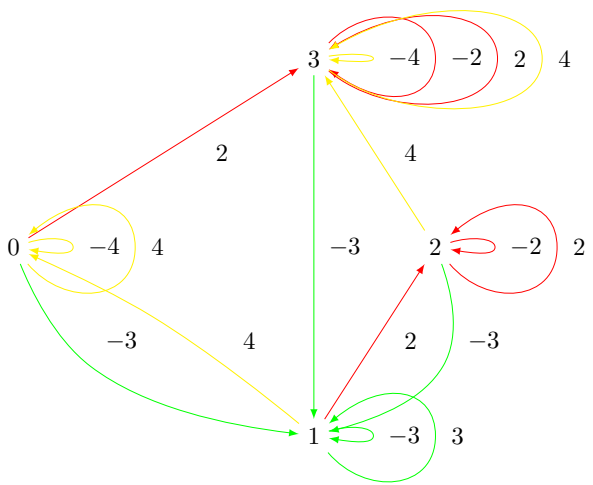
Algorithmes en théorie des représentations des monoïdes

Nicolas M. Thiéry

Laboratoire de Mathématiques d'Orsay, Université Paris Sud, France



Abstract

Algorithmes en théorie des représentations des monoïdes finis
La théorie des représentations est un outil classique pour extraire des informations, en particulier combinatoires, d'une structure algébrique A. En combinatoire algébrique, A est souvent l'algèbre d'un semigroupe ou d'un monoïde M. Cette information peut-elle permettre de mieux comprendre, ou au moins calculer, la théorie des représentations, comme c'est le cas pour les groupes?
C'est un sujet en pleine effervescence et nous en présenterons quelques aspects algorithmiques faisant entrer en jeu graphes, ordres partiels, algèbre linéaire, monoïdes, groupes et caractères.

L'exposé s'appuiera sur des exemples concrets liés aux algorithmes de tri, et la

démarche exploratoire sera illustrée par quelques calculs typiques avec le logiciel Sage.

Representation Theory

Let A be an algebraic structure under study; say

- A finite **monoid** $(M, *)$
- A finite **group** $(G, *)$
- An finite dimensional **algebra** $(A, +, ., *)$

Key questions of representation theory

- How does A relate with other algebraic structures?
- How to represent A using well known objects? (transformations, permutations, matrices, ...)

In practice

Search for all **representations**

- $A \mapsto T_n$ (action on a finite set)
- $A \mapsto \mathfrak{S}_n$ (permutation action on a finite set)
- $A \mapsto M_n(\mathbb{K})$ (linear action on $\mathbb{K}^n$)

Combinatorial Representation Theory

A tool to extract combinatorics from algebras:

- Dimension of simple and indecomposable projective modules
  ($\mathfrak{S}[n], \mathrm{gl}_n$: Kostka numbers)
- Induction and restrictions multiplicities
  ($\mathfrak{S}[m] \times \mathfrak{S}[n] \to \mathfrak{S}[m+n]$: Littlewood-Richardson rules)
- Cartan invariant matrices and quivers
  ($H_n(0)$: counting permutation by descents and recoils)
- Decomposition map
  ($H_n(q \mapsto 0)$: counting tableaux by shape and descents)

Computer exploration?

Computational Representation Theory

Finite groups (characteristic zero)

- Simple modules $\Leftrightarrow$ conjugacy classes
- Complexity: $O(\log(|G|))$
- Main tool: strong generating sets, character theory

Finite dimensional algebras

- Complexity: $O(\dim A^3)$ (in practice: $\dim \leq 1000$)
- Main tools: linear algebra, minimal polynomial, ...

Finite dimensional commutative algebras

- Simple modules $\Leftrightarrow$ solutions of the polynomial system
- Tools: Gröbner basis, RUR, ...

Several recent examples are monoid algebras

- 0-Hecke algebras (Norton, Carter, Krob-Thibon, Duchamp-Hivert-Thibon, Fayers, Denton)
- Non-decreasing parking function (Denton-Hivert-Schilling-T)
- Solomon-Tits algebras (Schocker, Saliola)
- Left Regular Bands (Brown) ...

How to take advantage of this fact?

Goals of the talk

- Anatomy of monoids
  Well known to the semigroup community

- Representation theory of monoids
  In progress: Clifford, Munn, Ponizovskiĭ, Mc Alister, Putcha, Saliola, Steinberg, Margolis, Bergeron, Denton, Hivert, Schilling, Thiéry, ...

- Algorithm for the Cartan invariants matrix

Running example: Order preserving functions on the chain

Definition

$f : \{1, \ldots, n\} \longmapsto \{1, \ldots, n\}$ is **order preserving** if:

$$i \leq j \quad \Longrightarrow \quad f(i) \leq f(j)$$

Example

The order preserving functions on $\{1 < 2 < 3\}$:

$$\{111, 112, 113, 122, 123, 133, 222, 223, 233, 333\}$$

Remark

If f, g are order preserving, then so is fg .
Hence, the set $\mathcal{O}_n$ of such functions is a **monoid** *!*

This still works if $\leq$ is replaced by a partial order

Understanding the multiplication?

First approach: the multiplication table:

$*$	111	112	113	122	123	133	222	223	233	333
111	111	111	111	111	111	111	222	222	222	333
112	111	111	111	112	112	113	222	222	223	333
113	111	112	113	112	113	113	222	223	223	333
122	111	111	111	122	122	133	222	222	233	333
123	111	112	113	122	123	133	222	223	233	333
133	111	122	133	122	133	133	222	233	233	333
222	111	111	111	222	222	333	222	222	333	333
223	111	112	113	222	223	333	222	223	333	333
233	111	122	133	222	233	333	222	233	333	333
333	111	222	333	222	333	333	222	333	333	333

The Cayley graph of a monoid

Remark

Thanks to associativity, it is sufficient to consider products

$$xg, \quad \text{for } x \in M \text{ and } g \text{ a generator}$$

Definition (Cayley graph)

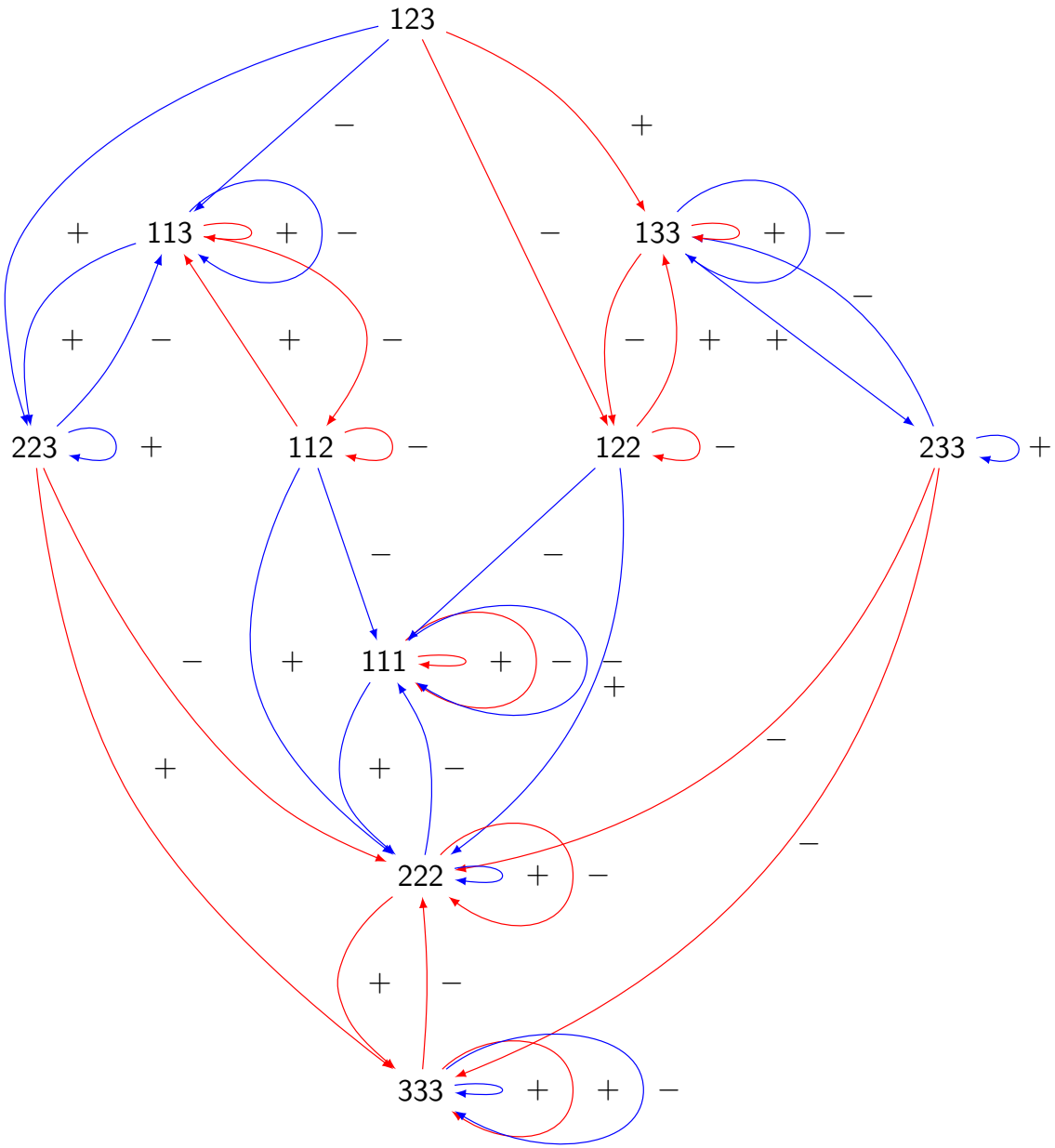
Graph with edges $x \overset{g}{\longmapsto} xg$

Example

Canonical generators for $\mathcal{O}_3$:

$$\pi_1^+ = 223, \qquad \pi_2^+ = 133$$
$$\pi_1^- = 113, \qquad \pi_2^- = 122$$

The right Cayley graph of $\mathcal{O}_3$



R-preorder (Green 50)

Definition ($\mathcal{R}$-preorder)

$$x \leq_R y \quad \text{if} \quad x \in yM$$

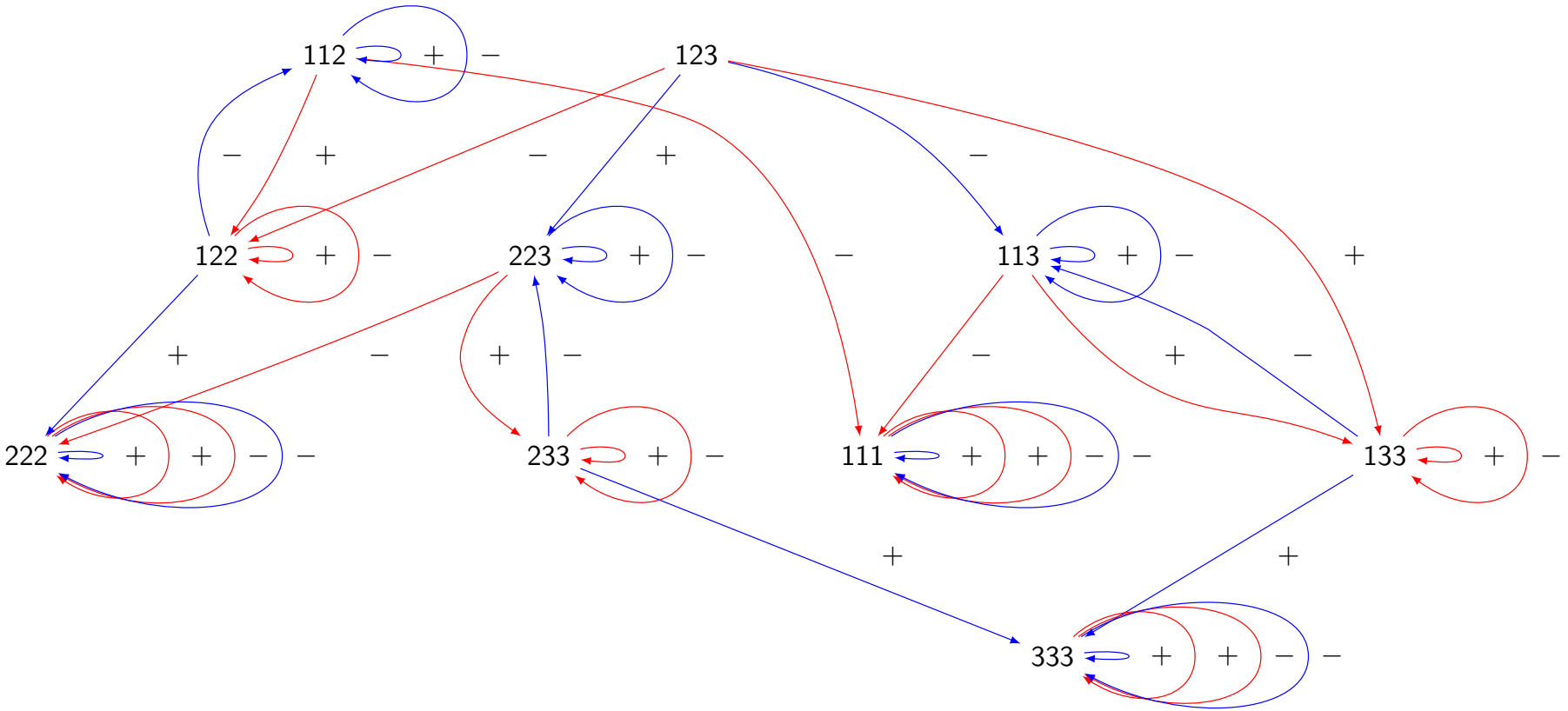
- $\mathcal{R}$-**class** $\mathcal{R}(x)$: strongly connected component
- $\mathcal{R}$-**order** on $\mathcal{R}$-classes
- $\mathcal{R}$-**trivial monoid**: all $\mathcal{R}$-classes are trivial

[

Algorithms]

- Basic: $O(|M||G|)$ (strongly connected components in a graph)
- Semigroupe: roughly $O(|M|)$ (ideas similar to F5)
- GAP+Monoid: $\leq O(|M|)$ using group theory, when possible

The left Cayley graph of $\mathcal{O}_3$



$\mathcal{L}$-preorder, $\mathcal{L}$-classes, ...

Transformation module X of M

Definition (Transformation module)

Finite set X with an action of M on X
I.e. representation of M as monoid of transformations in X^X

Described by its Cayley graph (an automaton)

Example

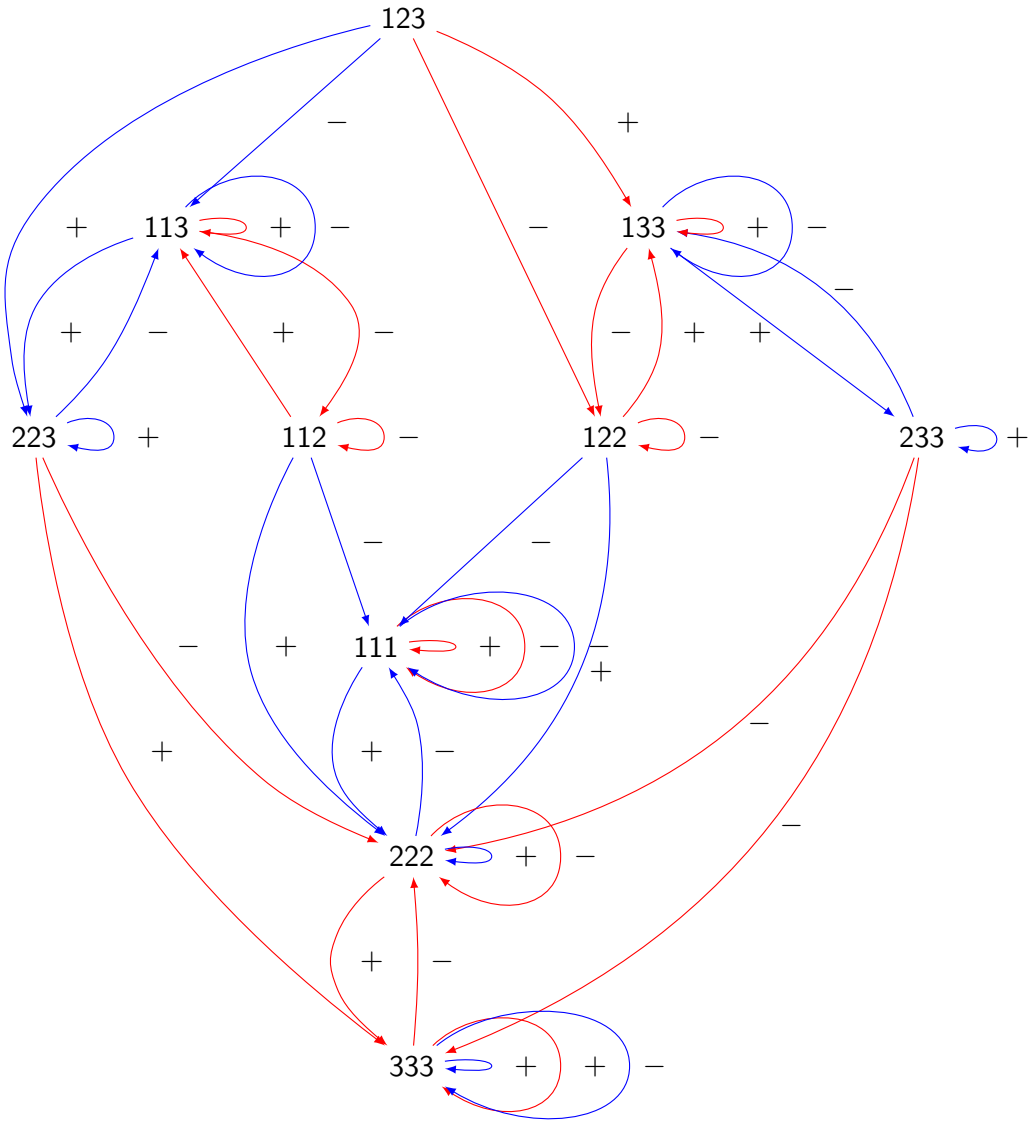
Regular representation of M acting on $X = M$ (associativity!)

Problem

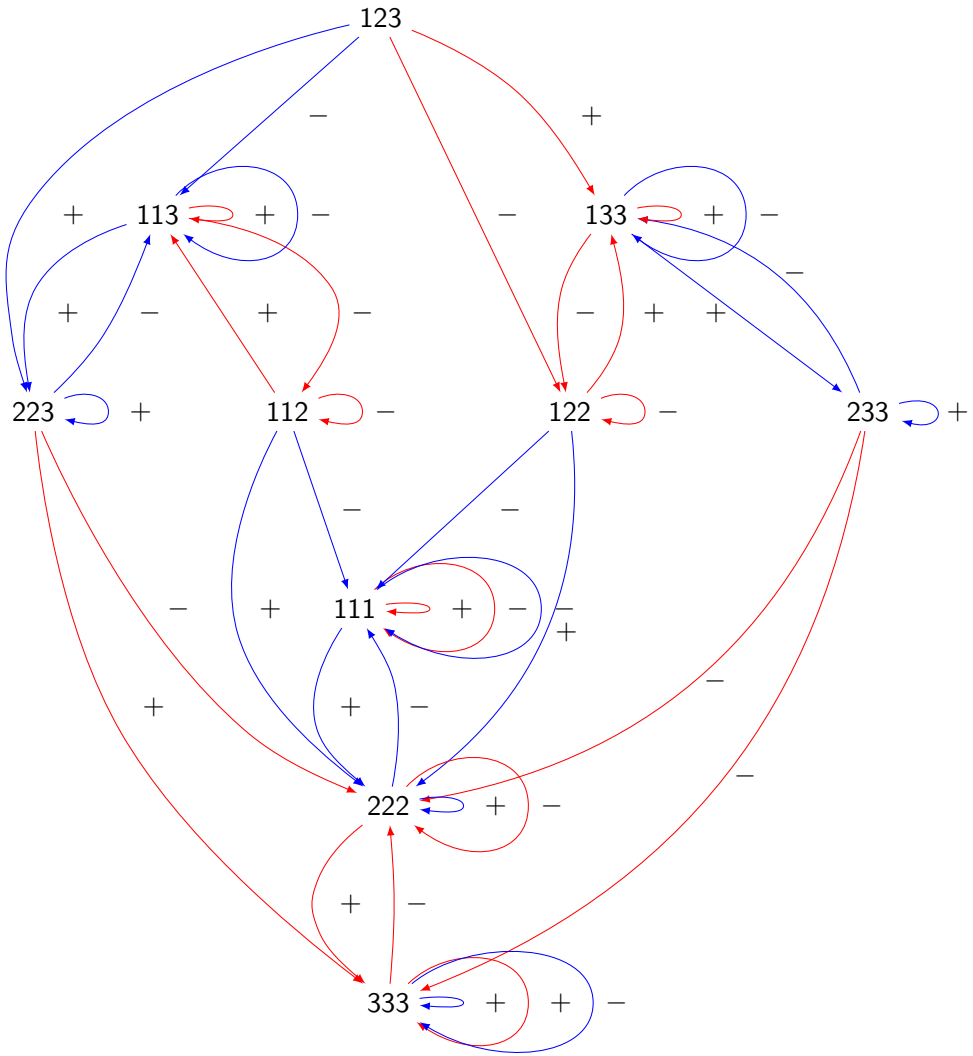
Describe all modules / representations of M?

Submodules

$X' \subset X$ is a **submodule** if it is stable under the action of M



Quotient by a submodule $X' \subset X$: $X \backslash X' \cup \{\emptyset\}$



$\mathcal{R}$-classes: smallest (transformation) subquotients

Left-right Cayley graph, $\mathcal{J}$-preorder

Problem

- *Why do we get several times the same module?*
- *Can we exploit associativity?*

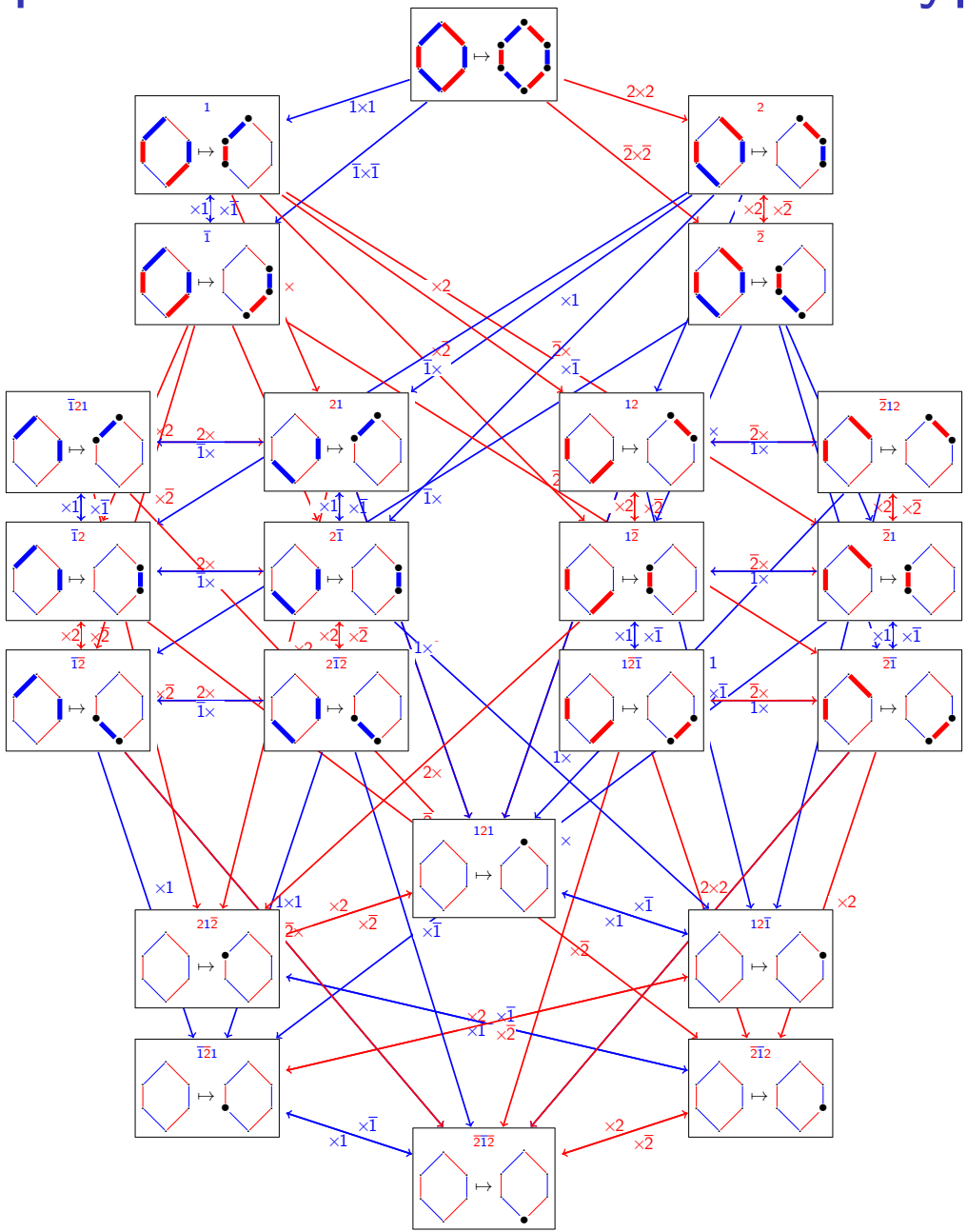
Definition ($\mathcal{J}$-preorder)

$$x \leq_J y \quad \text{if} \quad x \in MyM$$

- **Left-right Cayley graph**
- **$\mathcal{J}$-class**
- **$\mathcal{J}$-preorder**
- **$\mathcal{J}$-trivial**

Example: the left-right Cayley graph for $\mathcal{O}3$

Example: the biHecke monoid for type A_2



The eggbox picture

Proposition

Let J be a $\mathcal{J}$-class. Then,

$$J \quad \approx_{M-\mathsf{mod}-M} \quad L \times R$$

where L and R are respectively left and right classes

If e is an idempotent:

$$\mathcal{J}(e) = \mathcal{L}(e)\mathcal{R}(e)$$

Aperiodic Rees matrix monoid

Definition (Aperiodic Rees matrix monoid)

- Take $M := \{1, \ldots, n\} \times \{1, \ldots, m\}$
- Choose which (i, j) shall be idempotent
  Either none, or at least one per line and per column
- Set

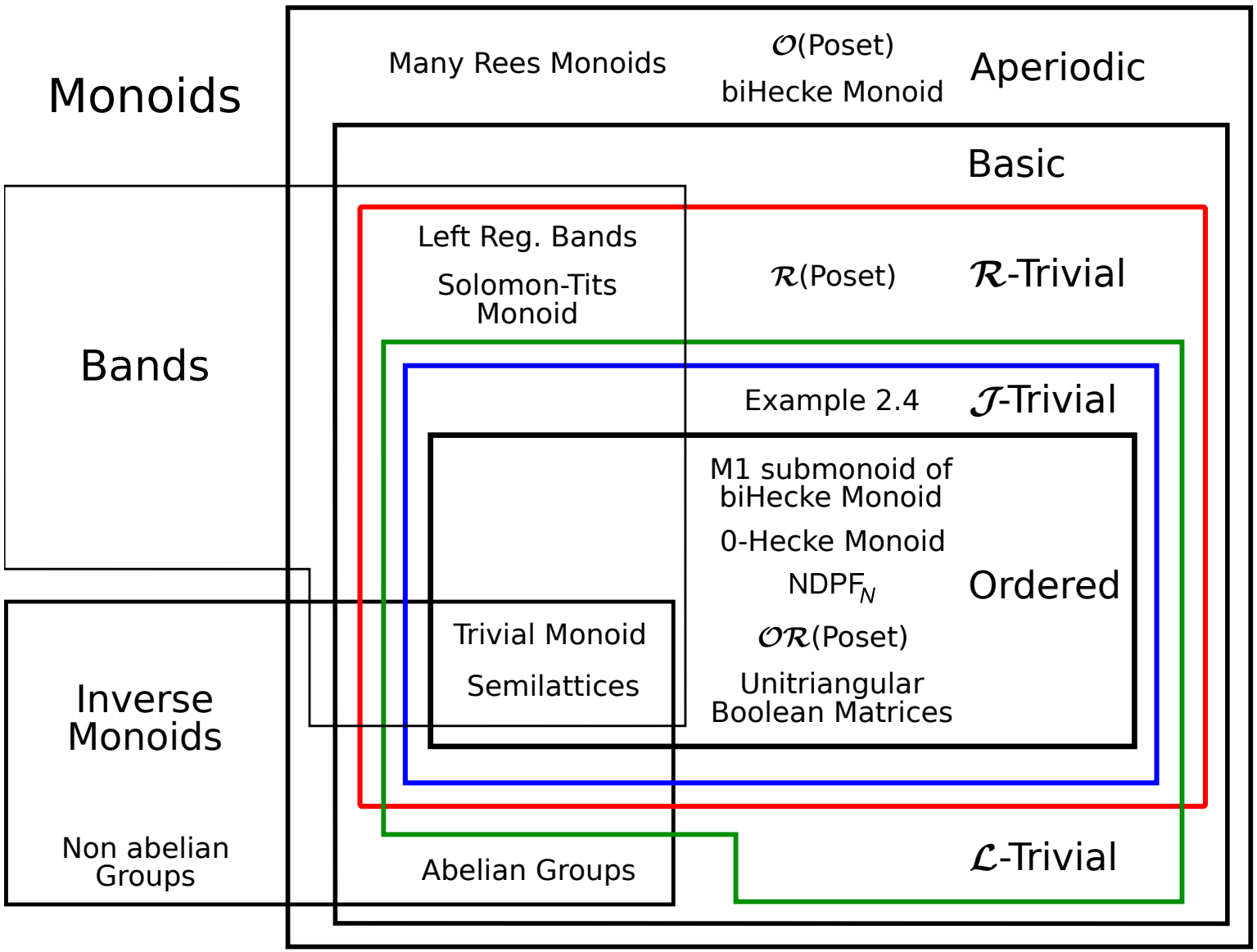
$$(i,j).(j,k) = \begin{cases} (i,k) & \text{if } (k,i) \text{ is idempotent} \\ 0 & \text{otherwise} \end{cases}$$

The structure of M is encoded by the **eggbox matrix** P.

Err, but groups are monoids, aren't they?

- A group has a single $\mathcal{J}$-class!
- So far we have been only speaking about **aperiodic monoids**
- **Rees matrix monoids** $R(P, G)$
- **Schützenberger representations**:
  - $R(e)$ is a G-mod-M module
  - $R(e)$ is a free G-mod (think coset decomposition)
  - Symmetrically for $L(e)$

Zoology of monoids



Simple modules, Clifford, Munn, Ponizovskiĭ

Let M be an aperiodic monoid

Proposition

For R a regular $\mathcal{R}$-class of M,

$$\mathrm{rad}\mathbb{K}R = \{x \in \mathbb{K}R,\ x.r = 0 \forall r \in R\}$$

Equivalently: $\mathrm{rad}\mathbb{K}R$ *is the kernel of the eggbox matrix*

Proposition

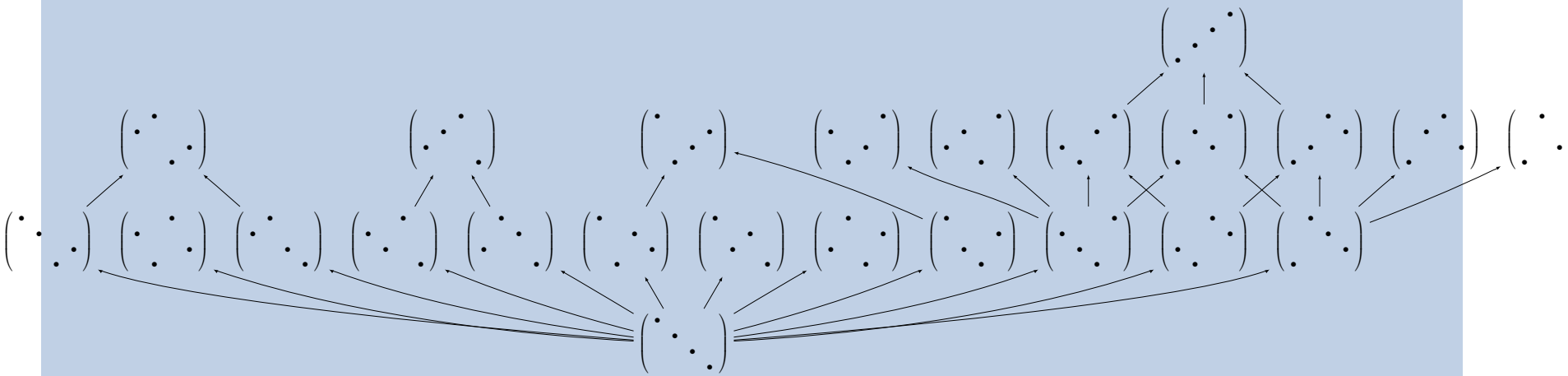
Define $S_i := \mathrm{top}R_i = \mathbb{K}R_i/\mathrm{rad}\mathbb{K}R_i$
All simple modules of M*:* $(S_i)_{i\in I}$

Simple modules for the biHecke Monoid

First non trivial aperiodic monoid admitting a combinatorial description of the (dimension) of the simple modules [HST09]

Combinatorial ingredients

- Intervals in right order
  (for right classes)
- Möbius inversion along the cutting poset
  (for moding out the radical)



Simple modules, Clifford, Munn, Ponizovskiĭ

Proposition

- *Take a J-class and G_i its group*
- *Take a simple module $S_{i,j}^{G_i}$ of G_i.*
- *Define $S_{i,j} := \mathrm{top}(S_{i,j}^{G_i} \uparrow_{G_i}^{M})$*

This construction gives each simple module of M exactly once.

In practice

- topR_i is only semi-simple
- Decompose it further as G_i-mod

Monoids are not semi-simple

Definition

- Semi-simple module: direct sum of simple modules
- Semi-simple group/monoid/algebra: all modules are semi-simple

- Groups are semi-simple
- Monoids are not semi-simple!
- Commutative case: are the roots simple?

New players in the game

- Composition factors
- Projective modules
- Cartan invariants matrix

Character table of a monoid, Mc Alister 1972

Definition (Character of an element x acting on a module V)

$\chi_V(x)$: trace of the matrix of x acting on V.

For each J-class, take one element $g_{i,j}$ in each conjugacy class of G_i.

Definition (Character of a module V)

$$\chi_V := \sum \chi_V(g_{i,j}) p_{i,j}$$

where $p_{i,j}$ are formal indeterminates.

Definition (Character table)

$$(\chi_{S_{i,j}}(g_{i',j'}))_{(i,j),(i',j')}$$

Properties of the character table

Theorem (Mc Alister 1972, T'11)

For an aperiodic monoid, the character table is upper unitriangular

For a monoid, the character table is upper block triangular, with the blocks being the character tables of the groups G_i

Corollary

Characters can be used to compute composition factors!

Cartan invariant matrix as linear refinement of $\mathcal{J}$-preorder

Definition

A: finite dimensional algebra (e. g. $A = \mathbb{Q}[M]$)
A is an A-mod-A module (or $A^{\mathrm{op}} \otimes A$-module)
Composition series: $\{0\} = A_0 \subset \cdots \subset A_\ell = A$

Proposition

$$A_{k+1}/A_k \approx_{A-\mathrm{mod}-A} S_i \otimes S_j^*$$

where S_i is a simple left module and S_j^* is a simple right module

See e.g. Curtis-Reiner.

The Cartan (invariants) matrix

Definition

$C = (c_{i,j})_{i,j}$, with:

$$c_{i,j} \quad = \quad |\{k,\ A_{k+1}/A_k \ \approx_{A-\mathrm{mod}-A}\ S_i \otimes S_j\}|$$

Equivalent definitions:

- On the left: $[P_j] = \sum_i c_{i,j}[S_i]$
- On the right: $[P_i] = \sum_j c_{i,j}[S_j]$
- Dimension of sandwiches by idempotents: $c_{i,j} = \dim e_i A e_j$

Cartan matrix by orthogonal idempotents

1. Build a decomposition of the identity into orthogonal idempotents e_i
2. Compute $e_i A e_j$
3. Build the projective modules as $e_i A$

Problem

Non trivial construction!

- *0-Hecke in type A: combinatorial formula [Denton'10]*
- *$\mathcal{R}$-trivial: recursive formula [Berg,Bergeron,Bhargava,Saliola'10]*
- *Aperiodic?*
- *Algebra: may require arbitrary algebraic extensions*

Idempotent free approach?

Cartan matrix of a monoid by characters

Let M be a finite monoid
$(g_i)_{i\in I}$: representatives of the conjugacy classes
Define the matrix $\overline{C} = (\overline{c}_{i,j})_{i,j\in I}$ by

$$\overline{c}_{i,j} := \left|\{s \in M, g_i s g_j^* = s\}\right|.$$

Lemma (T'11)

The bi-character of $\mathbb{K}M$ is given by

$$\chi_{\mathbb{K}M} = \sum_{i,j\in I} \overline{c}_{i,j} p_i \otimes p_j^*.$$

Cartan matrix for a general monoid

Algorithm to compute the Cartan matrix

- Build the simple modules (linear algebra)
- Compute the character table, i.e. the change of basis $p_i \mapsto s_i$
- Compute the matrix $\overline{C}$ (purely combinatorial)
- Apply the change of bases $p_i \otimes p_j^* \mapsto s_i \otimes s_j^*$

Cartan matrix for aperiodic monoids

Remark

- *The composition series of $\mathbb{Q}[M]$ refines the decomposition of M into $\mathcal{J}$-classes*
- *For J a $\mathcal{J}$-class of the form $L \times R$:*

$$\mathbb{Q}J \quad \approx_{\mathbb{Q}[M]-\mathsf{mod}-\mathbb{Q}[M]} \quad \mathbb{Q}L \otimes \mathbb{Q}R$$

Proposition (T'11)

M_L: decomposition matrix of left class modules into simples
M_R: decomposition matrix of right class modules into simples
Then, $C = M_{\mathcal{L}}^t M_{\mathcal{R}}$

Remark: $M_{\mathcal{L}}$ and $M_{\mathcal{R}}$ are upper unitriangular on regular $\mathcal{J}$-classes

Cartan matrix of aperiodic monoids

Algorithm [T'11]

Input: an aperiodic monoid

1. Construct representatives of left and right class modules
2. Construct the simple modules as quotients thereof
3. Compute the character table
4. Compute the character of each left and right class module
5. Compute the decomposition matrices $M_{\mathcal{L}}$ and $M_{\mathcal{R}}$

Output: The cartan matrix $C = M_{\mathcal{L}}^t M_{\mathcal{R}}$

Advantages of this algorithm

- Splits the linear algebra in small chunks
- Take advantage of the redundancy
- Rough complexity: $O(\sum_{i \in I} |R_i|^3)$
- Cartan matrix of a monoid of size 31103 in one hour
- Potential for parallelism!

Theoretical consequences

- Mostly characteristic free
- No algebraic extension needed (no surprise)
- Generalization to PIDs ($\mathbb{Z}$, ...)
- Completely combinatorial for $\mathcal{J}$-trivial monoids [Denton, Hivert, Schilling, T'2010]

Problems

- *Quiver?*
- *Socle/Radical filtration?*
- *Construction of projective modules*
- *Interesting examples in combinatorics?*

Cartan matrix of monoids in characteristic zero

Same principle, with some complications

Remark

For a $\mathcal{J}$-class J:

$$\mathbb{K}J \quad \approx_{\mathbb{K}M-\mathsf{mod}-\mathbb{K}M} \quad \mathbb{K}L \otimes_{\mathbb{K}G} \mathbb{K}R$$

Algorithm [T'11]

- For each $\mathcal{J}$-class J, regular or not
  - Compute the group G
  - Compute the M-mod-G character of $\mathbb{K}L$
  - Compute the G-mod-M character of $\mathbb{K}R$
  - Recombine the information to get the M-mod-M character of $\mathbb{K}J$
- Collect all and change basis

En résumé

- Convergence de deux communautés
- Semigroupes: toute une technologie
- Combinatoire algébrique: exemples intéressants
- Les progrès viennent d'idées simples
- Outil: exploration informatique

[

Implantation]

- Algorithmes très courts
- Beaucoup de prérequis
- Infrastructure objet avancée

Le projet Sage-Combinat (2000-2011)The Sage-Combinat projet (2000-2011)

MissionMission statement

« Améliorer MuPAD/Sage comme boîte à outils extensible pour l'exploration informatique en combinatoire algébrique, en fédérant et mutualisant les efforts de développements des chercheurs » "To improve MuPAD/Sage as an extensible toolbox for computer exploration in combinatorics, and foster code sharing among researchers in this area"

Stratégie

- Licence libre pour partager avec le plus grand nombre
  En restant pragmatique dans les collaborations
- Développement décentralisé et international
  Garantie d'indépendance vis-à-vis des tutelles
- Développé par des chercheurs pour des chercheurs
  Avec un usage plus large en vue
- Coeur du développement par des permanents

Le projet Sage-Combinat: 10 ans aprèsThe *-Combinat project: 10 years after

En quelques chiffresIn a nutshell

- MuPAD/C++: 600 classes, 5000 méthodes, 160k lignes de code+doc+tests
- Sage (Python): 300 tickets / 200k lignes de code+doc+tests
- FinancementsSponsors: ANR, PEPS, NSF, Google Summer of Code
- **70+ articles de rechercheresearch articles:**
  - Combinatoire algébrique, énumérative, des mots, ...
  - Test de programmes (LRI: Denise,Gaudel,Gouraud,Oudinet)

Une communauté internationaleAn international community

- Barcelonne, Davis, Lyon, Marne, Marseille, Montpellier, Orsay, Paris, Rouen, Philadelphie, Seattle, Stanford, Montreal, Toronto, ...
- Abbad, Berg, Borie, Bump, Bandlow, Boussicault, Chapoton, Delecroix, Dehaye,

Le projet Sage

Mission

« **Créer une alternative libre et viable à Maple™, Mathematica™, Magma™ et MATLAB™** »

En quelques chiffresIn a nutshell

- Initié en 2004 par Stein
- 40000 utilisateurs?
- 250 contributeurs

« **I certainly did not realise when the combinat people joined Sage how useful they and what they do would be for people like me!** »

John Cremona, number theorist.

Mon rôle dans Sage-Combinat

- **Organization et animation** de la communauté
  ($\approx$ 1000 mails par an)
- Formation, Conseil (modélisation, algorithmique, tests, ...),
  Revue de code, ...
- 2-3 invitations par an
- **Architecture logicielle**
  Élaboration de concepts et idiomes
- Développement logiciel
  (20 tickets, 20k lignes par an)

sagecombinat

Défi: gestion de grandes hiérarchies de classes