Résumé Cours PPC du MPRO
Complexité des problèmes de contraintes

Florent MADELAINE

1 Ubiquité des CSP et conjecture de la dichotomie.

Le problème de contraintes général est NP-complet puisqu'il permet de modéliser naturellement des problèmes difficiles bien connus comme SAT ou 3-COL. On cherche donc des restrictions du problème général pour lesquelles on dispose d'un algorithme efficace, c'est-à-dire qui s'exécute en temps polynomial. On parle donc des problèmes de contraintes. Ces problèmes de contraintes ont plusieurs définitions équivalentes.

- Définition IA classique « variables-valeurs-constraintes » ;
- Problème d'homomorphisme ;
- Inclusion de requêtes conjonctives ; et,
- Problème du *Model-checking* pour un fragment de FO[1].

De plus, on verra que la *complexité* peut être caractérisée *algébriquement* dans certains cas. L'étude des problèmes de contraintes et de leur complexité touche ainsi à plusieurs disciplines de mathématiques et d'informatique : algèbre, bases de données, combinatoire, complexité, logique.

Nous adoptons ici la définition en terme de problème d'homomorphisme. Vous avez dû rencontrer cette notion d'*homomorphisme* en algèbre pour des groupes, des anneaux, des corps *etc.* De manière générale, étant donné deux structures relationnelles $\mathcal{A}$ et $\mathcal{B}$ de même signature[2], il s'agit d'une application h (du domaine) d'une structure $\mathcal{A}$ vers (le domaine d')une structure $\mathcal{B}$ qui préserve les propriétés algébriques de ces structures. Dans le cas de structures relationnelles, l'application h doit vérifier que : pour tout symbole R d'arité r, pour tout tuple $a_1,a_2,...,a_r$ d'éléments de A, si $(a_1,a_2,...,a_r) \in R^{\mathcal{A}}$ alors $(h(a_1),h(a_2),...,h(a_r)) \in R^{\mathcal{B}}$. En particulier, pour le cas des graphes non orientés (une seule relation binaire qui est symmétrique) un homomorphisme envoie une arête du graphe $\mathcal{A}$ sur une arête du graphe $\mathcal{B}$. Si les graphes sont orientés (une seule relation binaire) alors on envoie un arc sur un arc en préservant l'orientation.

> CONSTRAINT SATISFACTION PROBLEM (CSP)
> - **instance** : deux structures relationnelles de même signature $\mathcal{A}$ et $\mathcal{B}$.
> - **question** : Est-ce-qu'il y a un homomorphisme de $\mathcal{A}$ dans $\mathcal{B}$?

Intuitivement, le domaine de $\mathcal{A}$ représente les variables de l'instance et celui de $\mathcal{B}$ les valeurs que ces variables peuvent prendre. Un tuple $(a_1,a_2,...,a_r)$ de la relation R de la structure $\mathcal{A}$ (ce qu'on note $R^{\mathcal{A}}$) signifie que la contrainte R porte sur les variables $a_1,a_2,...,a_r$. Un tuple de la relation correspondante $R^{\mathcal{B}}$ de la structure $\mathcal{B}$ liste une combinaison de valeurs autorisée par la contrainte R.

L'intérêt d'adopter la définition en terme d'homomorphisme est que les deux façons de restreindre le problème général apparaissent de manière naturelle. Les restrictions portent soit sur le *graphe des contraintes* (un graphe associé à la structure $\mathcal{A}$), soit sur le *langage des contraintes* (la structure $\mathcal{B}$). On

[1] FO = First Order, la logique du premier ordre.

[2] C'est-à-dire que les noms et arités des symboles de relations des deux structures correspondent. Par exemple, la signature pour des graphes orientés aura un seul symbole E binaire permettant de coder les arcs.

parle de problème de contrainte uniforme dans le premier cas et de problème de contrainte non-uniforme dans le second cas. [3]

NON-UNIFORM CONSTRAINT SATISFACTION PROBLEM (CSP($\mathcal{B}$))
- **paramètre** : une structure relationnelle $\mathcal{B}$ (le *langage des contraintes*).
- **instance** : une structure relationnelle de même signature $\mathcal{A}$.
- **question** : Est-ce-qu'il y a un homomorphisme de $\mathcal{A}$ dans $\mathcal{B}$?

UNIFORM CONSTRAINT SATISFACTION PROBLEM (CSP($\mathscr{C}$,_))
- **paramètre** : une signature relationnelle fixée σ et une classe de structures relationnelles $\mathscr{C}$ ayant cette signature.
- **instance** : deux structures relationnelles de même signature $\mathcal{A}$ et $\mathcal{B}$, où $\mathcal{A}$ appartient à $\mathscr{C}$.
- **question** : Est-ce-qu'il y a un homomorphisme de $\mathcal{A}$ dans $\mathcal{B}$?

Les problèmes de contraintes semblent suivre une *dichotomie* : une restriction est soit facile (dans P) soit difficile (NP-complet). Le premier résultat de dichotomie concerne le problème du H-coloriage, le cas d'un CSP non uniforme ayant pour langage de contrainte une seule relation binaire symmétrique (un graphe non orienté). Le second résultat concerne le cas des langages de contraintes de domaine booléen, qu'on peut voir comme une généralisation de SAT.

$\mathcal{H}$-COLORING
- **paramètre** : un graphe non orienté $\mathcal{H}$.
- **instance** : un graphe non orienté $\mathcal{G}$.
- **question** : Est-ce-qu'il y a un homomorphisme de $\mathcal{G}$ dans $\mathcal{H}$?

Si on prend pour le graphe $\mathcal{H}$ une clique de taille k, on obtient le problème du k-coloriage (k-COL).

Théorème 1 (Hell et Nešetřil)**.** *Le problème du $\mathcal{H}$-coloriage est polynomial si $\mathcal{H}$ est biparti et NP-complet sinon.*

GENERALIZED-SAT (SAT($\mathcal{B}$))
- **paramètre** : une structure relationnelle $\mathcal{B}$ de **domaine** $\{0,1\}$ (un *langage de contraintes booléen*).
- **instance** : une structure relationnelle de même signature $\mathcal{A}$.
- **question** : Est-ce-qu'il y a un homomorphisme de $\mathcal{A}$ dans $\mathcal{B}$?

Si on prend comme langage de contrainte toutes les relations correspondant aux modèles de 3-clauses (par exemple la relation $\{0,1\}^3 \setminus \{(000)\}$ pour coder la clause $x \vee y \vee z$), on a un problème NP-complet puisque 3-SAT est NP-complet. Si on prend comme langage de contrainte toutes les relations correspondant aux modèle de 2-clauses, on obtient un problème polynomial puisque 2-SAT est polynomial.

[3] Cette terminologie est dû au fait que si on disposait d'algorithmes polynomiaux pour CSP($\mathcal{B}$) où $\mathcal{B}$ appartient à une famille de langage de contrainte $\mathscr{C}$, rien ne garantie *a priori* qu'on puisse disposer d'un algorithme uniforme pour CSP(_,$\mathscr{C}$) (qu'on définit de manière analogue à CSP($\mathscr{C}$,_)). Il y a d'une part le fait que $\mathcal{B}$ fait partie de l'entrée ce qui peut rendre l'algorithme dédié non polynomial, et d'autre part, même si c'était le cas, pour pouvoir utiliser l'algorithme dédié il faudrait être capable de détecter efficacement lorsqu'on peut l'employer. Dans ce second cas, on parle souvent de *méta-problème* : puis-je décider si je me trouve dans un cadre polynomial, et si oui quel algo efficace je dois choisir ?

Théorème 2 (Schaefer). *Pour tout langage de contrainte booléen $\mathcal{B}$, le problème de contrainte non-uniforme* $\textsc{Csp}(\mathcal{B})$ *est polynomial si $\mathcal{B}$ est* 0*-valide,* 1*-valide, Horn, dual-Horn, bijonctif ou affine, sinon le problème est NP-complet.*

La complexité des problèmes de contraintes non-uniformes semble s'expliquer par des *propriétés algébriques* des langages de contraintes. On verra comment on peut s'appuyer sur le *treillis des clones booléens* pour démontrer le théorème de Schaefer. On ne le verra pas mais on peut aussi montrer que le théorème de Hell et Nešetřil s'explique algébriquement. Intuitivement, l'expressivité d'un langage de contrainte est caractérisé par les fonctions qui préservent les relations de ce langage. Lorsqu'un langage est préservé par certaines fonctions particulières, alors on peut montrer qu'il existe un algorithme polynomial. Par exemple, lorsqu'on considère les langages de type Horn – ceux pour lesquels les relations sont préservées par l'opération booléenne $\wedge$ – on peut montrer qu'une généralisation de la méthode d'arc-cohérence est complète.

Ce phénomène de dichotomie ne correspond pas à ce qu'on devrait observer en général, puisque un corollaire du théorème de Ladner est que si P est différent de NP, il existe des problèmes qui ne sont ni dans P ni NP-complet. En fait, la classe des problèmes de contraintes est dans un certain sens la plus large qui admetrait une dichotomie. Les deux résultats ci-dessus et d'autres résultats de dichotomie partielle ont conduit Feder et Vardi a postuler la conjecture suivante.

Conjecture 3 (Feder et Vardi). *Pour tout langage de contrainte $\mathcal{B}$, le problème de contrainte non-uniforme* $\textsc{Csp}(\mathcal{B})$ *est soit polynomial, soit NP-complet.*

En ce qui concerne les restrictions du graphe des contraintes, la dichotomie est établie si on suppose que l'arité des contraintes est borné par une constante (ce qui est le cas dans notre définition).

Théorème 4 (Dalmau, Vardi, Kolaitis et Grohe). *Sous couvert d'une hypothèse de complexité paramétrée ($FPT \subsetneq W[1]$, analogue de la conjecture $P \subsetneq NP$), on a la dichotomie suivante. Le problème de contrainte uniforme* $\textsc{Csp}(\mathscr{C},\_)$ *est polynomial si il existe une constante k telle que pour toute structure $\mathcal{A}$ de $\mathscr{C}$ la* largeur d'arborescence *du* core *de $\mathcal{A}$ est au plus k, et NP-complet sinon.*

Faute de temps, nous n'aborderons pas les problèmes de contraintes uniforme et nous concentrons dans ce cours sur les problèmes de contraintes non-uniformes et démontrons le théorème de Schaefer.

2 Exercices

Complexité de k-Sat

1. Montrez que 1-SATpeut être résolu en temps polynomial.
2. Montrez que 2-SAT peut être résolu en temps polynomial.
3. Montrez que si 3-SAT peut être résolu en temps polynomial alors SAT peut l'être aussi.

Complexité de Horn-Sat

1. Soit φ une formule de Horn à n variables. On suppose qu'il existe une clause constituée d'un seul litéral positif x dans φ. Montrez que satisfaire φ revient à satisfaire une formule de Horn à $n-1$ variables.
2. En utilisant le fait qu'on peut écrire une clause de Horn sous forme implicative, en déduire un algorithme de résolution pour HORN-SAT.
3. Quelle est sa complexité dans le pire des cas en fonction de n (nombre de variables) et m (nombre de clauses).

4. Appliquer cet algorithme à
$$\varphi := (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_2 \vee \neg x_4) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_4 \vee \neg x_3) \wedge (x_3) \wedge (x_5 \vee \neg x_1 \vee \neg x_2 \vee \neg x_3)$$

Complexité du coloriage

1. Montrez que 2-col est polynomial.
2. Donnez un programme DATALOG pour le complément de 2-col.
3. Montrez que H-col se réduit à 2-col lorsque H est un graphe biparti avec au moins une arête.
4. Montrez que 2-col se réduit à H-col lorsque H est un graphe biparti avec au moins une arête.

Preuve du théorème de Chandra Merlin

Inclusion de requêtes : $\varphi_{\mathcal{B}} \subseteq \varphi_{\mathcal{A}}$ est une abréviation pour :
pour toute base de donnée $\mathcal{D}$, si $D \models \varphi_{\mathcal{B}}$ alors $D \models \varphi_{\mathcal{A}}$. Démontrez le théorème suivant.

Théorème 5 (Chandra, Merlin '77)**.** *Il y a équivalence entre les points suivants.*

- $\varphi_{\mathcal{B}} \subseteq \varphi_{\mathcal{A}}$
- $\mathcal{B} \models \varphi_{\mathcal{A}}$
- $\mathcal{A} \to \mathcal{B}$

3 Les cas polynomiaux du théorème de Schaefer

Puisque dans la suite de ce cours nous nous penchons sur les problèmes ayant le même domaine {0,1}, nous allons paramétrer le problème GENERALIZED-SAT non pas en terme de $\mathcal{B}$ mais en terme des relations booléennes Γ de $\mathcal{B}$.

GENERALIZED-SAT (SAT(Γ))
- **paramètre** : un ensemble fini de relations booléennes Γ.
- **instance** : une structure relationnelle $\mathcal{A}$.
- **question** : Est-ce-qu'il y a un homomorphisme de $\mathcal{A}$ dans $\mathcal{B}$ ($\mathcal{B}$ est la structure relationnelle dont les relations sont Γ) ?

Dans l'immédiat il s'agit juste d'avoir une notation plus pratique. On verra plus tard qu'on peut en fait considérer des restrictions où Γ est arbitraire (pas forcément fini).

Des fonctions booléennes importantes. On considère les 6 fonctions booléennes suivantes.
- c_0 est la fonction unaire constante égale à 0 et c_1 celle qui est égale à 1 ;
- $\wedge$,$\vee$ les fonctions booléennes binaires « et » (min) et « max » (ou) ;
- m la fonction ternaire dite de *majorité* qui retourne l'argument le plus fréquent e.g. $m(0,0,1)=1$; et,
- l la fonction ternaire linéaire $l(x,y,z):=x+y+z \mod 2$.

Préservation. Une fonction booléenne f d'arité n *préserve* une relation R d'arité m (on dira aussi que R est *invariante* ou *close* par m) si, et seulement si, pour tous tuples $t_1,t_2,...,t_n$ appartenant à cette relation R, le tuple $f(t_1,t_2,...,t_r)$ appartient aussi à R.
Si $t_1=(a_{11},...,a_{1m})...t_n=(a_{n1},...,a_{nm})$ le tuple $f(t_1,...,t_r)$ est obtenue en appliquant f composante par composante :

$$\begin{array}{ccccccc} & f & & f & & f & \\ (& a_{11} & , & \dots & , & a_{1m} &) \in R \\ & \vdots & & \vdots & & \vdots & \vdots \\ (& a_{n1} & , & \dots & , & a_{nm} &) \in R \\ \hline (& f(a_{11},...,a_{n1}) & , & \dots & , & f(a_{1m},...,a_{nm}) &) \in R \end{array}$$

Si on considère la relation $R=\{(0,0,0),(1,0,0),(0,0,1)\}$. On peut facilement vérifier que cette relation R est close pour $\wedge$. Par exemple,

$$\begin{array}{ccccccc} & \wedge & & \wedge & & \wedge & \\ (& 1 & , & 0 & , & 0 &) \in R \\ (& 0 & , & 0 & , & 1 &) \in R \\ \hline (& 0 & , & 0 & , & 0 &) \in R \end{array}$$

(on peut vérifier les autres cas). Mais l'opération binaire $\vee$ ne préserve pas R puisque,

$$\begin{array}{ccccccc} & \vee & & \vee & & \vee & \\ (& 1 & , & 0 & , & 0 &) \in R \\ (& 0 & , & 0 & , & 1 &) \in R \\ \hline (& 1 & , & 0 & , & 1 &) \notin R \end{array}$$

Définitions 5.1. Soit Γ un langage booléen.
- Γ est 0-*valide* (respectivement, 1-*valide*) si les relations de Γ sont closes par c_0 (respectivement, c_1) ;

- Γ est *Horn* (respectivement, *dual-Horn*) si les relations de Γ sont closes par $\wedge$ (respectivement, $\vee$) ;
- Γ est *bijonctif* si les relations de Γ sont closes par m ; et,
- Γ est *affine* si les relations de Γ sont closes par l.

Il est clair que lorsque le langage est 0-valide ou 1-valide, le problème est trivial (soit une relation est vide et l'instance non satisfaite, soit l'assignement constant correspondant est solution). Lorsque Γ est Horn on peut réduire le problème à HORN-SAT (et dual-Horn à la version duale de HORN-SAT). Lorsque Γ est bijonctif, on peut réduire le problème à 2-SAT. Lorsque Γ est affine, on peut se ramener à un système linéaire qu'on peut résoudre en temps polynomial (on ne traitera pas ce cas dans ce cours). Alternativement, on peut proposer des algorithmes polynomiaux sans faire de réduction à SAT. L'avantage de cette méthode est que ces algorithmes sont polynomiaux même si les relations booléennes décrivant les contraintes font partie de l'entrée.

Lemme 6. *Si Γ est Horn, alors l'algorithme GAC (Generalized Arc Consistency) résoud* SAT(Γ).

Démonstration. L'algorithme GAC est l'analogue de AC dans le cas où les contraintes ne sont pas forcément binaires : on enlève une valeur du domaine d'une variable, si cette valeur n'est pas supportée par au moins un tuple autorisé pour chaque contrainte portant sur cette variable ; on répète ce filtrage jusqu'à stabilité des domaines. Si GAC vidait le domaine d'une variable, on peut répondre INSATISFAIT (pour n'importe quelle instance). Si ce n'est pas le cas, on ne sait pas en général si il existe une solution. Cependant, dans le cas spécifique d'un langage de Horn, on peut montrer que l'assignement consistant à prendre le minimum de chaque domaine après un appel de GAC est solution. Voir les exercices pour les détails. □

Lemme 7. *Si Γ est bijonctif, alors l'algorithme 3-consistency (Path Consistency) résoud* SAT(Γ).

Démonstration. La preuve est similaire au cas précédent. Si l'algorithme établissant la 3-cohérence vidait un domaine, on peut toujours répondre INSATISFAIT. Dans le cas spécifique d'un langage bijonctif, on peut utiliser la fonction majorité m, pour reconstruire une solution. Voir les exercices pour les détails. □

Proposition 8. *Si Γ est 0-valide (clos par c_0), 1-valide (clos par c_1), Horn (clos par $\wedge$), dual Horn (clos par $\vee$), bijonctif (clos par m) ou affine (clos par l) alors* SAT(Γ) *peut être résolu en temps polynomial.*

4 Exercices

Inter Réduction entre Sat et Generalized Sat

1. Réduisez k-SAT à GENERALIZED SAT.
2. On considère le cas de GENERALIZED SAT restreint à une seule relation R d'arité r. Réduisez ce problème à SAT.

Clôture des modèles de certaines clauses particulières

1. Soit C une clause de Horn à r variables. Soit R_C l'ensemble des assignements (un assignement est vu comme un tuple de r valeurs) t satisfaisant la clause C. Montrez que R_C est clos par $\wedge$: c'est-à-dire que pour tout t_1 et t_2 de R_C le tuple $t_1 \wedge t_2$ (on applique $\wedge$ composante par composante) est dans R_C.
2. Montrez comment simuler la relation $R = \{000,100,010,011,111\}$ par plusieurs clauses de Horn.
3. Montrez comment simuler une relation R close par $\wedge$ par plusieurs clauses de Horn.

4. Donnez un algorithme polynomial pour une restriction de GENERALIZED SAT à des relations de Horn (indications : définir une généralisation de la notion de cohérence d'arc et utiliser la clôture par $\wedge$ pour guider le choix d'une solution particulière lorsque l'instance est cohérente).
5. Soit C une 2-clause et R_C la relation binaire des assignements satisfaisant cette clause C. Montrez que R_C est close par l'opération de majorité m. Cette opération ternaire retourne l'argument le plus fréquent : e.g. $m(0,0,0)=m(0,0,1)=m(1,0,0)=m(0,1,0)=0$.
6. Soit R une relation close par l'opération m. Réduisez la restriction de GENERALIZED SAT au langage de contrainte $\{R\}$ à 2-Sat (indication : remplacer la contrainte R par toutes les contraintes induites par les projections sur deux coordonnées).

5 Expressivité

Si on autorisait n'importe quelle réduction polynomiale, alors on pourrait réduire n'importe quelle version polynomiale de notre problème à HORN-SAT, puisque ce problème est complet pour la classe P. Dans le cadre des problèmes de contraintes, il est plus naturel de n'autoriser que des réductions *primitive positive* : ceci correspond aux contraintes implicites qu'on peut simuler en combinant un nombre fini de contraintes du langage à l'aide de variables supplémentaires si nécessaire.

Par exemple, si Γ contient deux relations binaire R_1 and R_2, codant respectivement l'existence d'une connexion par train et par vélib, alors dans l'instance

$$((x,z),R_1),((z,y),R_2)$$

la contrainte *implicite* sur (x,y) est $R_3 = R_1 \circ R_2$, où $R_3(x,y) = \exists z R_1(x,z) \wedge R(z,y)$. Cette contrainte implicite R_3 exprime la contrainte : connexion en train puis vélib. Ainsi $R3 \notin \Gamma$, mais les langages Γ et $\Gamma \cup \{R_3\}$ sont équivalents (du point de vue de l'expressivité). On peut montrer de surcroît que les problèmes de contraintes associés sont de complexité équivalentes (inter-réductibles en temps polynomial) puisqu'on peut toujours remplacer une occurrence de R_3 comme $R_3(x,y)$ par $\exists z R_1(x,z) \wedge R_2(z,y)$ où z est une variable annexe. Cette idée se généralise naturellement.

On note par $[\Gamma]$ l'ensemble des relations qui peuvent être *simulées* par les relations de Γ. Combinatoirement, il s'agit de toutes les relations qu'on peut construire en fabriquant des « gadgets » à l'aide de variable annexes et de relations de Γ. Plus formellement, il s'agit de toutes les relations qui sont interprétables par une *formule primitive positive* sur Γ, c'est-à-dire utilisant

- des relations de Γ,
- l'égalité $=$,
- la conjonction $\wedge$, et
- la quantification existentielle $\exists$.

Théorème 9 (Jeavons)**.** *Si Γ_1 et Γ_2 sont des langages de contraintes tels que Γ_1 est fini et $[\Gamma_1] \subseteq [\Gamma_2]$ alors* $\text{SAT}(\Gamma_1)$ *se réduit à* $\text{SAT}(\Gamma_2)$.

Démonstration. Il suffit de simuler chaque relation de Γ_1 par des relations de Γ_2. Puisque $[\Gamma_1] \subseteq [\Gamma_2]$, pour toute relation R_1 de $[\Gamma_1]$ (et donc de Γ_1) il existe une formule primitive positive φ_{R_1} utilisant un nombre fini de relations de Γ_2 telle que φ_{R_1} définit R_1. Si on prend une définition logique (en terme de model checking) de CSP, la réduction consiste à remplacer toute occurrence de R_1 par φ_{R_1} dans l'instance. Si on prend une définition combinatoire (en terme d'homomorphismes), la réduction consiste à remplacer chaque tuple de R_1 par le « gadget » correspondant à φ_{R_1}. Voir les exercices pour plus de détails. □

6 Exercices

Expressivité d'un langage

1. Simulez la relation booléenne $x \neq y$ par des 2-clauses. En déduire une réduction de 2-col à 2-Sat.
2. Donnez une réduction de K_5-Col à C_5-Col. Quelle est la complexité de C_5-Col ?
3. Montrez que si H était un graphe polynomial non biparti, ayant un nombre de sommets minimal puis parmi ceux-ci un nombre d'arêtes maximal alors H contiendrait nécessairement un triangle.
4. Soit $R_1(x_1,x_2,x_3)$ le modèle de $(x_1 \vee \neg x_2 \vee \neg x_3)$ et $R_2(x_3)$ celui de x_3. Calculez la relation R' définie par
$$\exists x_3 R_1(x_1,x_2,x_3) \wedge R_2(x_3).$$
5. Exprimez la réduction de K_5 à C_5 comme une p.p. réduction. On rappelle qu'une *p.p. réduction* est déterminée par une formule du premier-ordre utilisant les symboles de relations de Γ, la conjonction $\wedge$, la quantification existentielle $\exists$ et l'égalité $=$.

7 Théorie des clones

Toutes les relations et fonctions considérées dans cette section sont booléennes. Les résultats de cette section se généralisent à un domaine fini quelconque. Pour $n \geq 1$ et $1 \leq i \leq n$, la *projection* e_i^n est la fonction d'arité n telle que $e_i^n(x_1,x_2,...,x_n) := x_i$.

Définition 9.1 (Clones)**.**

- Soit Γ un ensemble de relations. $[\Gamma]$ (l'ensemble des relations que l'on peut simuler par p.p. réduction depuis les relations de Γ) est appelé *clone relationnel* généré par Γ.
- Soit F un ensemble d'opérations. Soit $\langle F \rangle$ l'ensemble des opérations obtenues par composition des opérations de F (et des projections).
  E.g. si $f_1, f_2 \in F$ alors $f_2(f_1(x,y),x,f_2(f_1(z,y),x,z)) \in \langle F \rangle$.
  On dit que $\langle F \rangle$ est le *clone (fonctionnel)* généré par F.

Si on considère les clones relationnels ou fonctionnels par rapport à l'inclusion, il est facile de montrer qu'on obtient en fait un treillis. Par exemple, pour deux clones relationnels $\Gamma_1 = [\Gamma_1]$ et $\Gamma_2 = [\Gamma_2]$, on a
$$\Gamma_1 \wedge \Gamma_2 := [\Gamma_1 \cap \Gamma_2]$$
et
$$\Gamma_1 \vee \Gamma_2 := [\Gamma_1 \cup \Gamma_2]$$
Le plus petit clone relationnel est celui qui est généré par l'ensemble vide $[\emptyset]$ et ne contient que des relations généralisant l'égalité (puisqu'on autorise l'utilisation de $=$ dans les p.p. réductions). Ce clone contient donc par exemple : {00,11} (via la p.p. réduction $x_1 = x_2$) ou encore {00000,11000,00011,11011,00100, (via la p.p. réduction $x_1 = x_2 \wedge x_4 = x_5$). Le plus grand clone relationnel contient toutes les relations possibles.

Le plus petit clone (fonctionnel) contient toutes les projections. Le plus grand clone est celui qui contient toutes les fonctions.

Correspondance de Galois. Nous allons voir que ces deux treillis sont anti-isomorphes. On peut passer de l'un à l'autre via la notion de préservation / invariance. De plus, si on prend un ensemble de relations qui n'est pas forcément clos pour l'expressivité [_], faire l'aller-retour en utilisant les anti-isomorphismes revient à calculer la clôture par [_] (il en est de même dans l'autre sens pour le

calcul de la clôture par $\langle\_\rangle$ d'un ensemble de fonctions). Cette relation particulière entre treillis d'objets clos s'appelle une *correspondance de Galois (antitone)* en algèbre universelle[4].

Définition 9.2. Soit Γ un ensemble de relations booléennes et F un ensemble de fonctions booléennes.

- $\mathrm{Pol}(\Gamma)$ dénote l'ensemble des fonctions qui préservent toutes les relations de Γ (on parle des ***pol****ymorphismes* de Γ, d'où la notation).
- $\mathrm{Inv}(F)$ dénote l'ensemble des relations **inv**ariantes par les fonctions de F.

Théorème 10 (Geiger '68 ; Bodnarchuk et al. '69)**.** *Les opérateurs* Pol *et* Inv *établissent une correspondance de Galois entre le treillis des clones relationnels et celui des clones.*

- *Pour tout langage de contrainte* Γ, $[\Gamma] = \mathrm{Inv}\big(\mathrm{Pol}(\Gamma)\big)$
- *Pour tout ensemble d'opérations* F, $\langle F\rangle = \mathrm{Pol}\big(\mathrm{Inv}(F)\big)$

Notons en particulier, que les projections sont les seules fonctions qui préservent l'ensemble de toutes les relations et que les relations définissables à partir de l'égalité sont préservées par toutes les fonctions (voir Fig. 1 pour une illustration).

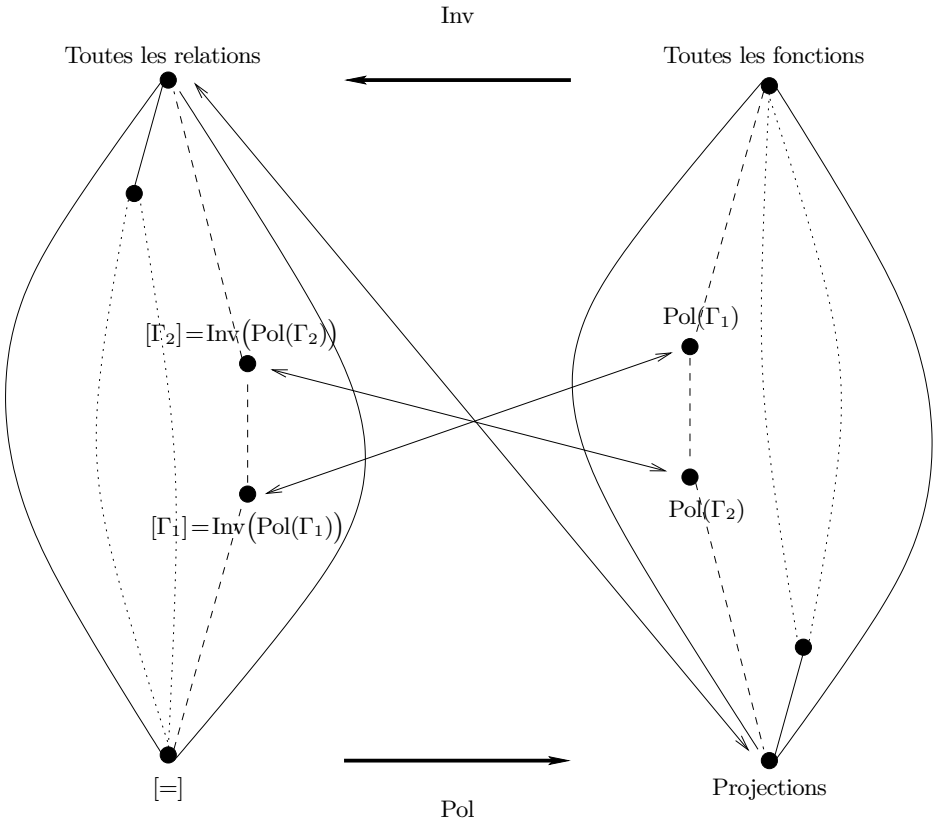

FIGURE 1 : Correspondance entre clones relationnels et clones fonctionnels

Le treillis des clones est connu sous le nom de *treillis de Post* dans le cas booléen : ce treillis est complètement décrit et pour chaque clone, on dispose d'une description dans le treillis fonctionnel (ensemble

[4]C'est une abstraction, pour deux ordres partiels quelconques, de la correspondance entre sous-corps d'une extension galoisienne et sous-groupes de son groupe de Galois.

générateur du clone) comme dans le treillis relationnel (ensemble générateur du clone relationnel). Seule une toute petite partie du treillis de Post nous concerne ici. Il s'agit des clones dit minimaux, c'est-à-dire ceux qui sont immédiatement au dessus du clone ne contenant que les projections et côté relationnel des clones dit maximaux. Il y a 7 clones minimaux et 6 d'entres eux correspondent aux clones relationnels / langages de contrainte polynomiaux que nous avons évoqué. En effet, il s'agit des clones générés respectivement par les opérations c_0,c_1,$\wedge$,$\vee$,m et l. Le dernier clone minimal est généré par la fonction booléenne $\neg$. Au dessus de ce clone dans le treillis on tombe sur des clones qui contiennent forcément l'une des 6 opérations. En conséquence, en passant par l'opérateur Inv du côté relationnel, ceci signifie que tout langage de contrainte strictement moins expressif que Inv($\neg$) est forcément polynomial (puisqu'il sera clos par l'une des 6 opérations caractérisant les langages polynomiaux). Voir Fig. 2 pour une illustration. Pour conclure notre preuve du théorème de Schaefer, il reste à montrer que si $\mathrm{Pol}(\Gamma) \subseteq \langle \neg \rangle$ alors $\mathrm{SAT}(\Gamma)$ est NP-complet.

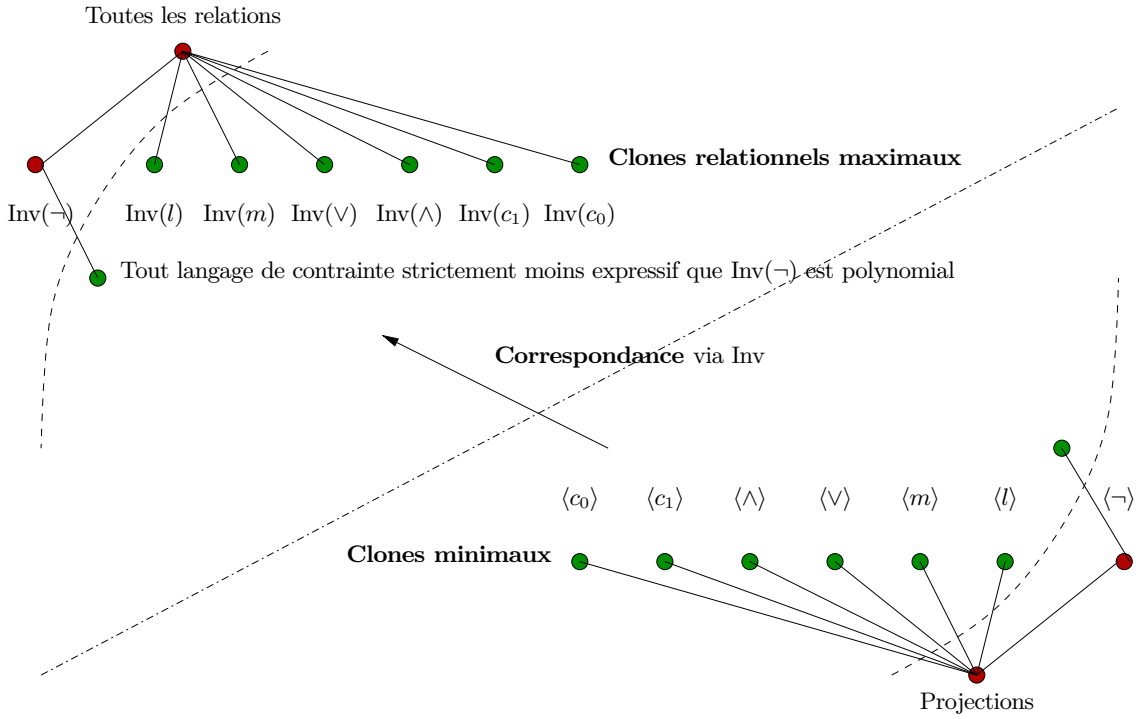

FIGURE 2 : Correspondance entre clones minimaux et clones relationnels maximaux.

8 Les cas NP-complets du théorème de Schaefer

Pour conclure notre preuve du théorème de Schaefer, il reste à montrer que si $\mathrm{Pol}(\Gamma) \subseteq \langle \neg \rangle$ alors $\mathrm{SAT}(\Gamma)$ est NP-complet. Pour cela on va utiliser une variante du théorème de Jeavons (voir lemme ci-dessous) et une seule relation ternaire {001,010,011,100,101,110} , correspondant au problème NP-complet NOT-ALL-EQUAL-3-SAT.

On a vu avec le théorème de Jeavons que [_] correspond à l'expressivité du langage de contrainte. On peut maintenant utiliser la correspondance de Galois pour passer dans le treillis des clones (fonctionnels). Ainsi comme annoncé, on observe que les polymorphismes contrôlent le pouvoir d'expression.

Lemme 11. *Soit R une relation et Γ un langage de contraintes fini. Si* $\mathrm{Pol}(\{R\}) \supseteq \mathrm{Pol}(\Gamma)$ *alors* $\textsc{Sat}(\{R\})$ *se réduit à* $\textsc{Sat}(\{\Gamma\})$.

Démonstration. On utilise la correspondance de Galois : $\mathrm{Pol}(\{R\}) \supseteq \mathrm{Pol}(\Gamma)$ implique $[\{R\}] = \mathrm{Inv}\Big(\mathrm{Pol}(\{R\})\Big) \subseteq \mathrm{Inv}\Big(\mathrm{Pol}(\Gamma)\Big) = [\Gamma]$. Donc par le théorème de Jeavons, $\textsc{Sat}(\{R\})$ se réduit à $\textsc{Sat}(\{\Gamma\})$. □

Proposition 12. *Si Γ n'est pas clos par l'une des 6 opérations suivantes c_0,c_1,$\wedge$,$\vee$,l,m, alors* $\textsc{sat}(\Gamma)$ *est NP-complet.*

Démonstration. D'après la description du treillis de Post on peut conclure des hypothèses que $\mathrm{Pol}(\Gamma)$ est soit trivial (toutes les projections), soit le clone minimal $\langle\neg\rangle$. La relation ternaire $R := \{001,010,011,100,101,110\}$ (permettant de coder Not-All-Equal-3-Sat) est manifestement préservée par $\neg$. On peut facilement vérifier que cette relation n'est préservée par aucune des fonctions c_0,c_1,$\wedge$,$\vee$,m ou l. Donc on a $\mathrm{Pol}(\{R\}) = \langle\neg\rangle \supseteq \mathrm{Pol}(\Gamma)$ et d'après le lemme ci-dessus $\textsc{Sat}(\{R\})$ se réduit à $\textsc{Sat}(\{\Gamma\})$. Il reste donc seulement à montrer que Not-All-Equal-3-Sat est NP-complet (voir exercices). □

9 Exercices

Autour de notre preuve du théorème de Schaefer

1. Pour chacune des relations suivantes, indiquez si elles sont closes ou non pour chacune des opération suivantes c_0,c_1 (opérations constantes), $\wedge$,$\vee$,l (opération ternaire : somme modulo 2 de ces arguments), m (majorité ternaire) et $\neg$ (unaire, complément).
   $R_1 = \{000,100,010,011,111\}$ $R_2 = \{000,111\}$
   $R_3 = \{01,11\}$ $R_4 = \{01,10\}$
   $R_5 = \{100,010,001\}$ $R_6 = \{001,010,011,100,101,110\}$
2. Montrez que si une opération f préserve un ensemble de relations Γ, alors f préserve $[\Gamma]$, où $[\_]$ indique l'ensemble des relations qu'on peut exprimer par *p.p.* définitions à partir de celles de Γ. (indication : procédez par récurrence sur la structure de la formule p.p.).
3. Démontrez le théorème de Jeavons.
4. Démontrez que Not-All-Equal-3-Sat est NP-complet.
   Indications. On prendra initialement une variante du problème autorisant les littéraux positifs et négatifs. Faites la réduction en deux temps depuis 3-Sat via le cousin du problème considéré Not-All-Equal-4-Sat. L'idée consiste à ajouter une variable spéciale θ pour toute la formule : un assignement dans lequel z prend la même valeur que θ étant traduit en un assignement où z est faux. La réduction de Not-All-Equal-4-Sat à Not-All-Equal-3-Sat est similaire à celle de k-Sat à 3-Sat.
5. Démontrez qu'on ne peut pas simuler (au sens de la p.p. définissabilité) les relations de 3-Sat avec la relation de Not-All-Equal-3-Sat.
6. Démontrez sans faire de réduction explicite que 1-In-3-Sat (vu comme le CSP booléen avec le langage $\{R_5\}$) est NP-complet par réduction depuis Not-All-Equal-3-Sat (vu comme le CSP booléen avec le langage $\{R_6\}$).

10 Une forme générale du théorème de Schaefer

Nous avons démontré une dichotomie pour les problèmes CSP($\mathcal{B}$), où $\mathcal{B}$ est une structure booléenne. Nous avons vu que ce qui importe pour la complexité ce sont les polymorphismes des relations Γ de la structure $\mathcal{B}$. On peut en fait donner une forme un peu plus générale de ce résultat de dichotomie.

Uniformisation Nous avons supposé jusqu'à présent que Γ était fixé et ne faisait pas partie de l'instance (version dite non-uniforme de CSP). On peut en fait proposer une version un peu plus générale du problème où Γ n'est pas forcément fini, et où les relations de Γ font partie de l'instance. Ceci est dû au fait que l'algorithme est le même pour tous les langages du même type (il n'y a qu'un algo pour Horn) et que les algorithmes proposés sont polynomiaux même si les relations listant les tuples autorisés de chaque contrainte font partie de l'instance.

GENERALIZED-SAT (SAT(Γ))

version uniforme

- **paramètre** : un ensemble (arbitraire) de relations Γ.
- **instance** : une conjonction φ d'atomes positifs $R(\bar{x})$, où R est un symbole d'une relation de Γ.
- **question** : est-ce-que φ est satisfaisable ?

Théorème 13. *(version uniforme) Si* Γ *est* 0*-valide,* 1*-valide, Horn, dual-Horn, bijonctif ou affine alors* SAT(Γ) *est polynomial sinon* SAT(Γ) *est NP-complet.*

Méta-problème. Par ailleurs, puisque les cas polynomiaux sont caractérisés comme étant ceux dont les relations sont closes par certaines opérations booléennes, on peut montrer que le méta-problème est polynomial.

MÉTA-PROBLÈME

- **instance** : un langage de contraintes booléen fini Γ.
- **question** : est-ce-que SAT(Γ) est polynomial ou NP-complet ?

Proposition 14. *Le méta-problème est polynomial.*

Démonstration. Soit P=NP et on répond toujours oui, soit P≠NP et on utilise le théorème de Schaefer. On peut détecter en temps polynomial si SAT(Γ) est polynomial puisqu'il suffit de tester si les relations de Γ sont closes pour l'une des 6 opérations booléennes c_0,c_1,$\wedge$,$\vee$,m ou l (pour une relation la complexité est $O(e^3m)$, où e est le nombre de tuples et m l'arité puisqu'on teste au pire une fonction ternaire nécessitant de vérifier $\binom{e}{3} = O(e^3)$ possibilités). □

Algorithme adaptatif. On peut donc proposer un algorithme adaptatif. Si les relations de l'instance appartiennent à un langage de contraintes polynomial, on le détecte en temps polynomial (puisque le méta-problème est polynomial) puis on appelle l'algorithme de résolution polynomial associé. Sinon on utilise la méthode générale Search + propagation.

11 Exercices

Utilisation du théorème de Schaefer

1. Quelle est la complexité de Generalized Sat pour les langages suivants ?
   $\Gamma_1 = \{R_1, R_2, R_3\}$ $\Gamma_2 = \{R_1, R_4\}$
   $\Gamma_3 = \{R_2, R_4\}$ $\Gamma_4 = \{R_3, R_6\}$

2. Montrez que pour tous les cas polynomiaux du théorème de Schaefer, on peut également trouver une solution en temps polynomial si elle existe (indication : on utilisera les algos du théorème de Schaefer comme des boîtes noires).
3. Quelle est la complexité du model checking pour le fragment $\exists, \wedge, \neq$ de la logique du premier ordre lorsqu'on prend le modèle comme paramètre ? Plus précisément, donner une classification complète de la complexité du problème de décision suivant selon le choix de $\mathcal{B}$.

> $\{\exists, \wedge, \neq\}$-FO($\mathcal{B}$)
> - **paramètre** : une structure $\mathcal{B}$.
> - **entrée** : une formule φ du premier ordre utilisant seulement les symboles $\exists, \wedge, \neq$ et les symboles de relation de $\mathcal{B}$.
> - **question** : est-ce-que $\mathcal{B} \models \varphi$?

12 L'approche duale : restreindre le graphe des contraintes

Nous avons évoqué dans l'introduction, le résultat suivant.

Théorème 15 (Dalmau, Vardi, Kolaitis et Grohe)**.** *Sous couvert d'une hypothèse de complexité paramétrée ($FPT \subsetneq W[1]$, analogue de la conjecture $P \subsetneq NP$), on a la dichotomie suivante. Le problème de contrainte uniforme* Csp($\mathscr{C}$,_) *est polynomial si il existe une constante k telle que pour toute structure $\mathcal{A}$ de $\mathscr{C}$ la* largeur d'arborescence *du* core *de $\mathcal{A}$ est au plus k, et NP-complet sinon.*

Nous ne démontrons pas ce résultat faute de temps. Nous expliquons juste ici la phrase « *largeur d'arborescence* du *core* de $\mathcal{A}$ ».

Le *core* d'une structure $\mathcal{A}$ est la[5] plus petite structure $\mathcal{A}'$ qui est homomorphiquement équivalente à $\mathcal{A}$, c'est-à-dire telle qu'il existe un homomorphisme de $\mathcal{A}$ à $\mathcal{A}'$ et un homomorphisme de $\mathcal{A}'$ à $\mathcal{A}$.

La *largeur d'arborescence d'une structure* est celle de son *graphe primal*[6] : ce graphe a un sommet pour chaque élément du domaine de la structure et deux sommets sont adjacents si ils appartiennent tous les deux à un tuple d'une relation de la structure. Dans le cas spécial d'une structure $\mathcal{A}$ « de variables », le graphe primal est appelé *graphe des contraintes*.

La *largeur d'arborescence d'un graphe* est le plus petit entier k tel qu'il existe un arbre de décomposition de largeur k. Les sommets d'un tel arbre sont des ensembles de sommets du graphe (on parle informellement de sac).

- un sac contient au plus $k+1$ sommets[7].
- Pour chaque sommet du graphe, l'ensemble des sacs contenant ce sommet doit former un sous-arbre.

[5] On peut montrer que cette structure est unique à isomorphisme près.

[6] On parle aussi de graphe de Gaifman.

[7] Ce +1 est utilisé pour des raisons esthétiques : ainsi les graphes qui sont des arbres ont pour largeur arborescente 1.

- Les sommets voisins du graphe doivent appartenir simultanément à au moins un sac.

Un résultat bien connu et simple à montrer est le suivant.

Théorème 16 (Dalmau, Vardi, Kolaitis et Grohe)**.** *Le problème de contrainte uniforme* $\textsc{Csp}(\mathscr{C},\_)$ *est polynomial si il existe une constante* k *telle que pour toute structure* $\mathcal{A}$ *de* $\mathscr{C}$ *la* largeur d'arborescence *du graphe des contraintes de* $\mathcal{A}$ *est au plus* k*.*

Démonstration. Puisque k est une constante fixée, on peut construire une décomposition du graphe des contraintes de $\mathcal{A}$ (voir références sur le sujet, e.g. Bodlaender).

Supposons pour simplifier que $k=1$. Le graphe des contraintes est en fait un arbre (et donc toutes les contraintes sont binaires). Si on applique AC et que AC vide un domaine, on peut répondre insatisfait. Sinon on peut répondre satisfait. En effet, on peut fixer une racine x dans l'arbre des contraintes. On peut choisir une valeur dans le domaine D_x. Pour un voisin quelconque y de x, on peut choisir une valeur y compatible avec x (puisque l'instance est cohérente). On peut étendre cet assignement partiel en une solution de cette manière en faisant un parcours « top-down » de l'arbre des contraintes.

Dans le cas général, on peut utiliser une version plus forte de cohérence (Generalized $k+2$-consistency). □

13 Conclusion

Au delà du booléen. L'approche algébrique permet d'obtenir certains résultats pour le cas où le domaine est fini et contient plus que deux valeurs. Pour étendre la conjecture de la dichotomie au delà du booléen, Jeavons et d'autres chercheurs ont proposés de nombreuses classes naturelles qui sont polynomiales. Par exemple, si le langage des contraintes est clos par une opération dite de demi-treillis (une opération binaire, associative, commutative et idempotente) alors GAC est complet (la preuve est la même que celle que nous avons donné pour Horn). Mais dès qu'on a 3 valeurs, le treillis des clones n'est pas dénombrable et on ne dispose pas de sa description. Puisque la partie importante du treillis concerne les clones minimaux, on pourrait espérer décrire cette partie du treillis. Malheureusement, on ne dispose en général que de conditions nécessaires (théorème de Rosenberg). Il faut donc *a priori* une autre approche qui fasse abstraction du treillis pour pouvoir obtenir une classification. C'est ce que Bulatov a fait pour le cas à *3 éléments* (2002) et pour le cas *conservatif* (2006), deux résultats de dichotomie remarquables.

Second théorème de Bulatov : le cas conservatif Ce théorème concerne les langage de contraintes ayant un domaine fini quelconque mais autorisant *toutes les contraintes unaires*. Ceci signifie qu'on peut se restreindre à l'étude de polymorphismes conservatifs, c'est-à-dire dont l'image est forcément une valeur de l'entrée. Ce théorème qui n'est pas simple à énoncer sans introduire plus d'algèbres confirme ce que Feder et Vardi avaient suggéré dans leur papier fondateur, à savoir la pauvreté relative de l'arsenal algorithmique polynomial. Il existe essentiellement *deux algorithmes*, pour résoudre un problème non-uniforme : l'un généralise la cohérence et correspond à l'existence d'un programme Datalog, et l'autre découle du fait qu'on peut dans certain cas représenter de manière compacte l'ensemble des solutions (e.g. utilisation d'une base dans le cas affine).

chronologie de la conjecture.

- 1993 : conjecture de la dichotomie (Feder & Vardi)
- de 1993 à 1998, beaucoup de progrès (Jeavons *et. al.*)
- à partir de 2002, Bulatov revampe l'approche algébrique et démontre deux dichotomies (3 éléments, conservatifs).

- Depuis 7 ans, la communauté s'est étoffée et des preuves mélangeant de manière subtile algèbre, logique et combinatoire apparaissent
  - La conjecture est étendue à d'autres classes de complexité
  - Il y a beaucoup d'efforts concernant l'aspect complexité descriptive
  - La conjecture se réduit à une question assez simple (à expliquer).
  - Une dichotomie pour le cas à 4 éléments est annoncée (2012)
  - *la conjecture reste ouverte...*

Pour démontrer la conjecture. On sait depuis Feder et Vardi qu'il suffit de montrer la dichotomie dans le cas des digraphes (le $\overrightarrow{\mathcal{H}}$-coloring, où $\overrightarrow{\mathcal{H}}$ est un graphe orienté quelconque). Pendant longtemps, il fallait montrer que si Γ avait un polymorphisme d'un certain type mais dont l'arité était arbitraire alors CSP(Γ) serait polynomial. Il y a maintenant une forme plus simple où l'arité est au plus 4. Un *terme de Siggers* est une opération d'arité 4 satisfaisant les identités $f(x,x,x,x)=x$ et $f(y,x,y,z)=f(x,y,z,x)$.

Conjecture 17 (dernière forme connue)**.** *Si* Pol(Γ) *contient un* terme de Siggers *alors* CSP(Γ) *est polynomial.*

Classification pour d'autres questions. L'approche algébrique présentée dans ce cours permet de faciliter la classification de problèmes autres que le problème de décision pour CSP. Dans le cas booléen, le treillis de Post permet de classifier la complexité pour d'autres questions : circomscription, abduction, comptage, énumération, *etc.* Le treillis n'est pas une panacée : il n'aide pas pour MaxSat ou l'approximation. En fait dans ce cas il faut utliser un autre treillis et une autre notion de correspondance.

Il existe aussi des correspondances de Galois qui permettent d'étudier la complexité pour d'autres problèmes : pour QCSP l'extension de CSP où on autorise des variables universelles ; mais aussi pour étudier plus finement la complexité des CSP (les *frozen partial co-clones* permettent d'étudier la constante c dans les algos exponentiels en $O(c^n)$ des cas NP-complets).

Biblio Pour aller plus loin, vous pouvez consulter l'article *De la complexité des problèmes de contraintes* (pdf sur ma page web) ainsi que les références qu'il contient.