Rapport de Stage

Réalisation d'un solveur pour la logique monadique du second ordre sur les structures à largeur arborescente bornée.

Coda Bourotte
coda.bourotte@ens-lyon.fr

Maitre de stage
Radu Iosif
Laboratoire VERIMAG — Grenoble INP

1. Résumé (Abstract)

Beaucoup de problèmes sur des graphes comme la k-colorabilité ou la planarité sont exprimables avec des formules logiques. Depuis la preuve de Courcelle [1] qui montre que la satisfaisabilité de n'importe quelle formule MSO est décidable sur certaines classes de modèles, on note une absence d'implémentation d'algorithmes résolvant ce problème. Ceci est causé principalement par une complexité FPT hyper-exponentielle. Dans ce rapport on implémentera une réduction du problème SAT sur les modèles de tree-width bornée (dont les graphes de tree-width bornée font parti) vers des problèmes sur les arbres pour ensuite envoyer la formule obtenue au logiciel Mona [2] conçu pour résoudre ce problème sur les arbres.

Table des matières

2. Les F-algèbres

Les définitions de cette partie sont principalement tirées de [3]. Le but de cette partie est de présenter le cadre mathématique dans lequel on évolue : on définira les notions importantes de F-algèbres (algèbres de termes) et d'automates de termes.

2.1. Définitions générales

On notera $\mathbb{N}$ l'ensemble des entiers naturels et $\mathbb{N}^* = \mathbb{N} \setminus \{0\}$. On notera $[n] \coloneqq \{1, ..., n\}$. Un *multi-graphe* est un graphe pouvant avoir plusieurs fois la même arête. On notera $\mathbb{G}^u$ l'ensemble des graphes non-orientés et $\mathbb{J}^d$ l'ensemble des multi-graphes orientés. Une *clique de taille* n est un graphe à n sommets où ils sont tous reliés.

Un *alphabet* Σ est un ensemble fini de symboles. Un *alphabet classé* est un couple $F = \langle \Sigma, \# \rangle$ avec Σ un alphabet et $\# : \Sigma \longrightarrow \mathbb{N}$ l'application qui à chaque symbole associe son *arité*. L'arité représente le nombre d'arguments d'un objet. On notera $f^{\#n}$ pour dénoter le symbole f d'arité $\#(f) = n$. Un objet d'arité 0 est appelé une *constante*.

On note par $\mathcal{T}(F)$ l'ensemble des *termes* sur $F = \langle \Sigma, \# \rangle$ qui sont les arbres étiquetés par Σ tels que pour tout $\alpha \in \Sigma$, les nœuds étiquetés par α ont $\#(\alpha)$ enfants. On notera un nœud α d'enfants $x_1, ..., x_{\#(\alpha)}$ par $\alpha\big(x_1, ..., x_{\#(\alpha)}\big)$. Si $\alpha(x, y)$ est d'arité 2, on pourra noter dans certains cas cela par $x\alpha y$. Un *langage* est un sous-ensemble de $\mathcal{T}(F)$. Une *interprétation* de $\mathcal{T}(F)$ (aussi nommé une F-*algèbre*) est un couple $\mathbb{M} = \langle M, \rho \rangle$ avec M un ensemble tel que pour tout $x \in F$, $\rho(x, \cdot) : M^{\#(x)} \longrightarrow M$. Si $\#(x) = 0$, on note $\rho(x) = \rho(x, ())$. On notera $\rho_{\mathbb{M}}$ le ρ de la F-algèbre $\mathbb{M} = \langle M, \rho \rangle$.

Une F-algèbre $\mathbb{M} = \langle M, \rho_M \rangle$ est dite *finie* si $|M| < +\infty$. On identifiera souvent $\mathbb{M}$ et M, par exemple en écrivant $x \in \mathbb{M}$ pour dire $x \in M$.

Exemple des mots Pour $F_{a,b} = \{a^{\#0}, b^{\#0}, \times^{\#2}\}$, une interprétation possible est la $F_{a,b}$-algèbre $\mathbb{M}_{a,b} = \langle \{a, b\}^*, \rho \rangle$ des mots sur $\Sigma = \{a, b\}$ munit de la concaténation définie par ρ:

$$\rho\Big(\times_{F_{a,b}}, (x, y)\Big) = xy \qquad \rho\Big(a_{F_{a,b}}\Big) = a \qquad \rho\Big(b_{F_{a,b}}\Big) = b$$

Interprétation Canonique Il est bon de remarquer que tout alphabet classé F possède une *interprétation canonique* $\mathbb{C}_F = \langle \mathcal{T}(F), \rho^{\text{id}} \rangle$, avec pour tout $x \in F$, $\rho^{\text{id}}\Big(x, \big(a_1, ..., a_{\#(x)}\big)\Big) = x\big(a_1, ..., a_{\#(x)}\big)$. On remarque que l'interprétation de $F_{a,b}$ dans $\mathbb{M}_{a,b}$ possède la propriété que $\times_{F_{a,b}}$ se comporte comme une opération associative, tandis qu'elle ne l'est pas dans $\mathbb{C}_{F_{a,b}}$: ainsi, deux interprétations distinctes auront des propriétés fondamentalement différentes.

Morphisme Soit $\mathbb{A}, \mathbb{B} = \langle A, \rho_A \rangle, \langle B, \rho_B \rangle$ deux F-algèbres, on dit que $\varphi : A \longrightarrow B$ est un morphisme d'algèbres si pour tout $\alpha\big(x_1, ..., x_{\#(\alpha)}\big) \in \mathcal{T}(F)$ on a

$$\varphi\Big(\rho_A\big(\alpha, \big(x_1, ..., x_{\#(\alpha)}\big)\big)\Big) = \rho_B\Big(\alpha, \big(\varphi(x_1), ..., \varphi\big(x_{\#(\alpha)}\big)\big)\Big)$$

Reconnaissabilité On dit que $E \subseteq \mathbb{M}$ une F-algèbre est *reconnaissable* s'il existe un morphisme de F-algèbre $\varphi : \mathbb{M} \to \langle A, \rho_A \rangle$ avec $\langle A, \rho_A \rangle$ une F-algèbre finie et $X \subseteq A$ tel que $E = \varphi^{-1}(X)$.

Exemple $\mathbb{G}^c$ des co-graphes On considère l'alphabet classé $G^c \coloneqq \{\otimes^{\#2}, \oplus^{\#2}, \mathbf{1}^{\#0}\}$ et l'interprétation $\mathbb{G}^c$ dans l'ensemble $\mathbb{G}^u$ des graphes non orientés telle que $\mathbf{1}$ représente un graphe à un sommet, $G \oplus H$ représente l'union disjointe des deux graphes et $G \otimes H$ l'union disjointe des deux graphes où l'on rajoute toutes les arêtes entre G et H. Ainsi $\rho_{\mathbb{G}^c}((((\mathbf{1} \oplus \mathbf{1}) \oplus \mathbf{1}) \otimes (\mathbf{1} \otimes (\mathbf{1} \otimes \mathbf{1}))))$ donne le graphe avec trois sommets disjoints x, y, z, tous reliés à chaque sommet d'une clique de taille 3 a, b, c (Fig. 1).

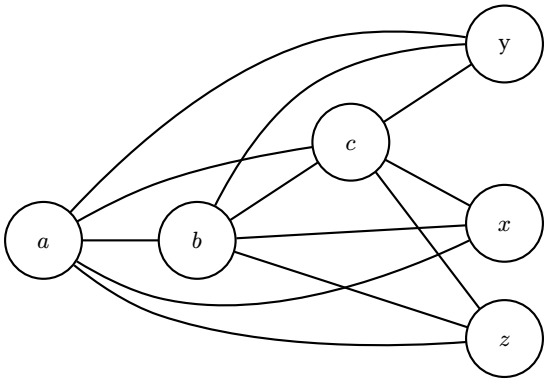


Fig. 1. – Le co-graphe donné par $\rho_{\mathbb{G}^c}((((\mathbf{1} \oplus \mathbf{1}) \oplus \mathbf{1}) \otimes (\mathbf{1} \otimes (\mathbf{1} \otimes \mathbf{1}))))$

Exemple $\mathbb{J}^p$ des graphes en série parallèle On considère l'alphabet classé $J^p := \{/\!\!/^{\#2}, \bullet^{\#2}, e\}$ et l'interprétation $\mathbb{J}^p$ dans l'ensemble des multigraphes orientés possédant un état source et un état puits tel que e soit un graphe à une arête, $G /\!\!/ H$ correspond à identifier la source de G et de H et identifier le puits de G et de H et $G \bullet H$ correspond à identifier le puits de G avec la source de H. Ainsi $\rho_{\mathbb{J}^p}(((e /\!\!/ e) \bullet e) /\!\!/ e)$ est représenté (Fig. 2)

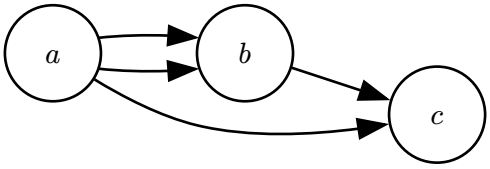


Fig. 2. – Le graphe donné par $\rho_{\mathbb{J}^p}(((e /\!\!/ e) \bullet e) /\!\!/ e)$
Après les opérations le sommet **a** sera la source et **c** le puits.

2.2. Automates de termes

Un *automate de termes* sur F est un quadruplet $A = \langle F, Q, Q_f, \delta \rangle$ avec F un alphabet classé, Q un ensemble fini d'états, $Q_f \subseteq Q$ l'ensemble des états finaux et δ la fonction de transition, telle que pour tout $x \in \Sigma$, $\delta(x, \cdot) : Q^{\#(x)} \to Q$. On défini inductivement l'évaluation $\delta^* : \mathcal{T}(F) \to Q$ d'un nœud $\alpha\left(x_1, ..., x_{\#(\alpha)}\right)$ par :

$$\delta^*\left(\alpha\left(x_1, ..., x_{\#(\alpha)}\right)\right) = \delta\left(\alpha, \left(\delta^*(x_1), ..., \delta^*\left(x_{\#(\alpha)}\right)\right)\right)$$

On a donc que $\delta^*(a) = \delta(a, ())$ si $\#(a) = 0$. Un terme $t \in \mathcal{T}(F)$ est *accepté* par A si $\delta^*(t) \in Q_f$. Le langage *reconnu par* A est le langage des termes accepté par A et est dénoté par $L(A)$

Exemple Le langage L des termes sur $F = \{f^{\#2}, a^{\#0}\}$ qui sont acceptés par $A = \langle F, \{0, 1\}, \{0\}, \delta \rangle$ avec δ telle que:

$$\delta(a, ()) = \delta(f, (0, 1)) = \delta(f, (1, 0)) = 1 \qquad \delta(f, (0, 0)) = \delta(f, (1, 1)) = 0$$

est exactement le langage des arbres ayant un nombre pair de feuilles.

Proposition La classe des langages reconnus par un automate d'arbre sur F est stable par union, complémentaire, intersection.

Beaucoup d'autres propositions qui sont vraies pour les automates de mots se transposent bien en proposition vraies sur les automates de termes. Par exemple, un automate non déterministe de termes (i.e. où pour tout $x \in F$, on a $\delta(x, \cdot) : Q^{\#(x)} \longrightarrow \mathcal{P}(Q)$) peut être déterminisé. Les automates existent aussi en version top-down (l'équivalent du miroir sur les mots), mais fournissent dans ce cas une classe de langage équivalente que s'ils sont non déterministes.

Théorème: Un langage L sur $\mathcal{T}(F)$ est reconnaissable ssi il est reconnaissable par un automate de terme.

$\boxed{\Rightarrow}$ Soit $\mathbb{A}$ une F-algèbre finie, soit $L \subseteq \mathcal{T}(F)$ et soit $h : \mathbb{C}_F \longrightarrow \mathbb{A}$ un morphisme et $X \subseteq \mathbb{A}$ tel que $L = h^{-1}(X)$. Alors on pose l'automate $A = \langle F, \mathbb{A}, X, \delta \rangle$ avec δ telle que $\delta\Big(\alpha, \big(q_1, ..., q_{\#(\alpha)}\big)\Big) = \rho_{\mathbb{A}}\Big(\alpha, \big(q_1, ..., q_{\#(\alpha)}\big)\Big)$. Comme $h : \mathcal{T}(F) \to \mathbb{A}$ est un morphisme on a que, soit $\alpha(x_1, ..., x_n) \in \mathcal{T}(F)$ avec $n = \#(\alpha)$:

$$h(\alpha(x_1, ..., x_n)) = h\Big(\rho_{\mathbb{C}_{\mathbb{A}}}(\alpha, (x_1, ..., x_n)) = \rho_{\mathbb{A}}(\alpha, (h(x_1), ..., h(x_n)))\Big) = \delta(\alpha, (h(x_1), ..., h(x_n)))$$

On remarque alors que h coïncide avec la définition de δ^*, on a donc $L(A) = \delta^{*^{-1}}\big(Q_f\big) = h^{-1}(X)$

$\boxed{\Leftarrow}$ Soit $A = \langle F, Q, Q_f, \delta \rangle$ un automate, on pose $\mathbb{A} = (Q, \delta)$. On a que le langage de l'automate $L(A) = \delta^{*^{-1}}\big(Q_f\big) \subseteq \mathcal{T}(F)$, et on alors $\delta^* : \mathcal{T}(F) \to Q$ est bien un morphisme de F-algèbre.

3. Logique Monadique du Second Ordre (MSO)

3.1. Définitions

Soit $\mathrm{Var} := \{x, y, z, x_1, ...\}$ un ensemble infini dénombrable de *variables de termes*, toutes d'arité 0. Soit $\widetilde{\mathrm{Var}} = \{X, Y, Z, X_1, ...\}$ un alphabet classé infini dénombrable de *variables relationnelles*, toutes d'arité 1. On notera toujours par une majuscule les variables relationnelles. Une *logique* est un couple $\langle F, R \rangle$ avec F un alphabet classé de *termes* et R un alphabet classé de *relations*.

Logique du Premier Ordre L'ensemble des *formules du premier ordre* noté $\mathcal{L}_1(F, R)$ avec $\langle F, R \rangle$ une logique est défini inductivement par
- "$r\big(x_1, ..., x_{\#(r)}\big)$" $\in \mathcal{L}_1(F, R)$ pour $r \in R$ et $x_1, ..., x_{\#(r)} \in \mathcal{T}(F \cup \mathrm{Var})$
- "$t_1 = t_2$" $\in \mathcal{L}_1(F, R)$ pour $t_1, t_2 \in \mathcal{T}(F \cup \mathrm{Var})$
- "$\varphi \vee \psi$", "$\neg\varphi$", "$\varphi \wedge \psi$", "$\varphi \to \psi$", "$\varphi \leftrightarrow \psi$" $\in \mathcal{L}_1(F, R)$ pour $\varphi, \psi \in \mathcal{L}_1(F, R)$
- "$\forall x.\varphi$", "$\exists x.\varphi$" $\in \mathcal{L}_1(F, R)$ pour $\varphi \in \mathcal{L}_1(F, R)$ et $x \in \mathrm{Var} \setminus F$

Logique Monadique du Second Ordre (MSO) L'ensemble des *formules MSO (Monadic Second-order)* noté $\mathcal{L}_{\mathrm{mso}}(F, R)$ avec $\langle F, R \rangle$ une logique est défini inductivement par
- "$r\big(x_1, ..., x_{\#(r)}\big)$" $\in \mathcal{L}_{\mathrm{mso}}(F, R)$ pour $r \in R \cup \widetilde{\mathrm{Var}}$ et $x_1, ..., x_{\#(r)} \in \mathcal{T}(F \cup \mathrm{Var})$
- "$t_1 = t_2$" $\in \mathcal{L}_{\mathrm{mso}}(F, R)$ pour $t_1, t_2 \in \mathcal{T}(F \cup \mathrm{Var})$
- "$\varphi \vee \psi$", "$\neg\varphi$", "$\varphi \wedge \psi$", "$\varphi \to \psi$", "$\varphi \leftrightarrow \psi$" $\in \mathcal{L}_{\mathrm{mso}}(F, R)$ pour $\varphi, \psi \in \mathcal{L}_{\mathrm{mso}}(F, R)$
- "$\forall x.\varphi$", "$\exists x.\varphi$" $\in \mathcal{L}_{\mathrm{mso}}(F, R)$ pour $\varphi \in \mathcal{L}_{\mathrm{mso}}(F, R)$ et $x \in \mathrm{Var} \setminus F$
- "$\forall X.\varphi$", "$\exists X.\varphi$" $\in \mathcal{L}_{\mathrm{mso}}(F, R)$ pour tout $X \in \widetilde{\mathrm{Var}} \setminus R$ et $\varphi \in \mathcal{L}_{\mathrm{mso}}(F, R)$

On remarque que l'on ne quantifie que sur des relations d'arité 1. De tels objets peuvent être représenté par un ensemble, précisément l'ensemble des éléments du modèle qui satisfait la propriété. C'est pourquoi on écrira "$y \in X$" pour désigner "$X(y)$" pour X une relation d'arité 1.

On désignera par $x \notin X$ la formule $\neg(x \in X)$ et par $a \neq b$ la formule $\neg(a = b)$.

Variables Libres On a donc $\mathcal{L}_1(F, R) \subsetneq \mathcal{L}_{\mathrm{mso}}(F, R)$. Pour tout $t \in \mathcal{T}(F \cup V)$ on défini $\mathrm{FV}(t)$ comme l'ensemble des nœuds étiqueté par V dans l'arbre t. On défini $\mathrm{FV}(\varphi)$ et $\widetilde{\mathrm{FV}}(\varphi)$ pour $\varphi \in \mathcal{L}_{\mathrm{mso}}(F, R)$ (et donc par extension sur $\mathcal{L}_1(F, R)$) comme étant l'ensemble des variables libres de respectivement termes et relations :
- $\mathrm{FV}(\neg\varphi) = \mathrm{FV}(\varphi)$ et $\widetilde{\mathrm{FV}}(\neg\varphi) = \widetilde{\mathrm{FV}}(\varphi)$
- $\mathrm{FV}(\varphi \vee \psi) = \mathrm{FV}(\varphi \wedge \psi) = \mathrm{FV}(\varphi \to \psi) = \mathrm{FV}(\varphi \leftrightarrow \psi) = \mathrm{FV}(\varphi) \cup \mathrm{FV}(\psi)$, pareil pour $\widetilde{\mathrm{FV}}$
- $\mathrm{FV}\Big(r\big(x_1, ..., x_{\#(r)}\big)\Big) = \mathrm{FV}(x_1) \cup ... \cup \mathrm{FV}\big(x_{\#(r)}\big)$
- $\widetilde{\mathrm{FV}}\Big(r\big(x_1, ..., x_{\#(r)}\big)\Big) = \{r\}$ si $r \in \widetilde{\mathrm{Var}}$ et $\widetilde{\mathrm{FV}}\Big(r\big(x_1, ..., x_{\#(r)}\big)\Big) = \emptyset$ sinon

- $\mathrm{FV}(t_1 = t_2) = \mathrm{FV}(t_1) \cup \mathrm{FV}(t_2)$ et $\widetilde{\mathrm{FV}}(t_1 = t_2) = \emptyset$
- $\mathrm{FV}(\forall x.\varphi) = \mathrm{FV}(\exists x.\varphi) = \mathrm{FV}(\varphi) \setminus \{x\}$ et $\widetilde{\mathrm{FV}}(\forall x.\varphi) = \widetilde{\mathrm{FV}}(\exists x.\varphi) = \widetilde{\mathrm{FV}}(\varphi)$
- $\mathrm{FV}(\forall X.\varphi) = \mathrm{FV}(\exists X.\varphi) = \mathrm{FV}(\varphi)$ et $\widetilde{\mathrm{FV}}(\forall X.\varphi) = \widetilde{\mathrm{FV}}(\exists X.\varphi) = \widetilde{\mathrm{FV}}(\varphi) \setminus \{X\}$

Une formule logique φ est dite *close* si $\mathrm{FV}(\varphi) \cup \widetilde{\mathrm{FV}}(\varphi) = \emptyset$.

Modèles Un *modèle sur* $\langle F, R\rangle$ pour φ une formule est un triplet $\langle \mathbb{M}, \rho_V, \rho_R\rangle$ telle que $\langle \mathbb{M}, \rho_V\rangle$ soit une F-algèbre et telle que l'on ait $\forall r \in R \cup \widetilde{\mathrm{FV}}(\varphi), \rho_R(r, \cdot) : \mathbb{M}^{\#(r)} \longrightarrow \{\top, \bot\}$ et $\forall f \in F \cup \mathrm{FV}(\varphi), \rho_F : \mathbb{M}^{\#(f)} \longrightarrow \mathbb{M}$. Pour $m = \langle \mathbb{M}, \rho_F, \rho_R\rangle$ un modèle, $a \in \mathbb{M}$, $v \in \mathrm{Var}$, $S \subseteq \mathbb{M}^n$, $V \in \widetilde{\mathrm{Var}}$, on pose $m[v \leftarrow a] = \langle \mathbb{M}, \rho'_F, \rho_R\rangle$ et $m[V \leftarrow S] = \langle \mathbb{M}, \rho_F, \rho'_R\rangle$ avec

$$\rho'_F(x) = \begin{cases} \rho_F(x) & \text{si } x \neq v \\ a & \text{sinon} \end{cases} \qquad \rho'_R(X, (x_1, ..., x_n)) = \begin{cases} \rho_R(X, (x_1, ..., x_n)) & \text{si } X \neq V \\ \top & \text{si } (x_1, ..., x_n) \in S \wedge X = V \\ \bot & \text{sinon} \end{cases}$$

Intuitivement, $m[v \leftarrow a]$ est le même modèle que m dans lequel la variable v vaux maintenant a. Pareillement, $M[V \leftarrow S]$ est le modèle ou la variable / relation du second ordre V ne vaux $\top$ que sur S.

Satisfaction On défini par induction que $m = \langle \mathbb{M}, \rho_V, \rho_R\rangle$ satisfait $\varphi \in \mathcal{L}_{\mathrm{mso}}(F, R)$ (noté $m \vDash \varphi$) par
- $m \vDash \varphi \wedge \psi$ si $m \vDash \varphi$ et $m \vDash \psi$.
  On fait similairement pour $\vee, \neg, \rightarrow, \leftrightarrow$ avec respectivement ou, non, implique, équivaut.
- $m \vDash \exists x.\varphi$ si il existe $e \in \mathbb{M}$ tel que $m[x \leftarrow e] \vDash \varphi$.
- $m \vDash \forall x.\varphi$ si pour tout $e \in \mathbb{M}$ on a $m[x \leftarrow e] \vDash \varphi$.
- $m \vDash \exists X.\varphi$ si il existe $S \subseteq \mathbb{M}$ tel que $m[X \leftarrow S] \vDash \varphi$
- $m \vDash \forall X.\varphi$ si pour tout $S \subseteq \mathbb{M}$ on a $m[X \leftarrow S] \vDash \varphi$
- $m \vDash r\big(x_1, ..., x_{\#(r)}\big)$ si $\rho_R\Big(r, \big(\rho_V(x_1), ..., \rho_V\big(x_{\#(r)}\big)\big)\Big) = \top$
- $m \vDash t_1 = t_2$ si $\rho_V(t_1) = \rho_V(t_2)$

Un modèle pour une logique est un modèle pour toutes les formules closes de la logique. L'ensemble des modèles sur $\langle F, R\rangle$ une logique est noté $\mathcal{M}(F, R)$. En général, on aura une manière de transformer des objets bien connus E (des mots, des arbres, des graphes) en modèles pour une certaine logique par une application $\lfloor\cdot\rfloor : E \longrightarrow \mathcal{M}(F, R)$. Dans ce cas on dit que m *vérifie* φ si $\lfloor m\rfloor \vDash \varphi$.

On dit que $P \in \mathcal{L}_{\mathrm{mso}}(F, R)$ est une *tautologie* si $\forall m \in \mathcal{M}(F, R), m \vDash P$. Si $\forall m \in \mathcal{M}(F, R), m \nvDash P$, on dit que P est une *antilogie*.

3.2. MSO WS1S (sur les mots)

Définition Soit Σ un alphabet, on considère *la logique* ***WS1S*** *sur* Σ (***W****eak* ***S****econd-order with* ***1*** ***S****uccessor*) définie par $\mathcal{L}_{\mathrm{WS1S}} \coloneqq \langle F_\Sigma, R_\Sigma\rangle$ avec $F_\Sigma = \{\mathrm{succ}^{\#1}\}$ et $R_\Sigma = \{p_\alpha^{\#1} : \alpha \in \Sigma\}$. L'idée est que $p_\alpha(x)$ indique qu'il y a la lettre α à la position x, et l'on quantifie sur les positions. On pose $\lfloor\cdot\rfloor : \Sigma^* \longrightarrow \mathcal{M}(F_\Sigma, R_\Sigma)$ tel que pour tout $w \in \Sigma^*$ on ai $\lfloor w\rfloor = \langle\{1, ..., |w|\}, \rho_V, \rho_R\rangle$ avec

$$\rho_V(\mathrm{succ}, x) = \begin{cases} x + 1 & \text{si } x < |w| \\ |w| & \text{sinon} \end{cases} \qquad \text{et} \qquad \rho_R(p_\alpha, x) = \begin{cases} \top & \text{si } \ w_x = \alpha \\ \bot & \text{sinon} \end{cases}$$

Pour $\varphi \in \mathcal{L}_{\mathrm{WS1S}}$ on écrit $L(\varphi) = \{w \in \Sigma \mid \lfloor w\rfloor \vDash \varphi\} \subseteq \Sigma^*$ le langage *reconnu* par une formule.

Exemples sur $\Sigma = \{a, b\}$:
- On a $\lfloor\mathrm{abbab}\rfloor \vDash \exists x.p_{\mathrm{b}}(x)$ car il y a une position $x \in \{1, ..., |w|\}$ telle que $\mathrm{abbab}_x = \mathrm{b}$, mais $\lfloor\mathrm{aaa}\rfloor \nvDash \exists x.p_{\mathrm{b}}(x)$.
- La formule $\forall x, x \neq x$ n'est vérifié que par le mot vide.

- On pose $\text{first}(x) \coloneqq$ "$\forall z, \text{succ}(z) \neq x$" une formule qui indique que $\rho_V(x) = 1$. On a alors en posant $\varphi \coloneqq$ "$\exists x, \text{first}(x) \wedge p_{\text{a}}(x)$" le langage $L(\varphi)$ est l'ensemble des mots commençant par un a. Similairement on pose $\text{last}(z) \coloneqq$ "$z = \text{succ}(z)$"
- La formule $\exists X, (\exists z, \text{first}(z) \wedge z \in X) \wedge (\forall z, z \in X \leftrightarrow \text{succ}(z) \notin X \vee \text{last}(z))$ reconnaît tous les mots de longueur impaire. Ici, le X est l'ensemble des positions impaires ($1 \in X$, et la deuxième condition impose l'alternance)
- On peut définir la relation $\leq$: On pose $\text{le}(x, y) \coloneqq$ "$\forall X, (x \in X \wedge (\forall z \in X, \text{succ}(z) \in X)) \rightarrow y \in X$"

Décidabilité Un langage L est régulier ssi il est reconnu par une formule MSO. [4]

La preuve consiste à encoder chaque automate déterministe complet $\mathcal{A}$ une formule $\varphi_{\mathcal{A}}$ qui reconnaît le même langage de la forme $\exists Y_{q_1}, ..., \exists Y_{q_n}, \psi$ avec $q_1, ..., q_n$ les états de l'automate et ψ qui donne des conditions de transition et de terminaison / début. La réciproque se fait par induction sur les formules sur un alphabet bien choisi permettant d'encoder des ensembles de positions.

3.3. MSO sur les arbres: WS2S et WSkS

Définition Soit Σ un alphabet, en notant $B(\Sigma)$ l'ensemble des arbres binaires ordonnés étiquetés par Σ (les feuilles n'ont pas d'étiquette et sont représentées par ε), on considère *la logique **WS2S** sur* Σ (***W**eak **S**econd-order with **2** **S**uccessor*) définie par $\mathcal{L}_{\text{WS2S}} \coloneqq \langle F_{B(\Sigma)}, R_{B(\Sigma)} \rangle$ avec $F_{B(\Sigma)} = \left\{ \text{s}_1^{\#1}, \text{s}_2^{\#1} \right\}$ et $R_{B(\Sigma)} = \{ p_\alpha^{\#1} : \alpha \in \Sigma \}$. Ici l'opération $\text{s}_1(t)$ correspond à prendre le fils gauche de t et l'opération $\text{s}_2(t)$ correspond à prendre le fils droit. On pose $\lfloor \cdot \rfloor : B(\Sigma) \longrightarrow \mathcal{M}\left(F_{B(\Sigma)}, R_{B(\Sigma)}\right)$ tel que pour tout $t \in B(\Sigma)$ on ai $\lfloor t \rfloor = \langle N(t), \rho_V, \rho_R \rangle$ avec $N(t)$ les nœuds de t et

$$\begin{cases} \rho_V(\text{s}_1, \alpha(t_1, t_2)) = t_1 \\ \rho_V(\text{s}_1, x) = x \text{ sinon} \end{cases} \quad \begin{cases} \rho_V(\text{s}_2, \alpha(t_1, t_2)) = t_2 \\ \rho_V(\text{s}_2, x) = x \text{ sinon} \end{cases} \quad \begin{cases} \rho_R(p_\alpha, \alpha(t_1, t_2)) = \top \\ \rho_R(p_\alpha, \varepsilon) = \bot \text{ sinon} \end{cases}$$

Pour $\varphi \in \mathcal{L}_{\text{WS2S}}$ on écrit $L(\varphi) = \{ t \in B(\Sigma) \mid \lfloor t \rfloor \vDash \varphi \} \subseteq B(\Sigma)^*$ le langage *reconnu* par une formule.

Décidabilité Un langage est reconnu par automates d'arbres sur $\{ \alpha^{\#2} : \alpha \in \Sigma \} \cup \{ \varepsilon^{\#0} \}$ ssi il est reconnus par une formule de $\mathcal{L}_{\text{WS2S}}$.

MSO WSkS La logique $\mathcal{L}_{\text{WSkS}}$ est celle ou l'on a k fonctions de successeurs $s_1^{\#1}, ..., s_k^{\#1}$, et permet de reconnaître des langages sur les arbres fini ordonné à branchement égal à k. Mais on peut facilement la réduire à la logique sur **WS2S** en considérant le morphisme d'algèbre d'une algèbre $F = \left\{ \alpha_i^{\#k} : i < k : \alpha \in \Sigma \right\} \cup \{ \varepsilon^{\#0} \}$ dans l'algèbre $B(\Sigma \cup \{\bullet\})$ par $\varphi : F \longrightarrow B(\Sigma \cup \{\bullet\})$ défini par induction par

$$\varphi(\alpha_i(x_1, ..., x_k)) = \alpha(\bullet(\varphi(x_1), \bullet(\varphi(x_2), ..., \bullet(\varphi(x_{k-1}), \varphi(x_k))))) \quad \text{et} \quad \varphi(\varepsilon) = \varepsilon$$

Comme la hauteur entre deux symboles de Σ dans $\varphi(t)$ est bornée par k, les langages reconnus sont les mêmes modulo le morphisme d'algèbre : l'automate peut retenir les k derniers états dans lequel il était.

3.4. MSO sur les graphes

MSO sur les sommets Soit Σ un alphabet. On pose $R_e = \{ \text{edg}^{\#2} \} \cup \{ p_\alpha^{\#1} : \alpha \in \Sigma \}$ et on considère alors la logique $\mathcal{L}_{\text{MSOS}} \coloneqq \langle \emptyset, R_e \rangle$. On pose $\lfloor \cdot \rfloor : \mathbb{G}^u \longrightarrow \mathcal{M}(\emptyset, R_e)$ tel que pour tout graphe $\mathcal{G} \in \mathbb{G}^u$ avec V comme ensemble de sommets on ai $\lfloor g \rfloor = \langle V, \rho_V, \rho_R \rangle$ tel que pour tout $x, y \in V$, on ai $\rho_R(\text{edg}, (x, y)) = \top$ si et seulement s'il existe une arrête entre x et y. La quantification se fait donc sur des sommets et des ensembles de sommets.

On écrit $L(\varphi) = \{ t \in \mathbb{G}^u \mid \lfloor t \rfloor \vDash \varphi \} \subseteq \mathbb{G}^u$ le langage *reconnu* par une formule. Par exemple, on a que $L(\forall x. \forall y.\ \text{edg}(x, y))$ est l'ensemble des graphes complets.

MSOA La logique $\mathcal{L}_{\text{MSOA}}$ (appelée par la suite logique sur les arbres) est la restriction de la MSO sur les sommets pour les graphes qui sont des arbres. Ainsi $L_{\text{MSOA}}(\varphi) = L(\varphi) \cap \mathcal{T}(\Sigma)$. Elle est à différencier de la logique **WSkS**, car dans la logique **MSOA** le degré des sommets n'est pas fixé ni borné et l'ordre des fils n'a pas d'importance.

Proposition Le problème de savoir si $L(\varphi) = \emptyset$ est indécidable, même pour $\varphi \in \mathcal{L}_1(\emptyset, R_e)$ (premier ordre) [5].

Il est à noter que d'autres MSO sur les graphes existent, notamment la **MSO sur les arêtes** pour les multigraphes orienté :

MSO sur les arêtes On pose $R'_e = \{\text{edg}'^{\#3}, \text{somm}'^{\#1}\}$ et on considère alors la logique $\langle \emptyset, R'_e \rangle$. On pose $\lfloor \cdot \rfloor : \mathbb{J}^d \longrightarrow \mathcal{M}(\emptyset, R'_e)$ tel que pour tous $g \in \mathbb{J}^d$ avec V comme ensemble de sommets et E l'ensemble de toutes les arêtes avec multiplicité on ai $\lfloor g \rfloor = \langle V \cup E, \rho_V, \rho_R \rangle$ tel que

$$\rho_R(\text{edg}', (x, y, e)) = \top \text{ ssi } e \text{ est l'arête entre } x \text{ et } y$$

et

$$\rho_R(\text{somm}', (x)) = \top \Leftrightarrow x \in V$$

Ici la quantification se fait sur des sommets et arrêtes, ou ensembles contenant sommets et arrêtes. La manière de distinguer les deux est d'utiliser la relation somm'.

La classe des langages reconnus par la **MSO sur les arêtes** est strictement plus grande que la **MSO sur les sommets**. Par exemple, l'existence d'un cycle eulérien est une formule exprimable avec la MSO sur les arêtes mais pas avec la MSO sur les sommets.

MSO sur les structures Soit R un alphabet classé, on considère alors la logique $\mathcal{L}_{\text{MSO}}(R) \coloneqq \langle \emptyset, R \rangle$. On pose $\lfloor \cdot \rfloor : X \longrightarrow \mathcal{M}(\emptyset, R)$ tel que pour tout $x \in X$, on ai $\lfloor x \rfloor = \langle M, \rho_\emptyset, \rho_R \rangle$ avec M fini.

On remarque que la MSO sur les structures permet d'encoder les 2 MSO de graphes vu prédédemment (que ce soit sur les sommets ou sur les arêtes). C'est pour cela (et pour éviter de vouloir faire des cas particuliers) que l'on s'intéressera à résoudre cette MSO pour les structures de logique bornée.

Afin de différencier les différentes applications $\lfloor \cdot \rfloor$, on les notera $\lfloor \cdot \rfloor_{\text{WS1S}}, \lfloor \cdot \rfloor_{\text{WS2S}}, \lfloor \cdot \rfloor_{\text{MSOA}}, \cdots$

4. Tree-Widths

4.1. Algèbres de graphes

Soit $\mathcal{A}$ un ensemble infini dénombrable de *couleurs* ou *labels*. On identifiera $\mathcal{A}$ avec $\mathbb{N}^*$

L'algèbre F_k^{VR} des graphes de largeur de clique bornée
On pose

$$F^{\text{VR}} \coloneqq \left\{\text{add}^{\#1}_{(a,b)}, \text{relab}^{\#1}_{a\to b} : (a,b) \in \mathcal{A}^2\right\} \cup \left\{\mathbf{1}^{\#0}, \mathbf{1}_l^{\#0}, \oplus^{\#2}\right\}$$

Soit $k \in \mathbb{N}^*$, on pose

$$F_k^{\text{VR}} = \left\{\text{add}^{\#1}_{(a,b)}, \text{relab}^{\#1}_{a\to b} : 0 < a, b \leq k\right\} \cup \left\{\mathbf{1}^{\#0}, \mathbf{1}_l^{\#0}, \oplus^{\#2}\right\}$$

On a donc $F_k^{\text{VR}} \subsetneq F^{\text{VR}}$. On définit $\mathcal{G}_u = (\mathcal{GP}, \delta)$ une interprétation de F^{VR} où $\mathcal{GP}$ est l'ensemble des graphes non dirigés munis d'un étiquetage de certains sommets par un $a \in \mathcal{A}$, et δ telle que

$\delta\big(\mathrm{add}_{(a,b)}, (G)\big) =$ G avec tout les noeuds labelisé par a réliée à tout les noeuds labelisé par b

$\delta(\mathrm{relab}_{a\to b}, (G)) =$ G où les labels a sont renommé en b

$\delta(\oplus, (G, G')) =$ $G \uplus G'$ l'union disjointe

$\delta(\mathbf{1}) =$ le graphe à 1 sommet de label 1

$\delta(\mathbf{1}_l) =$ le graphe à 1 sommet de label 1 avec une boucle

On pose alors $\mathbb{F}_k^{\mathrm{VR}} = \big\{\delta(t) : t \in \mathcal{T}\big(F_k^{\mathrm{VR}}\big)\big\}$ et $\mathbb{F}^{\mathrm{VR}} = \big\{\delta(t) : t \in \mathcal{T}\big(F^{\mathrm{VR}}\big)\big\}$.

On remarque que l'algèbre F^{VR} est une généralisation de l'algèbre des co-graphes et que $\mathbb{G}^c \subseteq \mathbb{F}_2^{\mathrm{VR}}$ par l'encodage des opérations :

$$\begin{aligned} G_1 \oplus_{G^c} G_2 &\longrightarrow \mathrm{relab}_{2\to1}(G_1 \oplus \mathrm{relab}_{1\to2}(G_2)) \\ G_1 \otimes_{G^c} G_2 &\longrightarrow \mathrm{relab}_{2\to1}\Big(\mathrm{add}_{(1,2)}\Big(\mathrm{add}_{(2,1)}(G_1 \oplus \mathrm{relab}_{1\to2}(G_2))\Big)\Big) \\ \mathbf{1}_{G^c} &\longrightarrow \mathbf{1} \end{aligned}$$

Le nom viens du fait que soit G un graphe, on a $G \in \mathbb{F}_k^{\mathrm{VR}}$ si et seulement si la clique de taille k n'est pas un mineur de G.

L'algèbre F^{HR} pour les graphes de Tree-Width bornée

On pose

$$F^{\mathrm{HR}} := \Big\{\mathrm{del}_a^{\#1}, \mathrm{relab}_{a\to b}^{\#1} : (a,b) \in \mathcal{A}^2\Big\} \cup \big\{\mathrm{compose}^{\#2}, \mathbf{1}^{\#0}, \boldsymbol{e}^{\#0}\big\}$$

Soit $k \in \mathbb{N}^*$, on pose

$$F_k^{\mathrm{HR}} := \Big\{\mathrm{del}_a^{\#1}, \mathrm{relab}_{a\to b}^{\#1} : 0 < a, b \leq k\Big\} \cup \big\{\mathrm{compose}^{\#2}, \mathbf{1}^{\#0}, \boldsymbol{e}^{\#0}\big\}$$

On a donc $F_k^{\mathrm{HR}} \subsetneq F^{\mathrm{HR}}$. On définit $\mathcal{J}_u = (\mathcal{JP}, \delta)$ une interprétation de F^{HR} avec $\mathcal{JP} = \big\{(G, \varphi) : G \in \mathbb{J}^d \mid \varphi : \mathcal{A} \longrightarrow V(G)\big\}$ l'ensemble des multigraphes munit d'une application φ choisissant un sommet dans $V(G)$ (les sommets de G) pour chaque label. On définit δ par

$\delta(\mathrm{del}_a), (G)) =$ G ou le sommet de label a est retiré du graphe

$\delta(\mathrm{relab}_{a\leftrightarrow b}, (G)) =$ G où les étiquettes a et b sont échangées

$\delta(\mathrm{compose}, (G, G')) =$ $G \cup G'$ où les noeuds de même étiquette sont identitifiés

$\delta(\boldsymbol{e}) =$ le graphe à 2 sommets reliés : la source est de label 1 et la destination de label 2

$\delta(\mathbf{1}) =$ le graphe à 1 sommet sans boucle

On pose alors $\mathbb{F}_k^{\mathrm{HR}} = \big\{\delta(t) : t \in \mathcal{T}\big(F_k^{\mathrm{HR}}\big)\big\}$ et $\mathbb{F}^{\mathrm{HR}} = \big\{\delta(t) : t \in \mathcal{T}\big(F^{\mathrm{HR}}\big)\big\}$.

On remarque aussi que l'algèbre F^{HR} est une généralisation de l'algèbre des graphes en série parallèle et que $\mathbb{J}^p \subseteq \mathbb{F}_3^{\mathrm{HR}}$ en associant la source au label 1 et le puits au label 2 et par l'encodage des opérations :

$$\begin{aligned} G_1 \bullet_{\mathbb{J}^p} G_2 &\longrightarrow \mathrm{relab}_{2\leftrightarrow3}(\mathrm{del}_2(\mathrm{compose}(G_1, \mathrm{relab}_{2\leftrightarrow3}(\mathrm{relab}_{2\leftrightarrow1}(G_2))))) \\ G_1 /\!\!/_{\mathbb{J}^p} G_2 &\longrightarrow \mathrm{compose}(G_1, G_2) \\ \boldsymbol{e}_{G^c} &\longrightarrow \boldsymbol{e} \end{aligned}$$

Il est à noter que la classe des graphes de Tree-width bornée est une des plus grande classe de graphe connue où le problème de satisfaisabilité de la MSO est décidable.

4.2. Décomposition arborescente de graphes

Définition de Largeur Arborescente Soit $G = (V, E)$ un graphe non orienté, on appelle une *décomposition arborescente* une paire $\left(\mathcal{T}, (B_t)_{t \in T}\right)$ telle que $\mathcal{T} = (T, F)$ soit un arbre et $(B_t)_{t \in T}$ soit une famille de sous-ensemble de V telle que :

1. Pour tout $v \in V$, l'ensemble $B^{-1}(v) \coloneqq \{t \in T \mid v \in B_t\}$ est un sous-arbre non vide de $\mathcal{T}$
2. Pour chaque arête $\{v, w\} \in E$, il y a un $t \in T$ tel que $v, w \in B_t$

La *largeur* d'une décomposition arborescente est donné par $\max\{|B_t| : t \in T\} - 1$. La *largeur arborescente* de G (noté $\mathrm{tw}(G)$) est la plus petite largeur de toutes les décompositions arborescentes.

Lien avec $\mathbb{F}_k^{\mathbf{HR}}$ On a que pour tout G un graphe, $G \in \mathbb{F}_k^{\mathrm{HR}}$ ssi $\mathrm{tw}(G) \leq k + 1$

On pourrait se contenter de travailler sur la MSO sur les sommets et donc les graphe de tree-width borné. Mais afin d'englober toutes les MSO intéressantes, on va regarder la MSO sur les structures, et donc définir une décomposition arborescente de modèle de MSO sur les structures, et non pas que les graphes.

4.3. Décomposition arborescente de modèles (ou structures)

Nous avons défini la tree-width pour des graphes (objets finis avec une relation binaire). On s'intéresse maintenant à pouvoir le définir pour des objets avec autant de symboles de relation que l'on veux. Notamment, cela permettra d'avoir la logique MSO de graphe sur les arêtes.

Définition Soit $L = \langle \emptyset, R, \rho \rangle$ une logique, soit $m = \langle \mathbb{M}, \rho_\emptyset, \rho_R \rangle$ un modèle de L, on définit la *décomposition arborescente de m* (un modèle) comme une paire $\left(\mathcal{T}, (B_t)_{t \in T}\right)$ où $\mathcal{T} = (T, F)$ est un arbre et $(B_t)_{t \in T}$ est une famille de sous-ensembles de $\mathbb{M}$ tels que

1. Pour tout $v \in M$, l'ensemble $B^{-1}(v) \coloneqq \{t \in T \mid v \in B_t\}$ est un sous-arbre non vide de $\mathcal{T}$
2. Pour chaque relation $r \in R$ et chaque $\#(r)$-uplets $\left(m_1, ..., m_{\#(r)}\right) \in \mathbb{M}^{\#(r)}$ tel que $\rho_R\left(r, \left(m_1, ..., m_{\#(r)}\right)\right) = \top$, il y a un $t \in T$ tel que $\left(m_1, ..., m_{\#(r)}\right) \in B_t$

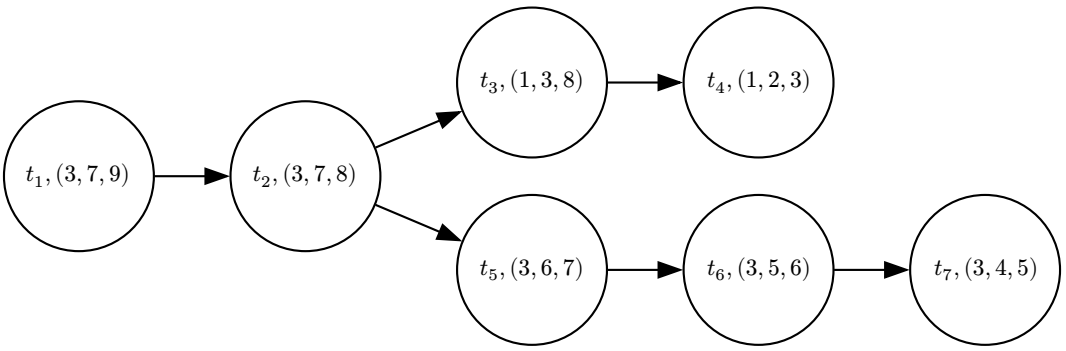
La *largeur* d'une décomposition arborescente est donnée par $\max\{|B_t| : t \in T\} - 1$. La *largeur arborescente* de m (noté $\mathrm{tw}(m)$) est la plus petite largeur de toutes les décompositions arborescentes.

Comme tous les ensembles (B_t) sont de cardinal borné par $\mathrm{tw}(m) + 1$, on les écrira sous la forme d'un $(\mathrm{tw}(m) + 1)$-uplet, quitte à avoir des doublons dans le $(\mathrm{tw}(m) + 1)$-uplet (par exemple $\{2, 3, 4\}$ sera $(3, 2, 3, 4)$ pour une Tree-Width de 3).

Exemple TW Soit une structure donnée par $R = \left\{r_1^{\#1}, r_2^{\#2}, r_3^{\#3}\right\}$ avec $m = \langle \mathbb{M}, \rho_V, \rho_R \rangle$ le modèle tel que:

- $\mathbb{M} = \{1, ..., 9\}$,
- $\{1, 2, 3\} = \{x \in \mathbb{M} \mid r_1(x)\}$
- $\{(1, 2), (2, 3), (3, 4), (4, 5), (6, 7), (7, 8), (8, 1)\} = \{(x, y) \in \mathbb{M}^2 \mid r_2((x, y))\}$
- $\{(a, b, c) \in \{1, 2, 3\}^3 \mid a \neq b \neq c \neq a\} = \{(x, y, z) \in \mathbb{M}^3 \mid r_3((x, y, z))\}$

Une décomposition arborescente possible serait alors l'arbre de sommets $T = \{t_1, ..., t_7\}$ donné par



5. Le probleme de satisfiabilite pour MSO

5.1. Preuve de Courcelle

L'idée derrière la preuve de Courcelle arrive naturellement après l'étude des *grammaire de termes* (une généralisation des grammaire pour les F-algèbres) et des *propriétés F-inductives* pour F une algèbre. Parce que ces notions ne sont pas au cœur de ce stage, je ne les a pas introduites. Toutefois, la preuve de Courcelle [1], [6] suit environ ces étapes :

- Montrer que si P est une propriété « F-inductive » et L un langage reconnu par une grammaire sur F, alors $\{t \in L \mid P(t)\}$ est aussi reconnu par une grammaire.
- Montrer que les satisfiabilités des formules MSO sont des propriétées F_k^{HR}-inductives.
- Montrer que pour tout langage L reconnu par une grammaire sur une F-algèbre, le problème de savoir si la grammaire est vide ou non est décidable. Précisément, $\{t \in \mathbb{C}_F \mid \rho(t) \in L\}$ avec $\mathbb{C}_F$ l'interprétation canonique est reconnu par un automate d'arbre.
- Et finalement, comme $\left\{t \in \mathbb{C}_{F_k^{\mathrm{HR}}} \mid \lfloor \rho(t) \rfloor \vDash \varphi\right\}$ est reconnu par un automate de termes, on peut savoir si l'automate est vide (φ est insatisfiable dans F_k^{HR}) ou non, et obtenir un exemple de graphe dans le cas où φ est satisfiable.

Mais cette preuve est très peu pratique : le nombre d'états de l'automate est hyper-exponentiel en le nombre de quantificateurs d'ensembles et ce qui se cache derrière le fait que les satisfiabilités des formules MSO sont des propriétées F-inductives est très compliqué.

C'est pourquoi on s'intéresse à réduire le problème de satisfiabilité de la MSO sur les structures de tree-width bornée en celui de la satisfiabilité sur les arbres, qui, elle, est décidable. Pour cela on regarde la réduction de **MSO** dans **MSOA** proposée dans [7], que l'on retravaillera pour minimiser le nombre de variables du second ordre et pour la transformer dans **WS2S** sur un alphabet vide. Une fois cette transformation faite, on pourra donner cette formule en entrée au logiciel Mona, qui est un solveur de **WS2S** avec alphabet vide.

5.2. Réduction d'une formule en MSO à une formule MSOA

Cette réduction est directement tirée de [7]. A noter que dans une section future elle sera simplifiée. On se fixe ici un $k \in \mathbb{N}$ et on s'intéresse aux modèles de tree-width $\leq k$.

Intuition Notre but est de transformer une formule logique $\varphi \in \mathcal{L}_{\mathrm{MSO}}(\emptyset, R)$ en une formule $\varphi^* \in \mathcal{L}_{\mathrm{MSOA}}$ telle que φ est satisfaisable par un modèle de tree-width bornée si et seulement si φ^* est satisfaisable par un arbre. Simplement considérer une formule φ^* et directement faire l'arbre $\left(\mathcal{T}, (B_t)_{t \in T}\right)$ ne marcherait pas car il nous faut un alphabet fini pour étiqueter l'arbre et les ensembles $(B_t)_{t \in T}$ ne sont pas bornés. A la place, on représentera l'ensemble $B_t = \{b_1, ..., b_{k+1}\}$ par la relation d'égalité sur les indices dans le couple $(B_t)_{t \in T}$.

Par exemple, le couple $(48, 23, 41, 48, 48, 23)$ sera représenté par l'ensemble $\{1 = 4, 1 = 5, 4 = 5, 2 = 6\}$. Afin d'être capable de propager cette information, on représentera aussi la relation d'égalité avec les éléments de notre parent.

Formalisme Soit $R \coloneqq \left\{R_1^{\#r_1}, ..., R_n^{\#r_n}\right\}$ et $m = \langle \mathbb{M}, \rho_\emptyset, \rho_R \rangle$ un modèle sur $\langle \emptyset, R \rangle$. On définit l'ensemble Σ des labels de notre arbre par

$$\Sigma = \left\{\left(\lambda_1, ..., \lambda_{k+1}, \lambda_{\text{eq}}, \lambda_{\text{pa}}\right) : \lambda_i \in \mathcal{P}([k+1]^{r_i}) : i \in [n] : \lambda_{\text{eq}}, \lambda_{\text{pa}} \in \mathcal{P}([k+1]^2)\right\}$$

Soit $\left(\mathcal{T}, (B_t)_{t \in T}\right)$ une décomposition arborescente de m. Pour tout sommet $t \in T$, on écrit $B_t = \{b_1^t, .., b_{k+1}^t\}$ (avec potentiellement des $b_i^t = b_j^t$ si $|B_t| < k+1$). Finalement, on pose

$$\lambda(t) \coloneqq \left(\lambda_1(t), ..., \lambda_n(t), \lambda_{\text{eq}}(t), \lambda_{\text{pa}}(t)\right) \in \Sigma$$

avec:

$$\begin{aligned}
\lambda_i(t) &\coloneqq \left\{\left(j_1, ..., j_{r_i}\right) \in [k+1]^{r_i} \mid \rho_R\left(R_i, \left(b_{j_1}^t, ..., b_{j_2}^t\right)\right) = \top\right\} \\
\lambda_{\text{eq}}(t) &\coloneqq \left\{(i, j) \in [k+1]^2 \mid b_i^t = b_j^t\right\} \\
\lambda_{\text{pa}}(t) &\coloneqq \begin{cases} \left\{(i, j) \in [k+1]^2 \mid b_i^t = b_j^s\right\} & \text{Si } s \text{ est le parent de } t \\ \emptyset & \text{Si } t \text{ est la racine} \end{cases}
\end{aligned}$$

On représentera un élément x de notre modèle par un $k+1$-uplets d'ensembles $U(x) = (U_1(x), ..., U_{k+1}(x))$ ou $U_i(x) \subseteq T$, tels que $t \in U_i(x) \Leftrightarrow b_i^t = x$. Informellement, $U_i(x)$ représente l'ensemble des nœuds de l'arbre où x apparaît en i-ème position. Ainsi on transformera $\exists x P$ en un $\exists U_1(x), ..., \exists U_{k+1}(x), \text{Elem}(U_1(x), ..., U_{k+1}(x)) \wedge P^*$, avec Elem une formule qui vérifie que $U_1(x), ..., U_{k+1}(x)$ sont bien des ensembles décrivant un élément du modèle m. On donne les détails de Elem juste après la réduction.

De la même manière, on représentera un ensemble d'éléments $X \subseteq \mathbb{M}$ dans notre modèle m par un $k+1$-uplets $\overline{U}(X) = \left(\overline{U_1}(X), ..., \overline{U_{k+1}}(X)\right)$ où pour tout i on a $\overline{U_i}(X) \subseteq T$, tel que $t \in \overline{U_i}(X) \Leftrightarrow b_i^t \in X$.

Exemple On reprend l'exemple **TW** précédent : on a $U(3) = (\{t_1, t_2, t_5, t_6, t_7\}, \{t_3\}, \{t_4\})$ et aussi $U(7) = (\emptyset, \{t_1, t_2\}, \{t_5\})$. On a $\overline{U}(\{9, 6, 2\}) = (\emptyset, \{t_5, t_4\}, \{t_1, t_6\})$ et finalement $\overline{U}(\{3, 7, 8\}) = (\{1, 2, 5, 6, 7\}, \{1, 2, 3\}, \{2, 3, 4, 5\})$

On peut maintenant effectuer notre transformation d'une formule $\varphi \in \mathcal{L}_{\text{MSO}}(\emptyset, R)$ par induction:

- Si $\varphi =$ "$\psi \wedge \chi$", on pose $\varphi^* \coloneqq \psi^* \wedge \chi^*$, et pareillement pour $\vee$, $\rightarrow$ et $\neg$
- Si $\varphi =$ "$\exists a, \psi$", on pose $\varphi^* \coloneqq \exists U_1(a), ..., \exists U_{k+1}(a), \text{Elem}(U_1(a), ..., U_{k+1}(a)) \wedge \psi^*$
- Si $\varphi =$ "$\forall a, \psi$", on pose $\varphi^* \coloneqq \forall U_1(a), ..., \forall U_{k+1}(a), \text{Elem}(U_1(a), ..., U_{k+1}(a)) \rightarrow \psi^*$
- Si $\varphi =$ "$\exists A, \psi$", on pose $\varphi^* \coloneqq \exists \overline{U_1}(A), ..., \exists \overline{U_{k+1}}(A), \text{Set}\left(\overline{U_1}(a), ..., \overline{U_{k+1}}(a)\right) \wedge \psi^*$
- Si $\varphi =$ "$\forall A, \psi$", on pose $\varphi^* \coloneqq \forall \overline{U_1}(A), ..., \forall \overline{U_{k+1}}(A), \text{Set}\left(\overline{U_1}(a), ..., \overline{U_{k+1}}(a)\right) \rightarrow \psi^*$
- Si $\varphi =$ "$R_i\left(a_1, ..., a_{r_i}\right)$", on pose

$$\varphi^* \coloneqq \exists x, \bigvee_{i_1, ..., i_{r_i} \in [k+1]} x \in U_{i_1}(a_1) \wedge ... \wedge x \in U_{i_{r_i}}\left(a_{r_i}\right) \wedge \left(\bigvee_{\substack{\alpha = (\lambda_1, ..., \lambda_n, \lambda_{\text{eq}}, \lambda_{\text{pa}}) \in \Sigma \\ \left(i_1, ..., i_{r_i}\right) \in \lambda_i}} p_\alpha(x) \right)$$

- Si $\varphi =$ "$a = b$", on pose

$$\varphi^* = \exists x, \bigvee_{i \neq j} x \in U_i(a) \wedge x \in U_j(b) \wedge \left(\bigvee_{\substack{\alpha=(\lambda_1,\dots,\lambda_n,\lambda_{\text{eq}},\lambda_{\text{pa}})\in\Sigma \\ (i,j)\in\lambda_{\text{eq}}}} p_\alpha(x) \right)$$

- Si $\varphi =$ "$a \in B$", on pose

$$\varphi^* = \exists x, \bigvee_{i \neq j} x \in U_i(a) \wedge x \in \overline{U_j}(B)$$

Il ne nous reste plus qu'à expliciter la formule $\text{Set}(U_1, ..., U_{k+1})$ et $\text{Elem}(U_1, ..., U_{k+1})$. Pour cela on décompose les formules en 2 (respectivement 5) conditions que les ensembles $U_1, ..., U_{k+1}$ doivent respecter.

Pour Set, seules 2 conditions sont nécessaires et suffisantes pour garantir que $\left(\overline{U_1}, .., \overline{U_{k+1}}\right)$ soit de la forme $\overline{U}(X)$ avec $X \subseteq \mathbb{M}$: le fait que les ensembles se comportent bien vis-à-vis de l'égalité et vis-à-vis des parents. Précisément, ce sont :

1. Si $(i,j) \in \lambda_{\text{eq}}(x)$, alors $x \in \overline{U_i}(X) \leftrightarrow p \in \overline{U_j}(X)$
2. Si f est une fille de p son parent et que $(i,j) \in \lambda_{\text{pa}}(f)$, alors $f \in \overline{U_i}(x) \leftrightarrow p \in \overline{U_j}(X)$

La preuve de l'équivalence est dans [7], et les formules s'expriment en MSO par

$$\varphi_1 := \forall x, \bigwedge_{i,j\in[k+1]} \left(\left(\bigvee_{\substack{\alpha=(\lambda_1,\dots,\lambda_n,\lambda_{\text{eq}},\lambda_{\text{pa}})\in\Sigma \\ (i,j)\in\lambda_{\text{eq}}}} p_\alpha(x) \right) \rightarrow \left(x \in \overline{U_i} \leftrightarrow x \in \overline{U_j} \right) \right)$$

$$\varphi_2 := \forall f, \forall p, \text{edg}(p,f) \rightarrow \bigwedge_{i,j\in[k+1]} \left(\left(\bigvee_{\substack{\alpha=(\lambda_1,\dots,\lambda_n,\lambda_{\text{eq}},\lambda_{\text{pa}})\in\Sigma \\ (i,j)\in\lambda_{\text{pa}}}} p_\alpha(f) \right) \rightarrow \left(f \in \overline{U_i} \leftrightarrow p \in \overline{U_j} \right) \right)$$

Pour Elem, nous avons 3 autres conditions à respecter : comme l'élément est « unique », les réciproques de φ_1 et φ_2, et le fait que $\bigcup_i U_i(x)$ est un sous-arbre non vide (définition de tree-width). Précisément :

3. Si $t \in U_i$ et $t \in U_j$, alors $(i,j) \in \lambda_{\text{eq}}(t)$
4. Si f est un enfant de p, et si $f \in U_i$ et $p \in U_j$, alors $(i,j) \in \lambda_{\text{pa}}(t)$
5. $\bigcup_{i\in[k+1]} U_i$ est connecté et non vide

La preuve de l'équivalence est aussi dans [7], et les formules s'expriment en MSO par

$$\varphi_3 := \forall x, \bigwedge_{i,j\in[k+1]} \left(x \in U_i \wedge x \in U_j \rightarrow \bigvee_{\substack{\alpha=(\lambda_1,\dots,\lambda_n,\lambda_{\text{eq}},\lambda_{\text{pa}})\in\Sigma \\ (i,j)\in\lambda_{\text{eq}}}} p_\alpha(x) \right)$$

$$\varphi_4 := \forall f, \forall p, \text{edg}(p,f) \rightarrow \bigwedge_{i,j\in[k+1]} \left(\left(f \in U_i \wedge p \in U_j \right) \rightarrow \bigvee_{\substack{\alpha=(\lambda_1,\dots,\lambda_n,\lambda_{\text{eq}},\lambda_{\text{pa}})\in\Sigma \\ (i,j)\in\lambda_{\text{pa}}}} p_\alpha(f) \right)$$

$$\varphi_5 := \exists z, \left(\bigvee_{i\in[k+1]} z \in U_i \right) \wedge \left(\forall f, \forall p, \text{edg}(p,f) \wedge \left(\bigvee_{i\in[k+1]} f \in U_i \right) \rightarrow f = z \vee \left(\bigvee_{i\in[k+1]} p \in U_i \right) \right)$$

A noter que la formule donnée pour φ_5 par [7] est différente et suspecte.

5.3. Réduction de MSOA à WS2S

Proposition Pour $\varphi \in \mathcal{L}_{\text{MSO}}(R)$ on a l'existence d'un arbre T tel que $\lfloor T \rfloor_{\text{MSOA}} \vDash \varphi^*$ si et seulement si on a l'existence d'un arbre binaire T' tel que $\lfloor T' \rfloor_{\text{MSOA}} \vDash \varphi^*$.

La preuve de cette proposition est totalement de moi.

Preuve La réciproque est simple : si on a un arbre binaire T tel que $\lfloor T \rfloor_{\text{MSOA}} \vDash \varphi^*$, alors on a le même arbre qui satisfait $\lfloor T \rfloor_{\text{MSOA}} \vDash \varphi^*$, car les arbres binaires sont des arbres.

Pour le sens direct on considère la transformation qui prend un nœud d'arité > 2 $\alpha(x_1, x_2, ..., x_n)$ avec $\alpha = (\lambda_1, ..., \lambda_m, \lambda_{\text{eq}}, \lambda_{\text{pa}})$ et qui le transforme vers $\alpha(x_1, \alpha'(x_2, x_3, ..., x_{n-1}, x_n))$ avec $\alpha' = (\lambda_1, ..., \lambda_m, \lambda_{\text{eq}}, \lambda'_{\text{pa}} := \{(i, i) : i \in [k+1]\})$

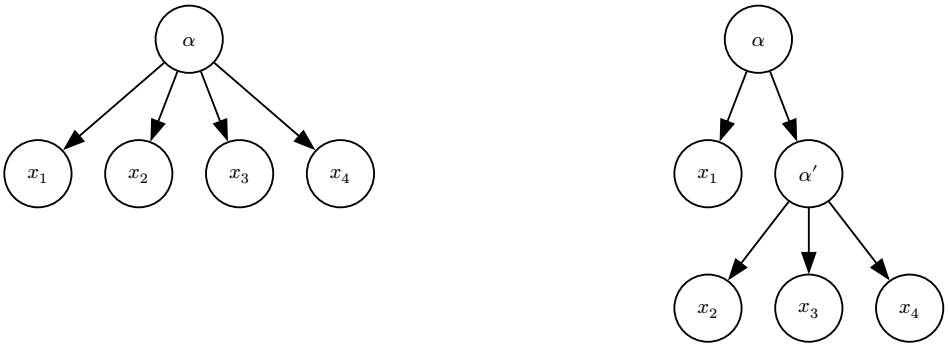


Fig. 3. – Exemple d'une étape de transformation

On notera $T \to T'$ quand T' est T sur lequel on a appliqué cette transformation sur un des sommets de degré maximal. La relation est terminante sur les arbres fini (l'ordre lexicographique sur les couples (a, b) avec a le degré max de l'arbre et b le nombre de nœuds de degré a décroît strictement à chaque itération), et un T est sous forme normale est toujours binaire. En montrant que la satifiabilité dans **MSOA** est un invariant, on montrera que pour tout arbre T tel que $\lfloor T \rfloor \vDash \varphi^*$ on a l'existence d'un T' (une[1] forme normale de T) qui est un arbre binaire tel que $\lfloor T' \rfloor \vDash \varphi^*$.

Afin de simplifier un peu la preuve, on dit qu'une formule φ est *locale* si elle ne possède pas de quantificateurs. On remarque que dans ce cas, seuls les symboles des variables libres de φ et les relations sur les variables libres de φ vont décidé de la satisfiabilité de φ. On note e le nœud de T de symbole α qui c'est transformé en $\overline{e}$ dans T' (celui de symbole α) et on note e' le nœud de T' qui a été ajouté (celui de symbole α'). Pour tout modèle $m = \langle N(T), \rho_F, \rho_R \rangle$ on défini $[m] = \langle N(T'), \rho'_F, \rho'_R \rangle$ tel que

$$\rho'_R(X, (\overline{e})) = \rho'_R(X, (e')) = \begin{cases} \top \text{ si } \rho_R(X, (e)) = \top \\ \bot \text{ sinon} \end{cases} \qquad \text{pour } X \in \widetilde{\text{Var}}$$

$$\rho'_R(r, c) = \rho_R(r, c) \qquad \text{pour tout les autres relations}$$

$$\rho'_F(x) = \begin{cases} \rho_F(e) \text{ si } x \in \{\overline{e}, e'\} \\ \rho_F(x) \text{ si } x \in N(T) \end{cases} \qquad \text{pour } x \in \text{Var}$$

Intuitivement, si on a un modèle m, on définit $[m]$ comme le modèle similaire sur T' où dans tous les ensembles et valeurs de valuation, e à été remplacé par $\overline{e}, e'$. Comme pour une formule close on a $\lfloor T' \rfloor = [\lfloor T \rfloor]$, on montre par induction sur la formule φ que $m \vDash \varphi^* \Rightarrow [m] \vDash \varphi^*$. En regroupant des cas, il y a 5 cas :

[1] Elle est unique, mais ce n'est pas nécessaire pour la preuve.

- Si $\varphi =$ "$x \in Y$", alors $m \vDash \exists x. \left(\bigvee_{i \in [k+1]} x \in Y_i \wedge x \in X_i \right)$. En posant $\psi(x) :=$ "$\left(\bigvee_{i \in [k+1]} x \in Y_i \wedge x \in X_i \right)$", ψ est locale. On choisi t le nœud de T qui permet de satisfaire $\psi(t)$.
  - Si $t = e$, on prend $t = \overline{e}$ dans T'. Comme ils ont le même symbole d'alphabet, $[m] \vDash \exists x. \psi(x)$
  - Si $t \neq e$, on garde le même t. Comme ils ont le même symbole d'alphabet, $[m] \vDash \exists x. \psi(x)$
- Si $\varphi =$ "$R_i(x_1, ..., x_n)$", alors le même raisonnement que précédemment fonctionne (juste avec une formule plus compliquée, mais qui est aussi locale).
- Si $\varphi =$ "$\psi \wedge \gamma$", alors $m \vDash \psi^* \wedge \gamma^*$. Donc $m \vDash \psi^*$ et $m \vDash \gamma^*$. Donc $[m] \vDash \psi^*$ et $[m] \vDash \gamma^*$ par hypothèse d'induction, donc $[m] \vDash \psi^* \wedge \gamma^*$.

  On fait les même raisonnement pour $\vee$, $\neg$, $\rightarrow$ et $\leftrightarrow$
- Si $\varphi =$ "$\exists Y. \psi$" alors $m \vDash \exists Y_1, ..., \exists Y_{k+1}, \text{set}(Y_1, ..., Y_{k+1}) \wedge \psi^*$. On garde alors les mêmes ensembles $Y_1, ..., Y_{k+1}$ pour T', juste on remplace e par $\overline{e}$ et e' dans le Y_i qui contenait e. On montre maintenant que $[m] \vDash \text{set}(Y_1, ..., Y_n)$ ssi $m \vDash \text{set}(Y_1, ..., Y_n)$, et comme par hypothèse d'induction $m \vDash \psi^*$ ssi $[m] \vDash \psi^*$, cela suffit. Il y a 2 formules à vérifier :
  - Supposons $m \vDash \text{set}(Y_1, ..., Y_n)$ et montrons $[m] \vDash \varphi_1$. On doit donc montrer que $\forall x, P(x)$ avec P qui est une formule locale. Pour tout les nœuds inchangés, $P(x)$ reste vrai par localité, on le montre donc pour $\overline{e}$ et e' .
    - Pour $x = \overline{e}$, on a que $P(e)$ est vrai par localité ($\overline{e}$ à le même symbole que e)
    - Pour $x = e'$, on a que $P(e')$ est vrai car $P(e)$ est vrai que et que P ne traite que de la composante λ_{m+1} qui est la même entre e et e'
  - Supposons $m \vDash \text{set}(Y_1, ..., Y_n)$ et montrons $[m] \vDash \varphi_2$. On doit donc montrer que $\forall s, t, \text{parent}(s, t) \rightarrow P(s, t)$ avec P qui est une formule locale. Pour tous les couples $(\text{parent}, \text{enfant})$ inchangés, $P(s, t)$ reste vrai par localité, on le montre donc pour le couple $(\overline{e}, e')$, et les couples de la forme $(\overline{e}, x)$ et (e', x).
    - Pour les couples $(\overline{e}, x)$, on a que $P(\overline{e}, x)$ est vrai dans T' par localité ($\overline{e}$ à le même symbole que e et l'arête (e, x) existait dans T)
    - Pour les couples (e', x), on a que $P(e', x)$ est vrai car $P(e, x)$ est vrai dans T que et que P ne traite que de la composante λ_{m+2} de x qui est la même.
    - Pour le couple $(\overline{e}, e')$, on doit vérifier que pour tout ensemble Y, on a $\overline{e} \in Y \Leftrightarrow e' \in Y$, car en écrivant $e' = (\lambda_1, .., \lambda_{m+2})$ on a $\lambda_{m+2} = \{(i, i) : i \in [k+1]\}$. Mais cela est vrai car tout les ensembles qui contenait e sont devenus des ensembles qui contiennent $\overline{e}$ et e'.
- Si $\varphi =$ "$\exists y. \psi$" alors $m \vDash \exists Y_1, ..., \exists Y_{k+1}, \text{elem}(Y_1, ..., Y_{k+1}) \wedge \psi^*$. On garde alors les mêmes ensembles $Y_1, ..., Y_{k+1}$ pour T', on remplace juste e par $\overline{e}$ et e' dans le Y_i qui contenait e. On montre maintenant que $[m] \vDash \text{elem}(Y_1, ..., Y_n)$ ssi $m \vDash \text{elem}(Y_1, ..., Y_n)$. Il y a 5 formules à vérifier dont 2 qui ont déjà été traitées au cas précédent. On a donc 3 cas :
  - Les deux cas de $m \vDash \varphi_3$ et $m \vDash \varphi_4$ ont sensiblement la même justification qu'avant : formule locale et on fait une disjonction sur x ou sur l'arête considérée.
  - Il ne reste que le cas avec φ_5. Mais on rappelle que φ_5 désigne que les ensembles $X_1, ..., X_{k+1}$ sont non vide et connectés. Comme la notion de connexion / ensemble de nœuds non vides est la même dans les arbres binaires et non binaires, la satisfiabilité de la formule est la même.

En écrivant la quantification universelle comme la négation de la quantification existentielle, cela conclut la preuve. □

Corollaire Pour $\varphi \in \mathcal{L}_{\text{MSO}}(\emptyset, R)$, il existe une formule $\varphi' \in \mathcal{L}_{\text{WS2S}}$ telle que φ soit satisfiable par un m de tree-width bornée par k ssi φ' est satisfiable dans **WS2S**.

Pour montrer cela, on considère une transformation de φ^* en $[\varphi^*]$ ou l'on a remplacé "$\text{edg}(x, y)$" par "$s_1(x) = y \vee s_2(x) = y$" dans φ^*. On pose alors $\varphi' := [\varphi^*]$

5.4. Passage de WS2S à Mona

Mona travaille sur la logique **WS2S** sur alphabet vide (sans étiquetage des nœuds). Comme Mona ne permet pas d'étiqueter l'arbre par nos éléments de Σ, on va représenter l'étiquette $\alpha \in \Sigma$ d'un nœud x par des ensembles dans lequel x peut appartenir. La manière naïve serai pour chaque $\alpha \in \Sigma$ de crée un ensemble X_α. Mais cela créerai un nombre exponentiel d'ensembles. A la place, l'on crée des ensembles $\text{Eq}_{i,j}$ et $\text{Pa}_{i,j}$ pour tout $i, j \in [k+1]$, et on crée des ensembles $\text{Rel}^r_{i_1,\dots,i_{\#(r)}}$ pour tout $r \in R$ et $\left(i_1, \dots, i_{\#(r)}\right) \in [k+1]^{\#(r)}$.

Par exemple, un nœud $(48, 23, 41, 48, 48, 23)$ qui est représenté par l'ensemble $\{1 = 4, 1 = 5, 4 = 5, 2 = 6\}$ sera maintenant représenté par un nœud appartenant aux ensembles $\text{Eq}_{1,4}, \text{Eq}_{1,5}, \text{Eq}_{4,5}, \text{Eq}_{2,6}$

La première optimisation que cela apporte c'est que l'on soit capable de ré-exprimer les

$$\bigvee_{\substack{\alpha=(\lambda_1,\dots,\lambda_n,\lambda_{\text{eq}},\lambda_{\text{pa}})\in\Sigma \\ (i,j)\in\lambda_{\text{eq}}}} p_\alpha(f)$$

par un simple "$f \in \text{Eq}_{i,j}$", et similairement pour les $\text{Pa}_{i,j}$ et $\text{Rel}^r_{i_1,\dots,i_{\#(r)}}$.

6. Implémentation

6.1. Optimisations

Comme le nombre de variables ensembliste est la première cause à la complexité hyper-exponentielle du problème de décision, on va dans cette partie discuter de différentes optimisations pour essayer de réduire le nombre de variables ensemblistes. Pour l'instant, nous avons $(k+1)^2$ variables de la forme $\text{Eq}_{i,j}$, $(k+1)^2$ variables de la forme $\text{Pa}_{i,j}$ et $\sum_{r \in R} (k+1)^{\#(r)}$ variables de la forme $\text{Rel}^r_{i_1,\dots,i_{\#(r)}}$.

Egalité Par symétrie et réflexivité de l'égalité, il est possible de se limiter que aux variables $\text{Eq}_{i,j}$ avec $i < j$, pour un total de $\frac{k(k+1)}{2}$. Utiliser la transitivité de l'égalité pour réduire le nombre de variables est théoriquement possible mais demande de changer la forme des variables, on ne le considérera donc pas cette optimisation ici.

Relation Ceci est la plus grosse et majeur optimisation. J'ai montrée en annexe qu'une seule variable d'ensemble pour chaque relation est suffisante et nécessaire, en montrant que si on a $m \vDash \varphi$ avec m de tree-width bornée, alors il existe $[m]$ ou chaque $r(x_1, \dots, x_n)$ satisfait dans B_t est tel que x_i est le i -ème élément. Et dans ce cas, seul $\text{Rel}^r_{1,2,3,\dots,\#(r)}$ est nécessaire.

Conclusion Dans le cas d'une unique relation $\text{Edg}^{\#2}$, on a retiré $\frac{(k+2)(k+1)}{2} + (k+1)^2 - 1$ variables du second ordre. Pour une tree-width de 1, cela donne 6 variables en moins, pour un total qui passe de 12 à 6.

6.2. Algorithme & Mona

En rust, j'ai ré-implémenté un sous-ensemble du parser de Mona [2] (celui qui correspond à la MSO telle que décrite ici) via la crate **pest.rs**. Une structure paramétré `struct MSO<F,R>` représente exactement $\mathcal{L}(F, R)$, et ainsi différentes MSO ont été définies qui sont des instanciations de cette structure paramétrique. J'ai utilisée la crate `clap` pour faire une CLI avec message d'aide et retour utilisateur.

La fonction `pub fn wvar_to_wmsoa(m: &Wvar, tw: usize) -> Wmsoa` s'occupe de faire la traduction et la méthode `pub fn wvar_to_mona(&self, tw: usize) -> String` s'occupe de faire la conversion

dans la syntaxe Mona d'une formule WS2S. A noter que dans `Wmsoa`, je ne stoque pas les φ_i ni Elem ni Set, mais une relation $\mathrm{Elem}_{U,k}$ et $\mathrm{Set}_{U,k}$ qui sera transformé en un prédicat $\mathrm{Set}(U_1, ..., U_{k+1})$ dans Mona car Mona permet la définition de prédicats.

Voir dans les annexes pour un exemple de traduction de formule généré par mon implémentation.

Le code source est disponible sur https://gitlab.aliens-lyon.fr/cypooos/mso-converter

6.3. Résultats

Du fait de la complexité de la réduction, il m'a été impossible de faire tourner Mona sur des formules de tree-width ≥ 2. J'ai pu faire tourner Mona sur un serveur de calcul, mais qui au final a été à chaque fois été plus lent que mon PC personnel. A noter toutefois que le serveur possédais plus de RAM que mon PC, ce qui explique que la dernière formule à été trouvé par le serveur mais pas le PC.

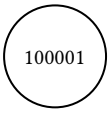
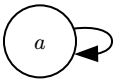
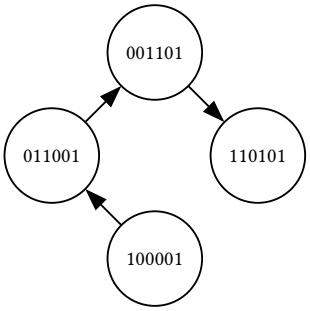
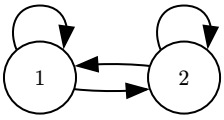
Pour donner un exemple de formule satisfiable seulement pour une tree-width ≥ 2, on a l'existence d'une clique de taille ≥ 3 : $(\exists a, \exists b, \exists c, a \neq b \wedge b \neq c \wedge c \neq a) \wedge (\forall a, \forall b, \mathrm{edg}(a, b))$.

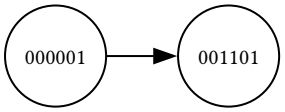
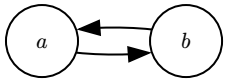
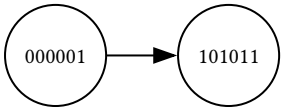
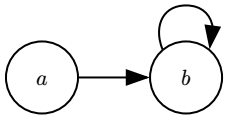
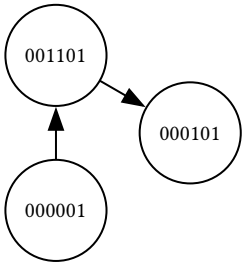
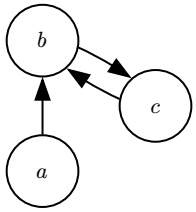
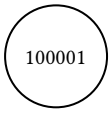
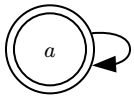
Dans le tableau récapitulatif, on ne considère ici que un symbole de relation Edg d'arité 2 et un symbole P d'arité 1, et on représentera les modèles donnée par des graphes orienté ou les nœuds respectant P seront doublement encerclé.

Mona affiche sur chaque nœud de l'arbre un mot indiquant pour chaque variable $\mathrm{Eq}_{1,2}, \mathrm{Pa}_{1,1}, \mathrm{Pa}_{1,2}, \mathrm{Pa}_{2,1}, \mathrm{Pa}_{2,2}$ si le nœud y appartient (il y a un 1) ou pas. Les variables de relation sont mise à la fin dans l'ordre de lecture de gauche à droite. Par exemple, si Edg est la seule relation et que un nœud est étiqueté par 001101, cela veux dire qu'il est dans $\mathrm{Pa}_{1,2}, \mathrm{Pa}_{2,1}, \mathrm{Rel}^{\mathrm{Edg}}_{1,2}$ et qu'il n'est pas dans $\mathrm{Eq}_{1,2}, \mathrm{Pa}_{1,1}$ ni $\mathrm{Pa}_{2,2}$

La version PC a tourné sous un linux NIXOS interface plasma KDE avec 8Go de ram, un processeur i7 9ème génération. La version Serveur à tournée sous un linux Ubuntu 22.04LTS par SSH avec 512GB de Ram et un processeur Xeon Gold 6330/56cores/2Ghz. Le nombre de cœurs ou la carte graphique sont irrévérents car Mona ne les utilise pas. La métrique utilisé est celle renvoyé par le programme Mona.

Tests SAT Ci-après un extrait du tableau donné en annexe qui contiens les différents tests SAT réalisés.

Formule	**Temps**	**Output Mona**	**Modèle**
$\exists a, \forall b, \mathrm{R}(a, b)$	PC : 0.20s Serveur: 0.36s		
$(\forall a, \forall b, \mathrm{Edg}(a, b))$ $\wedge (\exists a, \exists b, a \neq b)$	PC : 2.24s Serveur : 2.58s		

Formule	Temps	Output Mona	Modèle
$(\forall a, \exists b, R(a,b)) \wedge$ $(\forall a, \neg R(a,a))$	PC : 4.53s Serveur : 5.03s		
$(\forall a, \exists b, R(a,b)) \wedge$ $(\exists a, \forall b, \neg R(b,a))$	PC : 17.50s Serveur : 19.08s		
$(\forall a, \exists b, R(a,b)) \wedge$ $(\exists a, \forall b, \neg R(b,a)) \wedge$ $(\forall a, \neg R(a,a))$	PC : 18.56s Serveur : 20.06s		
$\exists A, \forall a, a \in A \wedge \mathrm{P}(a)$	PC : crash Serveur : 3m 51s		

Tests UNSAT Ci-après un extrait du tableau donné en annexe qui contiens les différents tests insatisfiable (aucun modèle n'existe) réalisés.

Formule	Temps	Formule	Temps
$\exists x, R(x,x) \wedge$ $(\forall y, \neg(R(x,y))$	PC : 0.34s Serveur : 0.56s	$(\exists y, \forall x, R(x,y)) \wedge$ $\neg(\forall x, \exists y, R(x,y))$	PC: 4.05s Serveur : 4.57s

7. Conclusion

Pour beaucoup de raisons décrite on ne s'attendait pas d'être capable de faire tourner la réduction sur des Tree-Width très grandes. Mais malheureusement, on n'a pas pu monter à la tree-width 2.

La partie la plus chronophage du stage, derrière le temps de compréhension de beaucoup de notions (certaines qui n'ont pas été définies ici comme les systèmes d'équation d'ensembles, les propriétés F -inductives et les langages équationnels), a été la réalisation des tests sur le serveur du laboratoire. A cause de différents problèmes administratif, par exemple le fait que ma date de naissance était erronée dans la base de données des services de l'Université de Grenoble, j'ai mis ~1 semaine avant d'être capable de faire tourner les tests sur le serveur. Tout cela pour des résultats pas plus intéressant que sur mon PC personnel.

Pour aller plus loin, il sera possible de se demander comment on peut encore effectuer d'autres optimisations dans le nombre d'ensembles pour réduire le nombre d'ensemble libre dans la traduction final. Rien que prendre en compte la transitivité de l'égalité et le lien entre égalité et la relation parent nous permettrai de retirer un certains nombre de variables, mais il faudrait changer leur forme. De

plus, la logique CMSO (non introduite ici) qui rajoute un prédicat "$|X| \equiv 0[n]$" pour tout $n \in \mathbb{N}$ et $X \in \widetilde{\mathrm{Var}}$ est aussi solvable, tout en étant plus expressive dans les arbres d'arité non bornée. Mais car CMSO et MSO on la même expressivité pour les arbres d'arité finie (comme WS2S) [1], la réduction sera beaucoup plus difficile, si même possible.

Bibliographie

[1] B. Courcelle, « The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs », *INFORMATION AND COMPUTATION 85, 12-75 (1990).*

[2] N. Klarlund et A. Møller, « MONA Version 1.4 User Manual ». janvier 2001.

[3] B. Courcelle et J. Engelfriet, *Graph Structure and Monadic Second-Order Logic*, Encyclopedia of Mathematics and Its Application 138. Cambridge.

[4] B. Khoussainov et A. Nerode, *Automata Theory and its Applications.*

[5] A. Mostowski, « B. A. Trahténbrot. Névozmožnost' algorifma dlá problémy razréšimosti na konéčnyh klassah (Impossibility of an algorithm for the decision problem in finite classes). Doklady Akademii Nauk SSSR, vol. 70 (1950), pp. 569–572. », *Journal of Symbolic Logic*, vol. 15, nº 3, p. 229, 1950, doi: 10.2307/2266840.

[6] J. (János) A. Makowsky, « Algorithmic uses of the Feferman–Vaught Theorem ». 2004.

[7] J. Flum et M. Grohe, *Parameterized Complexity Theory.* Springer, p. 280-285.

Fin du rapport

8. Annexes

8.1. Contexte social du stage (annexe obligatoire)

Le stage s'est très bien passé, j'étais dans un bureau de (initialement) 4 personnes. J'ai fini avec une avance considérable, qui m'a permis de passer du temps sur mon rapport et sur les preuves de mes optimisations. Chaque midi j'ai pu manger avec l'équipe, et entre des discussions politique et humaines, on a pu se partager des problèmes intéressant.

Le laboratoire est à taille humaine, et représente un (1) étage du bâtiment IMAG. Mon maître de stage est partis en vacances pendant 2 semaines en milieu Juillet, et m'a confié à la charge d'un autre chercheur. Toutefois, j'ai été particulièrement autonome dans mon implémentation, la création des tests, et mes preuves (qui ont été relu). En tout le stage était bien. Il correspondait à ma vision de la recherche.

8.2. Preuve de que seul $\mathrm{Rel}^r_{1,2,3,\dots,\#(r)}$ est nécessaire pour chaque relation

On note k la tree-width, ici fixée. On décrit pour φ une formule MSO la formule φ' décrite dans le Chapitre 5.3 qui est la version WS2S sur alphabet vide de φ. On note par φ'' la version où l'on remplace tout les occurrences de

$$\exists x, \bigvee_{i_1,\dots,i_{r_i}\in[k+1]} x \in U_{i_1}(a_1) \wedge \dots \wedge x \in U_{i_{r_i}}\left(a_{r_i}\right) \wedge \left(x \in \mathrm{Rel}^r_{i_1,\dots,i_{r_i}}\right) \tag{1}$$

par

$$\exists x, x \in U_1(a_1) \wedge \dots \wedge x \in U_{r_i}\left(a_{r_i}\right) \wedge \left(x \in \mathrm{Rel}^r_{1,2,\dots,r_i}\right)$$

On rappelle que dans la transformation de φ, la formule (1) apparaît si φ est de la forme $r_i\left(x_1, \dots, x_{r_i}\right)$

On décrit l'énoncé que l'on souhaite prouver : pour tout formule $\varphi \in \mathcal{L}_{\mathrm{MSO}}$, φ' satisfiable dans WS2S si et seulement si φ'' est satisfiable dans WS2S. L'idée est pour chaque nœud qui appartient à un $\mathrm{Rel}^r_{i_1,\dots,i_n}$ de lui donner un fils qui appartient à $\mathrm{Rel}_{1,\dots,\#(r)}$ ou les $\mathrm{Pa}_{i,j}$ encode la permutation $\begin{pmatrix} i_1 & i_2 & \dots & i_{\#(r)} \end{pmatrix}$. Et si $\#(r) < k+1$, alors on met les $k+1-\#(r)$ éléments restant tous égaux à $i_{\#(r)}$

Transformation de l'arbre Soit $\lfloor T \rfloor$ un modèle, on considère la transformation d'un nœud $\alpha(x_1, x_2)$ présent dans $\mathrm{Rel}^r_{i_1,\dots,i_{\#(r)}}$ en $e(e'(\varepsilon,\varepsilon), e''(x_1,x_2))$. Dans ce cas on pose T' le même arbre que T avec :

- e dans les même ensembles que α sauf pour $\mathrm{Rel}^r_{i_1,\dots,i_{\#(r)}}$ ou il n'y sera pas
- e' seulement dans les ensembles :
  - $\mathrm{Eq}_{j,j'}$ si $\alpha \in \mathrm{Eq}_{i_j,i_{j'}}$ pour $0 < j < j' \leq \#(r)$
  - $\mathrm{Eq}_{i,j}$ pour $\#(r) \leq i < j \leq k+1$
  - Pa_{j,i_j} pour $0 < i \leq \#(r)$
  - $\mathrm{Pa}_{j,j'}$ si $\alpha \in \mathrm{Eq}_{j',i_j}$ ou $\alpha \in \mathrm{Eq}_{i_j,j'}$, pour $0 < j', i_j \leq \#(r)$
  - $\mathrm{Pa}_{j,i_{\#(r)}}$ pour $\#(r) \leq j \leq k+1$
  - $\mathrm{Rel}^r_{1,2,\dots,\#(r)}$
- e'' seulement dans les ensembles :
  - $\mathrm{Pa}_{i,i}$ pour $0 < i \leq k+1$
  - $\mathrm{Eq}_{i,i}$ pour chaque i, j tel que $\alpha \in \mathrm{Eq}_{i,j}$ – les mêmes ensembles Eq que α –

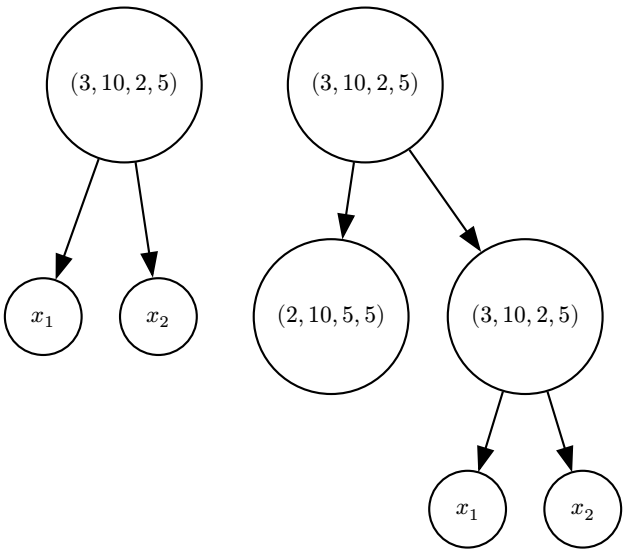


Fig. 4. – Exemple d'une transformation d'une décomposition arborescente avec le nœud α (qui encode ici $(3, 10, 2, 5)$) qui appartient à $\mathrm{Rel}^r_{3,2,4}$, et sa transformation pour que son fils gauche appartiennent à $\mathrm{Rel}^r_{1,2,3}$

On notera $T \to T'$ quand T' est T sur lequel on a appliqué cette transformation sur un des nœuds. La relation est terminante car le nombre de noeuds α qui appartiennent à un $\mathrm{Rel}^r_{i_1,\dots,i_{\#(r)}}$ avec $\left(i_1, \dots, i_{\#(r)}\right) \neq (1, 2, .., \#(r))$ décroit de 1 à chaque transformation, et un T en forme normale n'a que des nœuds dans des ensembles associé à la relation r de la forme $\mathrm{Rel}^r_{1,\dots,\#(r)}$.

Transformation du modèle La seule chose qu'il reste à définir pour définir le modèle correspondant à T' sont les valeurs sur les variables libre du premier et second ordre. On remarque que soit φ une formule (pas nécessairement close), toutes les variables libre du second ordre de φ' existent en blocs de $k+1$ variables. Pour chaque $x \in \mathrm{FV}(\varphi) \cup \widetilde{\mathrm{FV}}(\varphi)$, on les nottera $(U(x)_i)_{0<i\leq k+1}$. Soit $m \coloneqq \langle N(T), \rho_F, \rho_R\rangle$ un modèle de φ', on défini alors $[m] \coloneqq \langle N(T'), \rho'_F, \rho'_R\rangle$ un modèle de φ' tel que pour $x \in \mathrm{FV}(\varphi)$ et pour tout $X \in \widetilde{\mathrm{FV}}(X)$, on ai

$$
\begin{aligned}
&\rho'_R\big(U(X)_j, (e)\big), \rho'_R\big(U(X)_j, (e'')\big) \coloneqq \rho_R\big(U(X)_j, (\alpha)\big)\\
&\rho'_R\big(U(X)_j, (e')\big) \coloneqq \rho_R\Big(U(X)_{i_j}, (\alpha)\Big)\\
&\rho'_R\big(U(X)_j, y\big) \coloneqq \rho_R\big(U(X)_j, y\big) \qquad\qquad \text{pour } y \in N(T) \setminus \{\alpha\}\\
&\rho'_F(x) \coloneqq \begin{cases} e & \text{si } \rho_F(x) = \alpha \\ \rho_F(x) & \text{sinon} \end{cases} \qquad\qquad \text{pour } \ x \in \mathrm{Var}
\end{aligned}
$$

On se réfèrera à la partie **Transformation de l'arbre** pour les valeurs de $\rho'_R(X, x)$ pour X un ensemble comme $\mathrm{Eq}_{i,j}$, $\mathrm{Pa}_{i,j}$ ou $\mathrm{Rel}^r_{x_1,\dots,x_{\#(r)}}$.

Lemme 1 Remarquons que si $T \vDash \varphi''$, en retirant tout les nœuds des ensembles $\mathrm{Rel}^r_{i_1,\dots,i_{\#(r)}}$ avec $\left(i_1, \dots, i_{\#(r)}\right) \neq (1, 2, \dots, \#(r))$, on a l'existence d'un T^* en forme normale qui satisfait φ''.

Plan de la preuve La première étape de la preuve consiste en une induction sur φ pour montrer que $m \vDash \varphi' \Rightarrow [m] \vDash \varphi'$ (**Lemme 2**). Puis la deuxième est de montrer par induction sur φ que pour un T en forme normale, $\lfloor T \rfloor \vDash \varphi' \Leftrightarrow \lfloor T \rfloor \vDash \varphi''$ (**Lemme 3**). Et finalement, on aura terminé :

- Si $T \vDash \varphi'$, on a par le **Lemme 2** et le fait que la relation $\to$ termine que sa forme normale satisfait φ' et par le **Lemme 3** qu'elle satisfait φ''.

- Si $T \vDash \varphi''$, on a par le **Lemme 1** qu'il existe une forme normale qui satisfait φ'' et par le **Lemme 3** qu'elle satisfait φ'.

Preuve du Lemme 2 On montre que $m \vDash \varphi' \Rightarrow [m] \vDash \varphi'$. Cette preuve est sensiblement la même que dans la longue induction de la preuve de moi au Chapitre 5.3, on se permettra donc d'aller plus vite. On procède par induction sur φ.

- Si $\varphi = "x \in Y"$ alors $m \vDash \exists x. \left(\bigvee_{i \in [k+1]} x \in Y_i \wedge x \in X_i \right)$. On pose $\psi(x) := "\left(\bigvee_{i \in [k+1]} x \in Y_i \wedge x \in X_i \right)"$, qui est locale. On choisi t le nœud de T qui permet de satisfaire $\psi(t)$.
  - Cas $t \neq \alpha$. Donc $[m][x \leftarrow t] \vDash \psi(x)$ et donc $[m] \vDash \exists x. \psi(x)$
  - Cas $t = \alpha$. e et α sont dans les mêmes ensembles : $[m][x \leftarrow e] \vDash \psi(x)$, et donc $[m] \vDash \exists x. \psi(x)$
- Si $\varphi = "R_i(x_1, ..., x_n)"$, alors

$$m \vDash \exists x, \bigvee_{i_1, ..., i_{r_i} \in [k+1]} x \in U_{i_1}(a_1) \wedge ... \wedge x \in U_{i_{r_i}}\left(a_{r_i}\right) \wedge \left(x \in \mathrm{Rel}^r_{i_1, ..., i_{r_i}} \right)$$

  On pose $\psi(x) := "\bigvee_{i_1, ..., i_{r_i} \in [k+1]} x \in U_{i_1}(a_1) \wedge ... \wedge x \in U_{i_{r_i}}\left(a_{r_i}\right) \wedge \left(x \in \mathrm{Rel}^r_{i_1, ..., i_{r_i}} \right)"$, qui est locale. On choisi t le nœud de T qui permet de satisfaire $\psi(t)$.
  - Cas $t \neq \alpha$. Donc $[m][x \leftarrow t] \vDash \psi(x)$ et donc $[m] \vDash \exists x. \psi(x)$
  - Cas $t = \alpha$. On a alors que $[m][x \leftarrow e'] \vDash x \in U_1(a_1) \wedge ... \wedge x \in U_{(r_i)}\left(a_{r_i}\right) \wedge \left(x \in \mathrm{Rel}^r_{1, ..., (r_i)} \right)$ car $e' \in \mathrm{Rel}^{\mathrm{r}}_{1, ..., \#(r)}$ et que pour chaque $0 < j \leq k+1$ comme $e \in U_{i_j}(a_j)$, par définition de ρ'_R, $e' \in U_j(a_j)$. Donc $[m][x \leftarrow e'] \vDash \exists x, \psi(x)$
- Si $\varphi = "\psi \wedge \xi"$, alors $\varphi' = \psi' \wedge \xi'$. On suppose $m \vDash \varphi'$. Donc $m \vDash \psi'$ et $m \vDash \xi'$. Donc $[m] \vDash \psi'$ et $[m] \vDash \xi'$, et finalement $[m] \vDash \psi'$.
  Le même raisonnement marche pour $\rightarrow$, $\neg$, $\vee$ et $\leftrightarrow$.
- Si $\varphi = "\exists Y. \psi"$ alors $m \vDash \exists Y_1, ..., \exists Y_{k+1}, \mathrm{set}(Y_1, ..., Y_{k+1}) \wedge \psi^*$. On choisi alors les ensembles $S_1, .., S_{k+1} \subseteq N(T)$ tel que $m[Y_1 \leftarrow S_1]...[Y_{k+1} \leftarrow S_{k+1}] \vDash \mathrm{set}(Y_1, ..., Y_{k+1}) \wedge \psi^*$. Pour chaque S_x, on pose $S'_x \subseteq N(T')$ tel que

$$S'_x = \left\{ e' \text{ si } \alpha \in S_{i_x} \right\} \cup \begin{cases} S_x & \text{si } \alpha \notin S_x \\ S_x \setminus \{\alpha\} \cup \{e, e''\} & \text{si } \alpha \in S_x \end{cases}$$

  On pose alors $\overline{m} := [m][Y_1 \leftarrow S'_1]...[Y_{k+1} \leftarrow S'_{k+1}]$, et on montre que $\overline{m} \vDash \mathrm{set}(Y_1, ..., Y_{k+1}) \wedge \psi^*$. On a déjà par hypothèse d'induction et car $m[Y_1 \leftarrow S'_1]...[Y_{k+1} \leftarrow S'_{k+1}] \vDash \psi^*$ le fait que $m \vDash \psi^*$, il nous suffit donc de montrer que $m \vDash \mathrm{set}(Y_1, ..., Y_{k+1})$. Il y a 2 formules à vérifier :
  - Cas $\varphi_1 = \forall x, P(x)$ avec P locale. Soit $t \in N(T')$. Quatre cas, mais qui sont en vrai de 2 types :
    - Si $t \in N(t)$, on a $\overline{m}[x \leftarrow t] \vDash P(t)$ car $m[x \leftarrow t] \vDash P(t)$.
    - Sinon, pour $t = e$, $t = e'$ ou $t = e''$, par définition de la transformation qui a été crée pour, les nœuds respectent P. On a donc $m[x \leftarrow t] \vDash P(t)$.
  - Cas $\varphi_2 = \forall x, y, \mathrm{parent}(x, y) \rightarrow P(x, y)$ avec P locale. Soit $p \in N(T')$ et f une fille de p.
    - Si $p, f \in N(T)$, on a $\overline{m}[x \leftarrow p][y \leftarrow s] \vDash P(x, y)$ car $m[x \leftarrow p][y \leftarrow s] \vDash P(x, y)$.
    - Si $p \notin N(T)$ et $f \in N(T)$, alors $p = e''$ et $f \in \{x_1, x_2\}$, et comme $m[x \leftarrow \alpha][y \leftarrow f] \vDash P(x, y)$ et que e'' à le même symbole que α, on au aussi $\overline{m}[x \leftarrow p][y \leftarrow f] \vDash P(x, y)$.
    - Si $p \in N(T)$ et $f \notin N(T)$, alors $f = \alpha$, et comme $m[x \leftarrow p][y \leftarrow \alpha] \vDash P(x, y)$ et que e à le même symbole que α, on au aussi $m[x \leftarrow \alpha][y \leftarrow s] \vDash P(x, y)$
    - Et par construction on a si $p, f \notin N(T)$ que $m[x \leftarrow \alpha][y \leftarrow s] \vDash P(x, y)$

- Si $\varphi =$ "$\exists y.\psi$" alors $m \vDash \exists Y_1, ..., \exists Y_{k+1}, \mathrm{Set}(Y_1, ..., Y_{k+1}) \wedge \psi^*$. On choisi alors les ensembles $S_1, .., S_{k+1} \subseteq N(T)$ tel que $m[Y_1 \leftarrow S_1]...[Y_{k+1} \leftarrow S_{k+1}] \vDash \mathrm{Set}(Y_1, ..., Y_{k+1}) \wedge \psi^*$. Pour chaque S_x, on pose $S'_x \subseteq N(T')$ tel que

$$S'_x = \left\{e' \text{ si } \alpha \in S_{i_x}\right\} \cup \begin{cases} S_x & \text{si } \alpha \notin S_x \\ S_x \setminus \{\alpha\} \cup \{e, e''\} & \text{si } \alpha \in S_x \end{cases}$$

  On pose alors $\overline{m} := [m][Y_1 \leftarrow S'_1]...[Y_{k+1} \leftarrow S'_{k+1}]$, et on montre que $\overline{m} \vDash \mathrm{Elem}(Y_1, ..., Y_{k+1}) \wedge \psi^*$. On a déjà par hypothèse d'induction et car $m[Y_1 \leftarrow S'_1]...[Y_{k+1} \leftarrow S'_{k+1}] \vDash \psi^*$ le fait que $m \vDash \psi^*$, il nous suffit donc de montrer que $m \vDash \mathrm{Elem}(Y_1, ..., Y_{k+1})$. Il y a 5 formules à vérifier, dont 2 déjà réalisés :
  - Les cas φ_3 et φ_4 sont sensiblement les mêmes que les cas φ_1 et φ_2, juste sur la réciproque.
  - Cas φ_5 : si $\alpha \in \bigcup_i S_i$, alors $e, e'' \in \bigcup_i S'_i$, qui assurera la connexion entre le parent de α et x_1 ou x_2. Si α n'y est pas, la connexion de $[m]$ découle directement de la connexion de $\bigcup_i S_i$ dans m.

En écrivant la quantification universelle comme la négation de la quantification existentielle, cela conclut la preuve.

Preuve du Lemme 3 On montre alors par induction sur φ que pour tout m un modèle en forme normale, $m \vDash \varphi' \Leftrightarrow m \vDash \varphi''$. Il y a sensiblement deux cas : celui ou φ est de la forme "$R_i\left(x_1, ..., x_{\#(r)}\right)$" et les autres :

- Si $\varphi =$ "$\psi \wedge \xi$", alors $\varphi' = \psi' \wedge \xi'$, et on a alors :

$$\begin{aligned} m \vDash \psi' \wedge \xi' &\Leftrightarrow m \vDash \psi' \text{ et } m \vDash \xi' \\ &\Leftrightarrow m \vDash \psi'' \text{ et } m \vDash \xi'' \\ &\Leftrightarrow m \vDash \psi'' \wedge \xi'' \\ &\Leftrightarrow m \vDash \varphi'' \end{aligned}$$

  Le même raisonnement ce fait pour $\rightarrow, \neg, \vee, \leftrightarrow$, mais aussi pour $\exists x$, $\forall x$, $\exists X$, $\forall X$ car ils ne font pas apparaître de relation. Pour guise d'exemple, on fait le cas $\exists x$: si $\varphi =$ "$\exists x, \psi$", alors $\varphi' =$ "$\exists X_1, ..., X_{k+1}, \mathrm{Elem}(X_1, ..., X_{k+1}) \wedge \psi'$". On a alors :

$$\begin{aligned} m \vDash \varphi' &\Leftrightarrow m \vDash \exists X_1, ..., X_{k+1}, \mathrm{Elem}(X_1, ..., X_{k+1}) \wedge \psi' \\ &\Leftrightarrow \text{Il existe } S_1, ..., S_n \text{ tel que } m[X_1 \leftarrow S_1]...[X_{k+1} \leftarrow S_{k+1}] \vDash \psi' \\ &\Leftrightarrow \text{Il existe } S_1, ..., S_n \text{ tel que } m[X_1 \leftarrow S_1]...[X_{k+1} \leftarrow S_{k+1}] \vDash \psi'' \\ &\Leftrightarrow m \vDash \varphi'' \end{aligned}$$

- Si $\varphi =$ "$x = y$", alors $\varphi' = \varphi''$, et donc $m \vDash \varphi' \Leftrightarrow m \vDash \varphi''$. Pareillement pour "$x \in X$".
- Si $\varphi =$ "$r\left(a_1, ..., a_{\#(r)}\right)$", alors $\varphi' =$ "$\exists x, \bigvee_{i_1, ..., i_{\#(r)} \in [k+1]} x \in U_{i_1}(a_1) \wedge ... \wedge x \in U_{i_{\#(r)}}(a_{\#(r)} \wedge \left(x \in \mathrm{Rel}^r_{i_1, ..., i_{\#(r)}}\right)$". Mais comme m est en forme normale, pour tout $\left(i_1, ..., i_{\#(r)}\right) \neq (1, 2, ..., \#(r))$, on a que $m \nvDash x \in \mathrm{Rel}^r_{i_1, ..., i_{\#(r)}}$ donc $m \nvDash \exists x, x \in U_{i_1}(a_1) \wedge ... \wedge x \in U_{i_{\#(r)}}\left(a_{\#(r)}\right) \wedge \left(x \in \mathrm{Rel}^r_{i_1, ..., i_{\#(r)}}\right)$. Autrement dit,

$m \vDash \varphi'$

$\Leftrightarrow m \vDash \exists x, \bigvee_{i_1,\dots,i_{\#(r)} \in [k+1]} x \in U_{i_1}(a_1) \wedge \dots \wedge x \in U_{i_{\#(r)}}(a_{\#(r)} \wedge \left(x \in \mathrm{Rel}^r_{i_1,\dots,i_{\#(r)}}\right)$

$\Leftrightarrow$ Il existe e tel que $m[x \leftarrow e] \vDash \bigvee_{i_1,\dots,i_{\#(r)} \in [k+1]} x \in U_{i_1}(a_1) \wedge \dots \wedge x \in U_{i_{\#(r)}}\left(a_{\#(r)}\right) \wedge \left(x \in \mathrm{Rel}^r_{i_1,\dots,i_{\#(r)}}\right)$

$\Leftrightarrow$ Il existe e tel que $m[x \leftarrow e] \vDash x \in U_1(a_1) \wedge \dots \wedge x \in U_{\#(r)}(a_{\#(r)} \wedge \left(x \in \mathrm{Rel}^r_{1,\dots,\#(r)}\right)$

$\Leftrightarrow m \vDash x \in \exists x, U_1(a_1) \wedge \dots \wedge x \in U_{\#(r)}(a_{\#(r)} \wedge \left(x \in \mathrm{Rel}^r_{1,\dots,\#(r)}\right)$

$\Leftrightarrow m \vDash \varphi''$

Ce qui conclut la preuve de ce lemme. □

8.3. Exemple de traduction de formule Mona

La formule $\exists a, \forall b, \mathrm{Edg}(a, b)$ qui s'écrit en Mona par `ex1 a: forall b: R(a,b)` se transforme pour une tree-width $k = 1$ en :

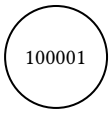
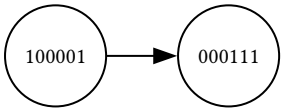
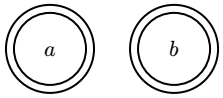
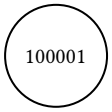
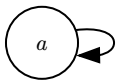
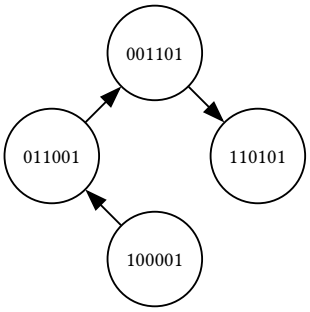
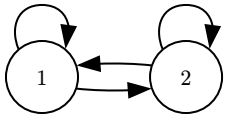
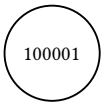
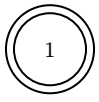
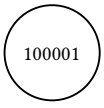
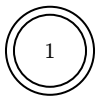
```
m2l-tree;
var2 Eq_1_2;
var2 Pa_1_1, Pa_1_2, Pa_2_1, Pa_2_2;
var2 RR_;

pred phi1(var2 x1, x2) =
    all1 t: (t in Eq_1_2 => (t in x1 <=> t in x2));
pred phi2(var2 x1, x2) =
    all1 t: all1 s: (s.1 = t | s.0 = t) => ((t in Pa_1_1 => (t in x1 <=> s in x1))
& (t in Pa_1_2 => (t in x1 <=> s in x2)) & (t in Pa_2_1 => (t in x2 <=> s in x1)) &
(t in Pa_2_2 => (t in x2 <=> s in x2)));
pred phi3(var2 x1, x2) =
    all1 t: ((t in x1 & t in x2) => t in Eq_1_2);
pred phi4(var2 x1, x2) =
    all1 t: all1 s: (s.1 = t | s.0 = t) => (((t in x1 & s in x1) => t in Pa_1_1) &
((t in x1 & s in x2) => t in Pa_1_2) & ((t in x2 & s in x1) => t in Pa_2_1) & ((t
in x2 & s in x2) => t in Pa_2_2));
pred connected(var2 X) =
    ex1 z: z in X & (all1 x: x in X => (x=z | x^ in X));
pred phi5(var2 x1, x2) =
    ex1 x: (x in x1|x in x2) & connected(x1 union x2);

pred Set(var2 x1, x2) = phi1(x1, x2) & phi2(x1, x2);
pred Elem(var2 x1, x2) = phi1(x1, x2) & phi2(x1, x2) & phi3(x1, x2) & phi4(x1, x2)
& phi5(x1, x2);

ex2 a2: ex2 a1: Elem(a1, a2) & (all2 b2: all2 b1: Elem(b1, b2) => (ex1 x: x in RR_
& x in a1 & x in b2));
```

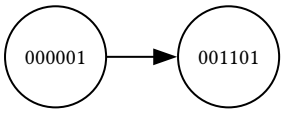
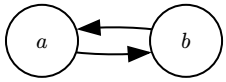
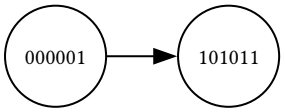
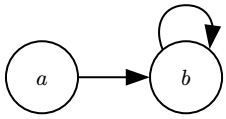
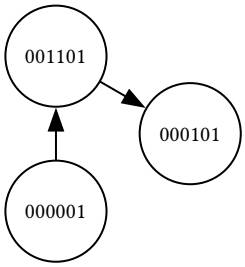
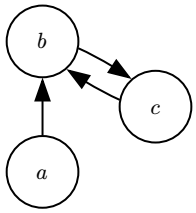
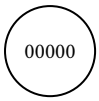
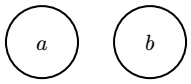
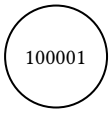
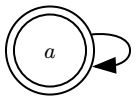
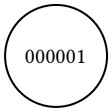
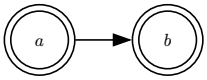
La vrai formule est sur les deux dernières lignes, le reste sont des prédicats (les φ_i, Elem et Set). Retraduite, elle donne

$$\exists\ \mathrm{A2}, \exists\ \mathrm{A1}, \mathrm{Elem}(\mathrm{A1}, \mathrm{A2}) \wedge \Big(\forall\ \mathrm{B2}, \forall\ \mathrm{B1}, \mathrm{Elem}(\mathrm{B1}, \mathrm{B2}) \rightarrow \Big(\exists x, x \in \mathrm{R}_{1,2}^{\mathrm{Edg}} \wedge x \in \mathrm{B1} \wedge x \in \mathrm{A2}\Big)\Big)$$

8.4. Tableau de tous les tests

Les tests de formule satisfaisable:

Formule	Temps	Output Mona	Modèle
$\exists a, \exists b, \mathrm{Edg}(a, b)$	PC : crash Serveur : crash		
$\forall a, \mathrm{P}(a)$	PC : 0.05s Serveur: 0.08s		
$\exists a, \exists b, a \neq b \wedge$ $\mathrm{Edg}(a, b)$	PC : crash Serveur : crash		
$\exists a, \exists b, \mathrm{P}(a) \wedge \mathrm{P}(b) \wedge$ $a \neq b$	PC : 58.22s Serveur: 63.37s		
$\exists a, \exists b, \exists c, \mathrm{P}(a) \wedge$ $\mathrm{P}(b) \wedge \mathrm{P}(c) \wedge a \neq b$	PC : crash Serveur : crash		
$\exists a, \forall b, \mathrm{Edg}(a, b)$	PC : 0.20s Serveur: 0.36s		
$(\forall a, \forall b, \mathrm{Edg}(a, b))$ $\wedge (\exists a, \exists b, a \neq b)$	PC : 2.24s Serveur : 2.58s		
$\exists a, \mathrm{P}(a) \wedge (\forall b, b = a)$	PC : 0.08s Serveur : 0.12s		
$\exists a, \forall b, b = a$	PC : 0.06s Serveur : 0.08s		

Formule	Temps	Output Mona	Modèle
$\exists a, \exists b, \exists c, \mathrm{Edg}(a,a) \wedge \mathrm{Edg}(c,a) \wedge \mathrm{Edg}(b,a) \wedge a \neq b \wedge b \neq c \wedge a \neq c$	PC : crash Serveur : crash		
$(\forall a, \exists b, \mathrm{Edg}(a,b)) \wedge (\forall a, \neg\, \mathrm{Edg}(a,a))$	PC : 4.53s Serveur : 5.03s		
$(\forall a, \exists b, \mathrm{Edg}(a,b)) \wedge (\exists a, \forall b, \neg\, \mathrm{Edg}(b,a))$	PC : 17.50s Serveur : 19.08s		
$(\forall a, \exists b, \mathrm{Edg}(a,b)) \wedge (\exists a, \forall b, \neg\, \mathrm{Edg}(b,a)) \wedge (\forall a, \neg\, \mathrm{Edg}(a,a))$	PC : 18.56s Serveur : 20.06s		
$\exists Y, (\exists a, a \in Y) \wedge (\exists b, b \notin Y)$	PC : 12.52s Serveur : 13.53s		
$\exists A, \forall a, a \in A \wedge \mathrm{P}(a)$	PC : crash Serveur : 3m 51s		
$(\forall x, \exists y, \mathrm{Edg}(x,y)) \rightarrow (\exists y, \forall x, \mathrm{Edg}(x,y))$	PC : 4.29s Serveur : 4.70s		
$(\exists y, \forall x, \mathrm{Edg}(x,y)) \rightarrow (\forall x, \exists y, \mathrm{Edg}(x,y))$	PC : 4.01 Serveur : 4.79s	Tautologie, tout modèle marche.	

Les tests de formules non satisfiable:

Les tests prouvé instatisfiable par mona sont :

Formule	Temps	Formule	Temps
$\exists x, \mathrm{Edg}(x,x) \wedge \neg\, \mathrm{Edg}(x,x)$	PC : 0.01s Serveur : 0.01s	$\exists x, \neg(\mathrm{Edg}(x,x)) \wedge (\forall y, \mathrm{Edg}(x,y))$	PC : 0.05s Serveur : 0.08s

Formule	**Temps**	**Formule**	**Temps**
$\exists x, \mathrm{Edg}(x,x) \wedge (\forall y, \neg(\mathrm{Edg}(x,y))$	PC : 0.34s Serveur : 0.56s	$\exists a, \exists b, \exists c, a \neq b \wedge b \neq c \wedge c \neq a \wedge \mathrm{Edg}(a,b) \wedge \mathrm{Edg}(b,c) \wedge \mathrm{Edg}(c,a)$	PC : crash Serveur : crash
$\exists a, \exists b, a = b \wedge a \neq b$	PC : 0.01s Serveur : 0.01s	$(\exists y, \forall x, \mathrm{Edg}(x,y)) \wedge \neg(\forall x, \exists y, \mathrm{Edg}(x,y))$	PC: 4.05s Serveur : 4.57s