Décompositions arborescentes

Bruno Courcelle
Université Bordeaux 1, LaBRI (CNRS),
courcell@labri.fr

October 26, 2005

Abstract

1 Introduction

AVERTISSEMENT : Ceci est une version provisoire. Les références aux chapitres A,B,C,D etc... concernent d'autres chapitres de l'ouvrage collectif auquel ce texte est destiné.

Ce chapitre est une introduction à la notion de décomposition arborescente d'un graphe, notion qui fait l'objet d'une recherche intense comme le montre le grand nombre de communications à son sujet présentées dans les colloques de théorie des graphes et d'algorithmique.

Les décompositions arborescentes interviennent de façon essentielle dans plusieurs domaines : pour la construction d'algorithmes polynomiaux, dans les caractérisations de classes de graphes par des *mineurs interdits*, dont le paradigme est la caractérisation de Kuratowski des graphes planaires, dans la caractérisation des ensembles de graphes, et plus généralement, de structures relationnelles pour lesquelles certains langages logiques et notamment la *Logique du Second-Ordre Monadique* sont décidables, et enfin dans la théorie des Grammaires de Graphes.

Ces quatre directions de recherche ont été initiées indépendamment et ont convergé en mettant en évidence les rôles centraux des décompositions de graphes et de la logique du second-ordre monadique. Esquissons l'historique de ces différentes approches.

Beaucoup de problèmes NP-complets ont des algorihmes polynomiaux pour des classes particulières de graphes. Les classes de *graphes structurés*, c'est à dire de graphes que l'on peut décrire comme des combinaisons finies de graphes de base (en nombre fini) au moyen d'un nombre fini d'opérations de "recollements de graphes" c'est à dire de concaténation généralisée aux graphes sont vite

apparues comme très propices à cette recherche. Le traitement d'exemples tels que la classe des graphes série-parallèle a permis de dégager la notion de k-arbre partiel, c'est à dire en fait, sous un autre nom celle de graphe de largeur arborescente au plus k. La bibliographie publiée en 1994 par S. Hedetniemi [Hedet] comporte 238 entrées : elle reflète bien les débuts de cette algorithmique des graphes structurés. Sa mise à jour comporterait sans doute un millier de titres. (A préciser).

La deuxième approche est celle de Robertson et Seymour. Ils ont commencé vers 1980 un travail de longue haleine qui a débouché sur la preuve du Théorème des Mineurs de Graphes. Ce théorème énonce que toute famille de graphes fermée par passage aux mineurs est caractérisée, telle la famille des graphes planaires, par un nombre fini de mineurs exclus. C'est de leur analyse de la notion de mineur que proviennent les notions de *décomposition et de largeur arborescente*s, ainsi que des variantes *linéaires* de ces notions.

La troisième approche consiste à tenter de caractériser les classes de graphes pour lesquelles *la Logique du Second-Ordre Monadique est décidable*, et à montrer, que en fin de compte seules des classes d'arbres ou de structures qui en sont proches possèdent cette propriété de décidabilité. D. Seese a montré ([Seese]) qu'une telle classe de graphes planaires est de largeur arborescente bornée, et donc que, dans ce cadre, la largeur arborescente est le bon paramètre caractérisant la proximité d'une classe de graphes à une classe d'arbres. Sa preuve s'appuie sur les résultats de Robertson et Seymour. Ce résultat a fait l'objet de nombreux développements par Courcelle et Oum ([Cou15,CouOum]).

Enfin, la quatrième approche consiste à étendre aux ensembles de graphes les descriptions des langages (ensembles de mots) et des ensembles d'arbres par des grammaires et des automates qui sont l'objet de la *Théorie des Langages Formels*. En particulier, l'une des deux familles de grammaires "context-free" de graphes est intrinsèquement liée aux décompositions arborescentes.

Ce chapitre est centré sur les approches des décompositions arborescentes issues des grammaires de graphes et de la théorie des mineurs, ainsi que leurs applications algorithmiques. Pour des exposés plus détaillés, on recommandera le livre de Downey et Fellows [DF] et l'ouvrage collectif [Handbook].

2 Décompositions arborescentes

2.1 Définitions

Les notions de décomposition arborescente et de largeur arborescente sont données pour les graphes non orientés. Elles s'appliquent aux graphes orientés par l'intermédiaire de leurs graphes nonorientés sous-jacents. Les principales

références sont le livre de Downey et Fellows [DF] et l'article de synthèse de Bodlaender [Bod1].

Définition 2.1.1 : Décomposition arborescente

Soit $G = (X, E)$ un graphe dont X est l'ensemble des sommets et E l'ensemble des arêtes. Une *décomposition arborescente* (*tree-decomposition* en anglais) de G est un couple (T, f) tel que T est un arbre, $T = (N, A)$, et f est une application qui associe à tout noeud n de T un ensemble $f(n) \subseteq X \cup E$, et qui satisfait les conditions suivantes :

(1) Tout sommet de G appartient à quelque $f(n)$,

(2) Toute arête de G appartient à un et un seul ensemble $f(n)$,

(3) Si une arête appartient à $f(n)$, ses extrémités appartiennent aussi à $f(n)$,

(4) Pour tout sommet x, l'ensemble des noeuds n tels que $x \in f(n)$ est connexe dans T.

(On utilisera le terme "noeud" pour désigner les sommets des arbres. Cette convention est commode lorsque l'on parle à la fois d'un graphe et d'un arbre qui représente sa structure.)

Les ensembles $f(n)$ sont les *boîtes* de la décomposition (T, f). Sa *largeur* est l'entier :

$wd(T, f) = Max\{Card(X \cap f(n)) \mid n \in N\} - 1$.

La *largeur arborescente de* G (*tree-width*) est la plus petite largeur d'une décomposition arborescente de G. Elle sera notée $twd(G)$, notation qui fait référence à la terminologie anglaise. Une décomposition arborescente est *optimale* si sa largeur est la plus petite possible pour le graphe considéré.

Une décomposition arborescente peut ne comporter qu'une boîte qui contient alors tout le graphe. Il en résulte que $twd(G) \leq Card(X) - 1$. Les arbres sont de largeur arborescente 1 et les graphes dont toutes les arêtes sont des boucles sont de largeur arborescente 0. Le cycle C_m : $1 - 2 - 3 - ... - m - 1$ (dont les sommets sont $1, 2, 3, ..., m$; les adjacences sont indiquées par $-$) a une largeur arborescente 2 si $m \geq 3$. Il admet comme décomposition arborescente (T, f) où T est l'arbre (en fait le chemin) $1 - 2 - 3 - ... - (m-1)$ et $f(i) \cap X = \{1, i, i+1\}$pour $i = 1, ..., m-1$. Le lecteur précisera sans peine les ensembles $f(i) \cap E$. Cela montre que $twd(C_m) \leq 2$. On a $twd(C_m) = 2$ si $m \geq 3$, car il n'en existe pas de décomposition de largeur 1. Cela sera montré plus loin mais le lecteur pourra le vérifier directement pour se familiariser avec la définition.

Interprétation intuitive

Soit (T, f) une décomposition arborescente de G. Pour chaque noeud n de T soit $G(n)$ le sous-graphe de G constitué des sommets et des arêtes qui appartiennent à $f(n)$. Le graphe G est ainsi partitionné en sous-graphes sans arêtes communes, et ces sous-graphes sont "recollés" au niveau de leurs sommets communs, selon la configuration globale de l'arbre T. La condition (4) de

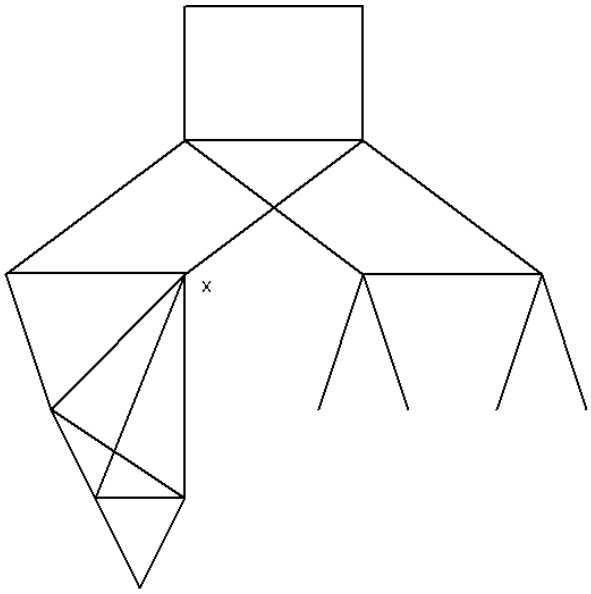

Figure 1: Un graphe G

la définition montre que si $G(n)$ et $G(n')$ ont des sommets en commun et donc sont recollés au niveau de ces sommets, alors, ils sont recollés par ces sommets à tous les graphes associés aux noeuds du chemin dans l'arbre qui relie n à n'.

Les figures 1, 2 et 3 montrent un graphe G et une décomposition arborescente (T, f) de ce graphe. Sur la figure 3, les boîtes de la décomposition arborescente sont représentées comme des graphes disjoints (traits pleins). Les arcs en pointillés relient les sommets de ces graphes qui sont à fusionner, de manière à recoller les graphes induits par les boîtes de la décomposition. Le sommet x du graphe G appartient aux boîtes b, c, d, e, lesquelles forment une partie connexe de T, conformément à la condition (4). Sur la figure 3, les sommets $(x, b), (x, c), (x, d), (x, e)$ de ces quatre boîtes correspondent à ce sommet x. Le graphe G est obtenu par contraction des arêtes en pointillés du graphe de la figure 3. (La contraction d'arête est définie dans la section 2.3)

Décomposition arborescentes linéaires

Si dans la définition d'une décomposition aborescente, on impose à l'arbre T d'être un chemin, on obtient la notion de *décomposition arborescente linéaire* (*path-decomposition*) et celle de *largeur arborescente linéaire* (*path-width*), notée $pwd(G)$. La décomposition proposée plus haut des cycles C_m est une décomposition arborescente linéaire et l'on a $pwd(C_m) = 2$ pour $m \geq 3$. Puisque tout chemin est un arbre, on a $twd(G) \leq pwd(G)$ pour tout graphe G. Les arbres sont de largeur arborescente 1 mais de largeur arborescente linéaire non bornée.

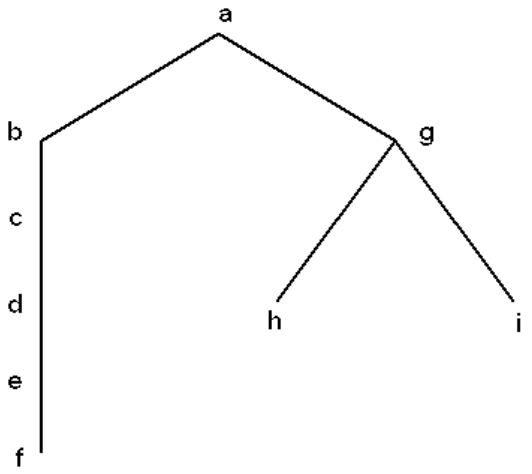

Figure 2: L'arbre T d'une décomposition arborescente de G

On désignera par $TWD(\leq k)$ et $PWD(\leq k)$ les ensembles des graphes de largeur arborescente au plus k et, respectivement, de largeur arborescente linéaire au plus k.

Quelques exemples

On a pour des graphes classiques les valeurs suivantes : $twd(K_m) = pwd(K_m) = m-1$; $twd(P_m) = pwd(P_m) = 1$; $twd(T_n) = 1$, $pwd(T_n) = \lceil n/2 \rceil$ (**à vérifier**) où T_n est l'arbre binaire complet de hauteur n, dont l'ensemble des noeuds est $\{0, 1, ..., 2^n - 2\}$, et les arêtes sont les paires $(i, 2i+1)$ et $(i, 2i+2)$ pour $i = 0, ..., 2^{n-1} - 2$.

On notera $G_{n\times m}$ la grille carrée définie comme le produit cartésien $P_n \times P_m$. Ses sommets sont les couples (i, j) pour $1 \leq i \leq n, 1 \leq j \leq m$ et ses arêtes relient (i, j) et (i', j') si et seulement si $\mid i - i' \mid + \mid j - j' \mid = 1$. Sauf si $n = m = 1$, on a $twd(G_{n\times m}) = pwd(G_{n\times m}) = Min\{n, m\}$. [Bod1].

Si G est un graphe planaire extérieur (voir ce livre, chapitre A) alors $twd(G) \leq 2$.(Exercice 1)

Dans tous ces cas, il n'est pas très difficile de construire des décompositions qui établissent les majorations. Il est plus difficile de montré qu'une décomposition est optimale. Nous verrons plus loin quelques méthodes pour ce faire.

Variantes des définitions

On n'a pas imposé que chaque boîte est non vide. Mais toute décomposition arborescente (T, f) d'un graphe non vide peut être transformée en une décom-

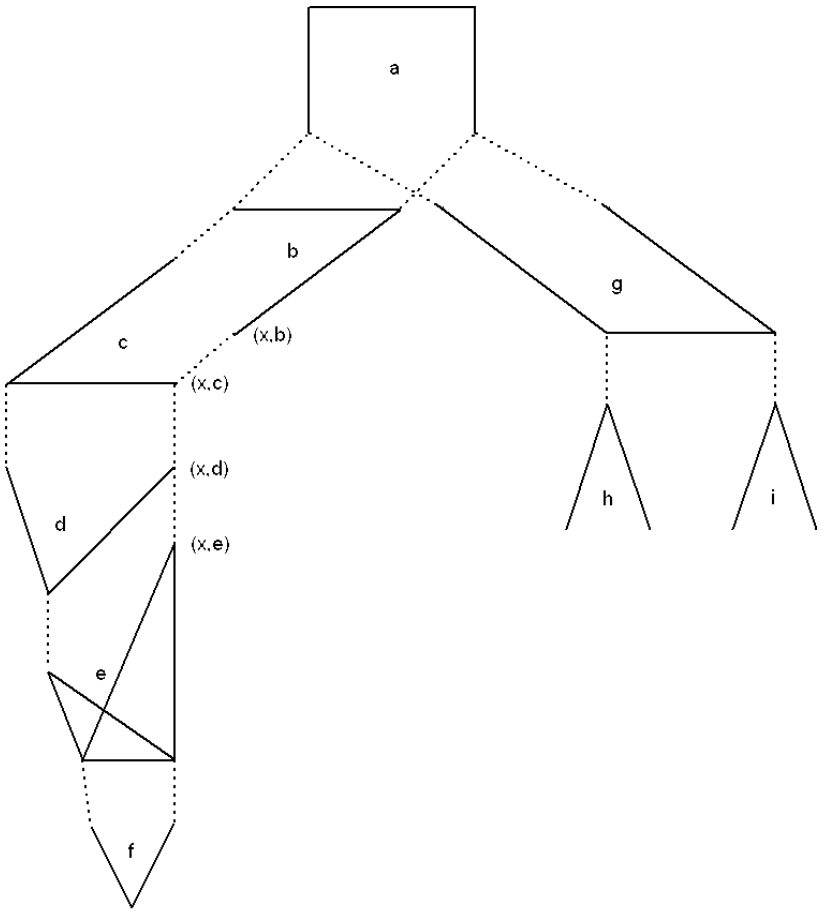

Figure 3: Une décomposition arborescente de G. Le graphe G est obtenu par contraction des arêtes en pointillé.

position arborescente de même largeur et du même graphe, dont aucune boîte n'est vide. En effet, si $f(n)$ est vide, si m est un noeud voisin de n, il suffit de fusionner ces noeuds (en contractant l'arête qui les relie, une définition formelle est donnée plus bas) et en prenant $f(m)$ comme boîte associée au noeud résultant de cette fusion. On vérifie sans peine que l'on obtient une décomposition arborescente du même graphe et de même largeur. On répète la transformation autant de fois que nécessaire.

D'autre part, on peut remplacer la condition (2) par la condition :

(2') Toute arête de G appartient à au moins un ensemble $f(n)$.

Dans ce cas les graphes $G(n)$ peuvent avoir des arêtes en commun. Ils constituent un recouvrement et non plus une partition du graphe G.

On peut également définir une décomposition arborescente (T, f) en demandant que $f(n) \subseteq X$ et en remplaçant les conditions (2) et (3) par l'unique condition que toute arête a ses extréminités dans une même boîte. On définit alors $G(n)$ comme le sous-graphe induit $G[f(n)]$.

Ces variantes, qui peuvent être utiles pour faciliter certaines constructions ne modifient pas les notions de largeur arborescente et de largeur arborescente linéaire.

Décompositions arborescentes orientées

Une décomposition arborescente (T, f) d'un graphe G est dite *orientée* si l'arbre T est muni d'une racine, et est orienté de telle sorte tout noeud soit accessible depuis la racine par un unique chemin orienté. La racine est de degré entrant 0. On notera $\leq_T$ la relation d'ordre telle que $x \leq_T y$ si et seulement si $x = y$ ou bien il existe un chemin orienté de x à y. Il résulte alors de la condition (4) de la Définition 2.1.1 que pour tout sommet x de G, il existe un unique noeud de T le plus proche possible de la racine, donc minimal pour $\leq_T$ (nous dirons le plus haut dans l'arbre, car nous dessinons les arbres avec la racine en haut, comme sur la figure 2) dont la boîte associée contient x. On dira que ce noeud est le *noeud d'introduction de* x et on le notera $NI(x)$. Avec ces définitions et notations :

Lemme 2.1 : Si deux sommets x et y de G sont adjacents, alors on a $NI(x) \leq_T NI(y)$ et $x \in f(NI(y))$ ou l'inverse, en échangeant les rôles de x et y.

On peut avoir $NI(x) = NI(y)$ et alors, $x \in f(NI(y))$ et $y \in f(NI(x))$.

Preuve : Soit n un noeud dont la boîte contient x et y. On a $n \geq_T NI(x)$ et $n \geq_T NI(y)$ et donc, $NI(x)$ et $NI(y)$ sont comparables pour $\leq_T$. Si $NI(x) \leq_T NI(y)$ alors $NI(y)$ est sur le chemin dans T de $NI(x)$ à n, et donc $x \in f(NI(y))$ d'après la condition (4) de la définition 2.1.1. □

Un *ordre d'élimination de type k pour un graphe simple* G est une énumération $x_1, ..., x_n$ de l'ensemble de ses sommets telle que pour tout i, l'ensemble des indices j tels que $j < i$ et $x_i - x_j$ a au plus k éléments.

Proposition 2.1.1 : Un graphe simple de largeur arborescente au plus k possède un ordre d'élimination de type k.

Preuve : Soit (T, f) une décomposition arborescente d'un graphe simple $G = (X, E)$, orientée et de largeur au plus k. Soit $(r = n_0, n_1, ..., n_p)$ un *tri topologique* de T (c'est à dire un ordre total $\leq'_T$ qui étend $\leq_T$). Pour tout $i = 0, ..., p$, soit X_i l'ensemble des sommets de G qui sont dans $f(n_i)$ mais dans aucune boîte $f(n_j)$ pour $j < i$. C'est l'ensemble des sommets "introduits à l'étape i" si on envisage $n_0, n_1, ..., n_p$ comme une suite d'étapes d'ajout de sommets et d'arêtes. On a $X_i = NI^{-1}(n_i)$. On choisit une énumération quelconque $\overrightarrow{X_i}$ de X_i et l'on prend la concaténation $\overrightarrow{X_0}... \overrightarrow{X_p}$ comme l'énumération cherchée de X. Soit x un sommet de G et y un sommet adjacent, situé avant x. Donc $NI(y) \leq_T NI(x)$ et $y \in f(NI(x))$ (Lemme 2.1). Il en résulte que ces sommets y sont en nombre au plus k puisque $f(NI(x))$ a au plus $k+1$ éléments et contient x. On a donc bien construit un ordre d'élimination de type k. □

Corollaire 2.1 : (1) Un graphe de largeur arborescente au plus k est $(k+1)$-coloriable. Pour toute décomposition arborescente de ce graphe de largeur au plus k, on peut choisir une coloration telle que deux sommets d'une même boîte sont de couleurs différentes.

(2) Un graphe simple et sans boucle à n sommets, de largeur arborescente au plus k a au plus $nk - k(k-1)/2$ arêtes.

Preuve : (1) Soit $G = (X, E)$ un graphe de largeur arborescente au plus k muni d'un ordre d'élimination $x_1, ..., x_n$ construit par la Proposition 2.1.1 à partir d'une décomposition (T, f). On utilise $C = [k+1]$ comme ensemble des couleurs. On définit un coloriage $c : X \longrightarrow C$ de la façon suivante : $c(x_1) = 1$, et pour $i > 1$, $c(x_i)$ est la plus petite couleur qui diffère de $c(y)$ pour tout sommet y dans $f(NI(x_i))$ qui précède x_i. Il existe toujours dans C une couleur disponible pour définir $c(x_i)$ puisque la décomposition donnée est de largeur au plus k et donc que $f(NI(x_i))$ a au plus $k+1$ éléments. Deux sommets d'une même boîte sont donc de couleurs différentes. Deux sommets adjacents sont dans une même boîte et sont donc de couleurs différentes.

(2) Vérification immédiate en utilisant un ordre d'élimination de type k.□

Remarque : Le corollaire 2.1 ne fournit pas nécessairement des colorations optimales. Pour un cycle C_{2n}, 2-coloriable et de largeur arborescente 2, il fournit une 3-coloration. En utilisant un ordre d'élimination de type 2 et en définissant pour $c(x_i)$ la plus petite couleur différente de $c(y)$ pour y adjacent à x_i et avant lui, on obtient une 2-coloration de C_{2n}, et en général, une coloration utilisant moins ou autant de couleurs que pour celle construite par la preuve du Corollaire 2.1.

2.2 Passage au sous-graphe

Lorsqu'un graphe est transformé en un autre, ou contient un autre graphe, comment se comparent leur largeurs arborescentes ? Nous allons donner quelques réponses qui fourniront des outils pour obtenir des encadrements et parfois même des valeurs exactes les largeurs arborescentes de certains types de graphes.

Proposition 2.2.1 : Si H est obtenu à partir de G au moyen d'ajouts et/ou de suppressions de boucles et/ou d'arcs doubles, alors $twd(H) = twd(G)$ et $pwd(H) = pwd(G)$.

Preuve : Il suffit de montrer que la largeur arborescente est invariante l'ajout d'une boucle ou le doublage d'un arc existant.

Si H est obtenu à partir de G en ajoutant une boucle incidente à un sommet x (respectivement en doublant une arête $e : x - y$) et si (T, f) est une décomposition arborescente de G on obtient une décomposition de H en ajoutant cette boucle à une boîte qui contient x (respectivement, en ajoutant la nouvelle arête à une boîte qui contient x et y. On transforme une décomposition arborescente de H en une de G en supprimant la boucle ou l'arête en question. Dans les deux cas, les décompositions optimales de G et H ont mêmes largeurs.□

Il en résulte que l'étude de la largeur arborescente peut se concentrer sur les graphes simples, c'est à dire sans boucle ni arc multiple.

Proposition 2.2.2 : (1) Si H est un sous-graphe de G alors $twd(H) \leq twd(G)$ et $pwd(H) \leq pwd(G)$.

(2) Si H est obtenu en supprimant de G une arête e, ou bien un sommet x et toutes les arêtes incidentes, alors $twd(H) \leq twd(G) \leq twd(H) + 1$ et $pwd(H) \leq pwd(G) \leq pwd(H) + 1$.

Preuve : (1) On construit à partir d'une décomposition (T, f) de G une décomposition de H en supprimant des éléments des ensembles $f(n)$ et la largeur de la décomposition ne peut que diminuer.

(2) Inversement, soit à construire une décomposition (T, f) de G à partir d'une décomposition (T, g) de H : soit y un sommet de l'arête e ou bien le sommet x. On obtient g en ajoutant y à chaque boîte de (T, f), ainsi que, dans les boîtes apprpriées, l'arête e ou les arêtes incidentes à x. □

Composantes biconnexes

La notion de composante biconnexe est vue dans le chapitre B de ce livre. Si un graphe est connexe, on peut définir un arbre dont les noeuds correspondent à ses composantes biconnexes et dont toute arête $A - B$ correspond à un sommet commun aux composantes A et B. On dira que cet arbre est *un arbre des*

composantes biconnexes de G. Cet arbre n'est pas défini de manière unique. Par exemple, si G est la réunion de 3 composantes biconnexes A, B, C qui ont un sommet en commun, on peut prendre comme arbre : $A - B - C$ ou $B - A - C$ ou $A - C - B$.

Proposition 2.2.3 : La largeur arborescente d'un graphe est le maximum des largeurs arborescentes de ses composantes biconnexes.

Preuve : Soit G un graphe. Puisque toute composante biconnexe C est un sous-graphe, on a $twd(C) \leq twd(G)$ et donc le maximum des $twd(C)$ est au plus $twd(G)$.

Pour démontrer l'inverse considérons les composantes biconnexes $C_1, ..., C_n$ et pour chacune d'elles, une décomposition arborescente optimale (T_i, f_i). Nous allons les regrouper en une décomposition arborescente de G. On suppose d'abord G connexe. Soit $U = (\{C_1, ..., C_n\}, A)$ un arbre des composantes biconnexes de G.

On prend l'union disjointe des arbres T_i (c'est à dire que certains des arbres T_i sont remplacés si besoin par des copies isomorphes disjointes de toutes les autres). A chaque arête $C_i - C_j$ de U correspond un sommet x de G. Soit u un noeud de T_i et v un noeud de T_j tels que $x \in f_i(u)$ et $x \in f_j(v)$. On crée une arête entre u et v. En faisant ainsi pour chaque arête de U on obtient un arbre T dont chaque T_i est un sous-arbre. On prend comme fonction f l'extension commune des fonctions f_i. Les conditions (1)-(3) de la définition 2.2.1 sont clairement vérifiées. Pour vérifier la condition (4) on observe que lorsqu'un sommet x est commun à plusieurs composantes C_i, l'ensemble des arêtes $C_i - C_j$ de U auxquelles il correspond forme un sous-arbre de U. D'autre part, dans chaque T_i l'ensemble A_i des noeuds n tels que $x \in f_i(n)$ est connexe. L'union de ces ensembles A_i forme donc dans T un unique ensemble connexe. □

La proposition qui suit permettra de minorer les largeurs arborescentes de certains graphes.

Proposition 2.2.4 : Soit (T, f) une décomposition arborescente d'un graphe $G = (X, E)$.

(1) Si Y est une partie de X qui induit un sous-graphe complet, il existe un noeud n de T tel que $Y \subseteq f(n)$.

(2) Si Y et Z sont deux parties disjointes de X telles $Y \otimes Z$ est un sous-graphe de G, alors il existe un noeud n de T tel que $Y \subseteq f(n)$ ou $Z \subseteq f(n)$.

(3) Pour tous m, n, $twd(K_m) = m - 1$, et $twd(K_{n,m}) = Min\{n, m\}$, et, si $n \geq 3$, $twd(C_n) = 2$.

On désigne par $Y \otimes Z$ le graphe formé de tous les arcs entre Y et Z, ensembles disjoints de sommets.

Preuve : (1) On rappelle (chapitre C, et exercice 3) que les parties connexes de l'ensemble des noeuds d'un arbre satisfont la *propriété de Helly* :

Si $A_1, ..., A_m$ sont des ensembles deux à deux d'intersection non vide, il existe un élément commun à tous ces ensembles.

Donc si (T, f) est une décomposition arborescente d'un graphe $G = (X, E)$ et si $Y = \{y_1, ..., y_m\} \subseteq X$ induit un sous-graphe complet, alors les ensembles A_i des noeuds p tels que $y_i \in f(p)$ sont connexes et deux à deux d'intersection non vide d'après les clauses (3) et (4) de la définition 2.1.1. Donc il existe un noeud n, commun à tous les A_i et donc $Y \subseteq f(n)$.

(2) Supposons que $Y = \{y_1, ..., y_m\} \subseteq X$, et que y_1 et y_2 ne sont pas dans une même boîte. Les ensembles A_1 et A_2 définis comme ci-dessus sont disjoints. Il existe un unique noeud n_1 dans A_1 et un unique noeud n_2 dans A_2 à distance minimale l'un de l'autre dans l'arbre T. Soit z quelconque dans Z. Il existe m_1 dans A_1 et m_2 dans A_2 tels que $f(m_1)$ et $f(m_2)$ contiennent respectivement les arêtes $y_1 - z$ et $y_2 - z$ et donc contiennent z. Il en résulte que z appartient à toute boîte $f(p)$ pour tout noeud p de l'unique chemin dans T de m_1 à m_2. Or ce chemin contient le chemin de n_1 à n_2. Ainsi Z est contenu dans $f(p)$ pour tout p du chemin de n_1 à n_2.

Donc si Z n'est pas contenu dans une boîte, cela veut dire que tout couple d'éléments de Y est contenu dans une même boîte. Il en résulte que Y est contenu dans une même boîte, en utilisant la propriété de Helly, comme pour prouver l'assertion (1).

(3) On a $twd(K_m) \geq m - 1$ d'après (1) et donc l'égalité. L'assertion (2) donne $twd(K_{n,m}) \geq Min\{n, m\} - 1$. Mais il n'est pas difficile de voir que K_{n+1} est mineur de $K_{n,m}$ si $n \leq m$ (la notion de mineur est définie dans la section suivante). On en déduit, avec la Proposition 2.3.2 que $twd(K_{n,m}) \geq twd(K_{Min\{n,m\}+1}) = Min\{n, m\}$. On a $twd(K_{n,m}) = Min\{n, m\}$. De même, K_3 est mineur de C_n si $n \geq 3$. On en déduit que $twd(C_n) = 2$. □

Les notions de décomposition et de largeur arborescentes s'étendent de façon très naturelle aux hypergraphes : les hyperarêtes qui généralisent les arêtes sont des ensembles de sommets de taille non limitée à 2. Il suffit de remplacer la condition (3) de la Définition 2.1.1 par la condition qui impose que si une hyperarête est dans une boîte, alors tous ses sommets le sont aussi. D'autre part, on peut associer à un hypergraphe H un graphe $K(H)$ qui a les mêmes sommets et une arête entre deux sommets si et seulement si ces deux sommets appartiennent à une même hyperarête. Il résulte de la Proposition 2.2.4(1) que H et $K(H)$ ont les mêmes décompositions arborescentes et donc la même largeur arborescente.

Certaines opérations sur les graphes sont difficilement compatibles avec la largeur arborescente. Notons $s(G)$ le nombre de sommets d'un graphe G, et $Diam(G)$ son *diamètre*, défini comme la longueur maximum d'un chemin induit. Pour deux graphes disjoints G et H, on note $G \otimes H$ leur union augmentée de toutes les arêtes reliant un sommet de G et un sommet de H. Des propositions 2.2.2 et 2.2.4(3) il résulte que (voir aussi l'exercice 1) :

$$Min\{s(G), s(H)\} \leq twd(G \otimes H) \leq Min\{s(G) + twd(H), s(H) + twd(G)\}.$$

L'opération de *produit cartésien* de deux graphes $G = (X, A)$ et $H = (Y, B)$ leur associe le graphe $G \times H$ dont $X \times Y$ est l'ensemble des sommets et où $(x, y) - (u, v)$ si et seulement si $x = u$ et $y - v$ (dans H) ou bien $y = v$ et $x - u$ (dans G). Ainsi $G_{n\times m} = P_n \times P_m$. On a donc (exercice 1) :

$$Min\{Diam(G) + 1, Diam(H) + 1\} \leq twd(G \times H) \leq$$
$$Min\{(s(G) + 1).twd(H), (s(H) + 1).twd(G)\}.$$

Pour des majorations fines, on consultera [Dje]. Ainsi, on ne peut pas majorer la largeur arborescente de $G \otimes H$ ni celle de $G \times H$ en fonction des largeurs arborescentes de G et de H. On a observé une situation contraire pour les composantes biconnexes.

2.3 Mineurs

L'étude des mineurs des graphes est l'une des origines et des motivations de la notion de décomposition arborescente. Nous allons examiner les liens entre ces différentes notions.

Définition 2.3.1 : Mineur d'un graphe

On définit d'abord la notion de *graphe quotient*. Soit $G = (X, E)$ un graphe et $\approx$ une relation d'équivalence sur X. Le *graphe quotient* $G/\approx$ est défini comme le graphe $(X/\approx, E)$ avec une fonction d'incidence telle que si l'on a $e : x - y$ dans G (ce qui veut dire "l'arête e relie les sommets x et y") alors $e : [x]_\approx - [y]_\approx$ dans $G/\approx$.

Soit $G = (X, E)$ un graphe et $F \subseteq E$. Le graphe obtenu par *contraction des arêtes de* F est le graphe $G \setminus F = (X, E - F)/\approx$ où $\approx$ est la relation d'équivalence sur X définie par $x \approx y$ si et seulement si $x = y$ ou il existe un chemin de x à y dont toutes les arêtes sont dans F. Cette opération peut créer des boucles et des arcs doubles. Contracter un arc de C_3 donne C_2 formé d'une paire d'arêtes "doubles", et contracter une arête de C_2 fournit une boucle.

On dit que H est un *mineur de* G si et seulement si H est isomorphe à $G' \setminus F$ où G' est un sous-graphe (sous-graphe partiel? cf Terminologie chapitre D) de G et F un ensemble d'arêtes de G'. On note cela $H \trianglelefteq G$.

Proposition 2.3.1 : La relation de mineur est un préordre. La relation d'équivalence associée est l'isomorphisme des graphes.

Preuve : La relation $\trianglelefteq$ est transitive : ce n'est pas immédiat d'après la définition. Voir l'exercice 4.

Prenons comme *taille d'un graphe* la somme du nombre de ses arêtes et du nombre de ses sommets. Si $H \trianglelefteq G$, la taille de H est inférieure ou égale à celle de G, et d'après la définition 2.3.1, on ne peut avoir l'égalité que si F est vide

et $G' = G$. Mais alors H est isomorphe à G. Donc $H \trianglelefteq G \trianglelefteq H$, implique que G et H sont isomorphes. (Ceci est faux pour les graphes infinis). □

Proposition 2.3.2 : (1) Si H est un mineur de G alors $twd(H) \leq twd(G)$ et $pwd(H) \leq pwd(G)$.

(2) Si H est obtenu en contractant une arête de G alors $twd(H) \leq twd(G) \leq twd(H) + 1$ et $pwd(H) \leq pwd(G) \leq pwd(H) + 1$.

Preuve: (1) Avec la Proposition 2.2.2 et la transitivité, il suffit d'examiner le cas où H est obtenu par contractions d'une arête e. On construit à partir d'une décomposition (T, f) de G une décomposition (T, g) de H en remplaçant chaque sommet de $f(n)$ par son image dans H (sa classe d'équivalence) et en supprimant les arcs contractés.

Vérifions la condition (4) pour le sommet x issu de la contraction de e. L'ensemble des noeuds n de T tels que $x \in g(n)$ est la réunion de deux sous-arbres de T qui ont en commun au moins le noeud m tel que $f(m)$ contient e ; il est donc connexe. Les autres conditions sont évidentes. La largeur de la décomposition ne peut que diminuer.

(2) Inversement, soit à construire une décomposition (T, f) de G à partir d'une décomposition (T, g) de $H = G \setminus e$ où $e : x - y$. On considère que la contraction de e supprime y et redirige vers x les arêtes incidentes à y. On obtient g en ajoutant y à chaque boîte de (T, f). On place les arêtes de G incidentes à x et/ou à y dans les boîtes appropriées, de manière à satisfaire les conditions (2) et (3) de la Définition 2.1.1. □

On dit qu'une famille de graphes $\mathcal{F}$ ou une propriété de graphes $\mathcal{P}$ est *préservée par minoration* si, pour tous graphes G et H, si $H \trianglelefteq G$ et si G est dans $\mathcal{F}$ ou si $\mathcal{P}$ est vraie pour G, alors il en est de même pour H. Ainsi, nous venons de montrer que les familles $TWD(\leq k)$ et $PWD(\leq k)$ sont préservées par minoration. Il en est de même de la planarité, car, à partir d'un dessin planaire de G, on peut construire par suppressions et par "contractions progressives" d'arêtes (on laissera le lecteur formaliser cette dernière notion) un dessin planaire d'un mineur donné de G. Comme les graphes K_5 et $K_{3,3}$ ne sont pas planaires (ce que l'on montre sans peine à partir de la relation d'Euler : nombre de sommets + nombre de faces - nombre d'arêtes = 2, cf Chapitre E, un graphe planaire ne peut contenir aucun d'entre eux comme mineur. Le *Théorème de Kuratowski* (dans une formulation voisine due à Wagner) montre la réciproque : un graphe est planaire si il ne contient comme mineur ni K_5 ni $K_{3,3}$.

Ce théorème se généralise aux graphes représentables sur des surfaces telles que le tore. Il existe un nombre fini de "mineurs interdits" qui jouent le rôle des graphes K_5 et $K_{3,3}$ pour le plan, mais ces graphes ne sont connus exactement que dans le cas du plan projectif (il y en a 35). Pour le tore, on en connait au moins 2200 et la liste n'est pas complète. Le lecteur consultera le livre de Mohar and Thomassen [MT].

La famille $TWD(\leq k)$ est préservée par minoration. Est-elle caractérisée par des mineurs exclus en nombre fini ? La réponse est oui. Mais les mineurs

exclus ne sont connus que dans peu de cas : il s'agit de K_3 pour $k = 1$, c'est à dire pour les arbres (parmi les graphes simples), de K_4 pour $k = 2$, famille qui contient les *graphes séries-parallèles* dont nous parlerons plus loin, de K_5 et de 3 graphes à 6,8 et 10 sommets pour $k = 3$ (Voir [Bod1]).

On peut en dire un peu plus sur les mineurs exclus caractérisant $TWD(\leq k)$. On a mentionné que $twd(G_{n\times n}) = n$. Il en résulte qu'un graphe H dans $TWD(\leq k)$ n'a pas $G_{(k+1)\times(k+1)}$ pour mineur. Donc $G_{(k+1)\times(k+1)}$ ou l'un de ses mineurs, nécessairement planaire, fait partie des mineurs exclus pour $TWD(\leq k)$. Donc les mineurs exclus minimaux qui caractérisent $TWD(\leq k)$ comportent au moins un graphe planaire.

Inversement, est-ce que $twd(H) \geq n$ implique $G_{n\times n} \trianglelefteq H$? La réponse est non. Par contre $twd(H) \geq 2^{9n^5}$ implique $G_{n\times n} \trianglelefteq H$. Il s'agit d'un résultat difficile dont plusieurs preuves ont été publiées. Du fait que tout graphe planaire est mineur d'une grille $G_{n\times n}$ (exercice 4), il résulte que si une famille de graphes est telle qu'un graphe planaire P n'est un mineur d'aucun de ses graphes, alors elle est de largeur arborescente bornée. Plus précisément, si P est un graphe planaire simple et sans boucle, si H est un graphe qui ne contient pas P comme mineur, alors $twd(H) \leq 20^{2(2s(P)+4e(P))^5}$ ([RST], où $s(P)$ est le nombre de sommets de P, et $e(P)$ son nombre d'arêtes).

Le résultat fondamental de cette théorie est le *Théorème des Mineurs de Graphes* de Roberston et Seymour (*Graph Minor Theorem*) qui dit que toute famille de graphes qui préserve la minoration est caractérisée par un nombre fini de mineurs exclus ([RS-GM20]). Ce résultat généralise donc les cas particuliers cités plus haut concernant les graphes plongeables dans des surfaces et les familles $TWD(\leq k)$.

La problématique est la suivante : Soit une $\mathcal{F}$ une famille de graphes *préservée par isomorphisme et par minoration.* Définissons l'ensemble $Obst(\mathcal{F})$ des *obstructions* de $\mathcal{F}$:

$$Obst(\mathcal{F}) = \{G \mid G \notin \mathcal{F} \text{ et pour tout graphe } H,\ H \lhd G \Longrightarrow H \in \mathcal{F}\}.$$

On note $H \lhd G$ si $H \trianglelefteq G$ et $G \not\trianglelefteq H$, donc H n'est pas isomorphe à G. Alors la famille $\mathcal{F}$ est charactérisée en fonction de ses obstructions par :

$$\mathcal{F} = \{G \mid \text{pour tout graphe } H \in Obst(\mathcal{F}),\ H \not\trianglelefteq G\}.$$

Le Théorème des Mineurs de Graphes établit que l'ensemble $Obst(\mathcal{F})$ est toujours fini à isomorphisme près. C'est un résultat important sur la structure des graphes, qui a des conséquences également importantes en termes de complexité. Pour tout graphe simple G d'ordre n, on peut vérifier en temps $O(n^3)$ si un graphe fixé H est mineur de ce graphe ([RS-GM13]). Il en résulte que toute famille $\mathcal{F}$ préservée par isomorphisme et minoration possède un algorithme d'appartenance en $O(n^3)$. Si de plus $\mathcal{F}$ est de largeur arborescente bornée, donc si et seulement si l'une de ses obstructions est planaire, l'appartenance peut être testée en temps linéaire. Cela résulte de [RST] et du Corollaire 4.3.3 ci-dessous.

Des résultats similaires s'appliquent aux familles $PWD(\leq k)$. Elles sont fermées par minoration. Les arbres sont de largeur arborescente linéaire non

bornée (voir Section 2.1). Réciproquement, si un graphe H ne contient pas comme mineur une forêt F (union disjointe d'arbres) alors, $pwd(H) \leq s(F) = k-2$ et cette borne est optimale ([DF]).

L'utilisation de ces résultats pour la construction d'algorithmes suppose que l'on connaisse les ensembles $Obst(\mathcal{F})$ pour les classes $\mathcal{F}$ considérées. Or la preuve du Théorème des Mineurs de Graphes ne fournit aucune méthode permettant cette construction.

Pour les classes $PWD(\leq k)$ et $TWD(\leq k)$ cette construction est théoriquement possible. En effet les graphes de $Obst(PWD(\leq k))$ et de $Obst(TWD(\leq k))$ ont respectivement, au plus 2^{ak^4} arcs et au plus $exp(2, bk^5)$ arcs avec $exp(1,n) = 2^n$, et $exp(i+1,n) = 2^{exp(i,n)}$ ([Lag]). Il existe donc des algorithmes qui à tout entier k associent les ensembles finis de graphes $Obst(PWD(\leq k))$ et $Obst(TWD(\leq k))$. Ils consistent à tester tous les graphes sans sommets isolés dont les nombres d'arêtes sont au plus les valeurs ci-dessus. Mais ces algorithmes ne sont pas implémentables. Notons que pour les obstructions des familles de graphes dessinables sans croisement d'arcs sur les surfaces orientables de genre $k, k \geq 1$, on ne dispose même pas d'un résultat semblable de calculabilité théorique. D'autre part, les ensembles d'obstructions sont en général très grands. Ainsi l'ensemble $Obst(PWD(\leq k))$ contient plus de $(k!)^2$ arbres et ces arbres ont tous $(5.3^k - 1)/2$ sommets ([TUK]).

2.4 Decompositions particulières et algorithmes

Un *graphe triangulé* (*chordal graph*) est un graphe simple, connexe qui admet une décomposition arborescente dont chaque boîte induit une clique. Ces graphes ont plusieurs autres caractérisations : ce sont les graphes sans cycle induit C_n pour $n > 3$ ou bien, les graphes des intersections des sous-graphes connexe d'un arbre. Pour un tel graphe G on a $twd(G) = \kappa(G) - 1$ où $\kappa(G)$ est l'ordre (nombre de sommets) maximum d'une clique. Ils ne sont donc pas de largeur arborescente bornée.

Un *k-arbre* (*k-tree*) est un graphe triangulé dont toutes les cliques maximum sont d'ordre k. Ces graphes peuvent être construits récursivement de la façon suivante : le k-arbre minimal est K_k, il a k sommets. On définit un k-arbre à n sommets en ajoutant à un k-arbre G à $n-1$ sommets un sommet supplémentaire et k arêtes reliant ce sommet aux k sommets d'une clique de G. Voir [Bod1]. Ils possède un ordre d'élimination de type k qui est *parfait*, c'est à dire tel que tout nouveau somment est adjacent aux sommets d'une clique.

Proposition 2.4.1 : (1) Pour un graphe simple G, $twd(G) = k-1$, où k est la valeur minimum de $\kappa(H)$ pour H triangulé et G sous-graphe de H.ayant les mêmes sommets.

(2) Pour un graphe simple G, $twd(G) = k-1$, où k est la valeur minimum telle qu'il existe un k-arbre H avec G sous-graphe de H.ayant les mêmes sommets.

En particulier, un graphe simple a une largeur arborescente au plus k si et seulement si c'est un *k-arbre partiel* (*partial k-tree*) c'est à dire un sous-graphe d'un k-arbre. La terminologie en terme de largeur arborescente s'est imposée. Elle a l'avantage de ne pas concerner que les graphes simples.

Preuve : (1) Conséquence immédiate des définitions.
(2) En ajoutant si besoin des boîtes intermédiares et des sommets dans les boîtes, on peut toujours transformer une décomposition arborescente de largeur k en une décomposition arborescente du même graphe dont toute boîte a $k+1$ sommets et telle que deux boîtes voisines ont en commun exactement k sommets. On ajoute alors des arêtes pour que chaque boîte induise une clique et l'on obtient ainsi un k-arbre. La preuve est facile à compléter. □

Décompositions arborescentes soumises à des conditions supplémentaires

Il peut être utile d'imposer des conditions supplémentaires aux décompositions arborescentes pour certaines applications algorithmiques, quitte à perdre l'optimalité. On peut considérer que les décompositions arborescentes linéaires entent dans ce cadre. La proposition suivante regroupe quelques résultats à ce sujet.

Proposition 2.4.2 : 1) Tout arbre à n noeuds possède une décomposition arborescente de largeur 2 dont l'arbre est de diamètre au plus $3.log(n)$; tout graphe à n sommets possède une décomposition arborescente de largeur au plus $3k+2$ dont l'arbre est de diamètre au plus $O(log(n))$.
2) Tout graphe G a une décomposition arborescente connexe de largeur au plus $twd(G)$.
3) Tout graphe G a une décomposition arborescente mengerienne de largeur au plus $twd(G)$.
4) Il existe une fonction f telle que tout graphe G a une décomposition arborescente de type "domino" de largeur au plus $f(twd(G), Deg(G))$ où $Deg(G)$ désigne le degré maximum de G.

Références : [BodHag] et [CouVan] pour 1) ; Fraignaud [Fra] pour 2) une décomposition (T, f) est *connexe* si pour tout arc $e : u - v$ de T, le sous-graphe de G induit par les sommets des boîtes de T accessible à partir de u par un chemin qui évite v est connexe ; 3) Voir [DF] pour ce résultat dû à R. Thomas qui fournit une voisine du Théorème de Menger ; 4) Voir [Bod1] pour ce résultat dû à Bodlaender et Engelfriet ; une décomposition est de type "domino" si aucun sommet n'appartient à plus de 2 boîtes. Il n'est pas possible de remplacer f par une fonction qui ne dépend que de $twd(G$.

Les applications algorithmiques que nous exposerons plus loin nécessitent que les graphes considérés soient munis d'un décomposition arborescente de

largeur la plus petite possible. La construction de telles décompositions est donc un élément important.

Proposition 2.4.3 : 1) Les problèmes consistant à décider si un couple donné (G, k) vérifie $twd(G) \leq k$ ou $pwd(G) \leq k$ sont NP-complets.

2) Le problème consistant à décider si un couple donné (G, k) vérifie $twd(G) \leq k$ est polynomial pour G dans des classes de graphes particulières telles que celle des graphes triangulés (chordal graphs) ou leurs compléments, des cographes, des *graphes d'intervalles* (*interval graphs*), des *graphes d'intersection des cordes d'un cercle* (*circle graphs*).

3) Pour tout k, les problèmes consistant à décider si un graphe simple donné G vérifie $twd(G) \leq k$ ou $pwd(G) \leq k$ sont linéaires en le nombre de sommets de G.

4) Pour tout k il existe un algorithme en $O(nlog(n))$ qui, pour un graphe simple donné G ou bien répond que $twd(G) > k$, ou bien produit une décomposition arborescente de largeur au plus $3k + 2$.

Preuve : Pour les références précises on consultera [Bod2]. Pour les graphes triangulés et les graphes d'intersection des cordes d'un cercle, le problème $pwd(G) \leq k$ est NP-complet pour k fixé. Le résultat 3) pour la largeur arborescente, dû à Bodlaender, est exposé dans [DF]. La complexité est en $2^{32k^3}.s(G)$, et cet algorithme n'est pas concrétement implémentable. Un graphe est de largeur arborescente linéaire au plus k ssi il est de largeur arborescente au plus k et ne contient comme mineur aucun graphe d'un ensemble fini. Le Corollaire 4.3.3 ci-dessous fournit un algorithme linéaire pour le test. □

3 Expressions algébriques et grammaires

Cette section introduit des *expressions algébriques* (ou *termes* au sens de la logique et de la théorie de la réécriture) qui désignent les graphes de largeur arborescente au plus k. Cette démarche offre plusieurs avantages : on peut ainsi représenter linéairement et sans ambiguïté les graphes autrement qu'en listant les sommets et les arêtes ; cette représentation est structurée en ce sens que la syntaxe reflète la structuration des graphes en termes sous-graphes recollés entre eux ; elle permet des preuves et des calculs par induction sur la structure des termes ; elle permet de définir des grammaires de graphes en évitant complètement les complications techniques liées aux réécritures de graphes.

3.1 Ecriture algébrique des décompositions arborescentes

Définition 3.1.1 : Graphes avec sources

On considère des graphes qui peuvent avoir des boucles et des arêtes multiples. Pour simplifier la présentation, on ne considèrera que des graphes non-orientés. La généralisation aux graphes orientés sera immédiate.

Afin de distinguer certains sommets, on définit un ensemble dénombrable $\mathcal{C}$ d'étiquettes. Un *graphe avec sources* est un couple G constitué d'un graphe (X, A) et d'une bijection $\mathbf{src}_G : C \longrightarrow S$ où C est une partie finie de $\mathcal{C}$ et S une partie de X. Les sommets de S sont les *sources* de G. Le *nom d'une source* s est l'étiquette c telle que $\mathbf{src}_G(c) = s$. On dira encore que s est une *c-source*. L'ensemble C noté $\tau(G)$ est le *type* de G. Un graphe est un graphe avec source dont le type est vide.

Un *isomorphisme de graphes avec sources* doit préserver les noms des sources de manière évidente. Un sous-graphe H d'un graphe avec sources G a les sources qui sont des sommets de G, et leurs noms sont les mêmes dans H et dans G. Ainsi $\tau(H) \subseteq \tau(G)$.

On appellera *graphe abstrait avec sources* une classe d'isomorphisme d'un graphe avec sources. Cette distinction qui alourdit certains énoncés est nécessaire car les expressions algébriques que nous allons définir vont définir des graphes abstraits.

On désigne par $\mathcal{G}$ l'ensemble des graphes abstraits avec sources, et par $\mathcal{G}_C$ le sous-ensemble de ceux de type C.

Définition 3.1.2 : Décomposition et composition parallèles des graphes avec sources.

Soit G un graphe avec sources. On écrit $G = H//K$ si et seulement si H et K sont des sous-graphes de G tels que : H et K n'ont pas d'arête en commun, G est l'union de H et de K et les sommets communs à H et à K sont des sources de G (et donc aussi de H et K). Il en résulte que :

$$\tau(G) = \tau(H) \cup \tau(K) \qquad (1)$$

et que les sommets communs à H et à K sont les c-sources pour c dans $\tau(H) \cap \tau(K)$. On dit que G est la *composition parallèle* de H et de K. Nous avons défini en fait un *opérateur de décomposition* d'un graphe en deux sous-graphes sans arêtes communes. On peut généraliser cette définition et faire de $//$ un *opérateur de composition* de graphes de la manière suivante :

Si H et K sont des graphes avec sources non nécessairement disjoints, on construit G en définissant H' et K', copies disjointes de H et K respectivement et en "recollant" H' et K' en identifiant leurs sources de mêmes noms. Formellement, on définit G comme le quotient du graphe $H' \cup K'$ par la relation d'équivalence $x \approx y \ :\Longleftrightarrow x = y$ ou $\{x, y\} = \{\mathbf{src}_{H'}(c), \mathbf{src}_{K'}(c)\}$ pour quelque c. Dans ce cas, G n'est défini qu'à isomorphisme près, car les copies disjointes H' et K' sont quelconques. Si l'on veut être très précis, on peut prendre $H' = (X \times \{1\}, A \times \{1\})$ avec les incidences évidentes si $H = (X, A)$, et de même $K' = (Y \times \{2\}, B \times \{2\})$ si $K = (Y, B)$.

Une écriture de la forme $G = H//K$ peut donc être lue de deux manières possibles. Plutôt que d'alourdir les notations en distinguant formellement un

graphe "concret" et le graphe abstrait correspondant, on utilisera une seule notation et le contexte lèvera les ambiguïtés éventuelles. Noter que l'écriture $G = H//H$ n'a qu'une lecture possible, celle qui considère // comme un opérateur de composition. Sauf si H ne comporte que des sources et aucune arête, $G = H//H$ n'est pas isomorphe à H.

On peut voir l'opération // comme une généralisation aux graphes de la concaténation des mots. Cette opération est commutative, ce qui n'est pas le cas de la concaténation.

Définition 3.1.3 : Opérations de manipulation des sources

On écrit $G = \mathbf{fg}_A(H)$ si et seulement si G est le même graphe que H avec une fonction $\mathbf{src}_G$ qui est la restriction de $\mathbf{src}_H$ à l'ensemble $\tau(H) - A$, où A est une partie finie de $\mathcal{C}$. On a donc :

$$\tau(\mathbf{fg}_A(H)) = \tau(H) - A. \qquad (2)$$

Cette opération est nommée *oubli de sources* car les a-sources (pour a dans A) sont "oubliées" en tant que sources mais les sommets correspondants existent toujours comme sommets "ordinaires". La notation **fg** fait référence aux *foncteurs d'oubli* (*forgetful functors*) en théorie des catégories. On notera $\mathbf{fg}_a$ lorsque A est réduit à a. Si $\tau(H) \cap A = \varnothing$ alors $\mathbf{fg}_A(H) = H$.

On écrit $G = \mathbf{ren}_h(H)$ si et seulement si G est le même graphe que H avec $\mathbf{src}_G = \mathbf{src}_H \circ h^{-1}$ où h est une bijection : $A \longrightarrow A$, A est une partie finie de $\mathcal{C}$, et h est étendue en l'identité en dehors de A. On a donc :

$$\tau(\mathbf{ren}_h(H)) = h(\tau(\mathbf{H})). \qquad (3)$$

Cette opération est un *renommage de sources* car la a-source de H devient la $h(a)$-source de G. Pour faciliter l'écriture, on notera $\mathbf{ren}_{a \longleftrightarrow b}$ l'opération $\mathbf{ren}_h$ lorsque $h(a) = b, h(b) = a$ et $h(c) = c$ pour $c \neq a, b$.

Ces opérations forment un ensemble dénombrable car elles sont spécifiées par des sous-ensembles finis A de $\mathcal{C}$ qui est dénombrable, ou par des fonctions sur de tels ensembles finis. D'autre part, elles s'appliquent à tous les graphes avec sources, quelques soient leurs types. Les opérations d'oubli et de renommage de sources s'appliquent aussi bien aux graphes concrets qu'aux graphes abstraits.

Pour construire des graphes au moyen de ces opérations, on utilisera également les constantes suivantes : $\mathbf{a}$, $\mathbf{a}^{bcl}$, $\mathbf{ab}$ qui désignent respectivement un sommet isolé qui est une a-source, un sommet avec une boucle qui est une a-source, et une arête dont les extrémités sont une a-source et une b-source, ceci pour tous a, b avec $b \neq a$ dans C. Ces constantes désignent en fait les graphes abstraits correspondants, de types respectifs $\{a\}, \{a\}$ et $\{a, b\}$.

Les définitions présentées ici simplifient la présentation de [Cou97] en ce qu'elles évitent d'utiliser des *F-algèbres à plusieurs sortes d'objets*. Cette simplification est due au fait que $G//H$, $\mathbf{fg}_A(H)$ et $\mathbf{ren}_h(H)$ sont bien définis quelques soient les types de G et de H.

L'opération // est associative et commutative. On utilisera donc la notation infixée sans parenthèses. Autrement dit, on considérera que $G//(H//K)$, $(G//H)//K$, et $(G//K)//H$ sont le même terme que l'on notera aussi bien $G//H//K$ que $G//K//H$. Noter que toute opération $\mathbf{ren}_h$ peut s'écrire comme une composition finie d'opérations du type $\mathbf{ren}_{a\longleftrightarrow b}$.

On notera $\mathcal{F}$ l'ensemble de ces constantes et de ces opérations et, pour tout C fini, $C \subset \mathcal{C}$, on notera $\mathcal{F}_C$ son sous-ensemble fini, :

$$\{//, \mathbf{fg}_A, \mathbf{ren}_h, \mathbf{a}, \mathbf{a}^{bcl}, \mathbf{ab} \mid A \subseteq C, a, b \in C, \ \ h \text{ est l'identité en dehors de } A\}$$

On définit ainsi sur $\mathcal{G}$ une structure d'algèbre avec un ensemble infini dénombrable $\mathcal{F}$ d'opérations et de constantes, ce qui n'est pas usuel. On notera toutefois que si l'on se restreint à des graphes avec des sources étiquettées dans un ensemble fini C, on n'utilise alors qu'un nombre fini d'opérations. Mais cet ensemble fini ne peut pas engendrer tous les graphes finis ainsi qu'il résulte du Théorème 3.1.1 ci-dessous.

On notera $T(\mathcal{F})$ l'ensemble des termes (finis), écrits avec les symboles de $\mathcal{F}$, et bien formés, c'est à dire qui respectent les arités. Tout terme de $T(\mathcal{F})$ appartient de fait à un ensemble $T(\mathcal{F}_C)$. Les opérations et constantes de $\mathcal{F}_C$ ne peuvent définir que des graphes de types contenu dans C. Un terme t dans $T(\mathcal{F})$ a pour valeur un graphe abstrait, ainsi qu'il résulte des définitions, que l'on peut définir comme la classe d'équivalence d'un graphe concret construit sur les occurrences dans t des symboles de constante. Cette construction fait l'objet de l'exercice 5. Le type de ce graphe peut être déterminé à partir du terme en utilisant les règles de typage (1), (2) et (3) indiquées plus haut.

Au moyen de ces opérations, on peut en définir d'autres appellées *opérations dérivées* (*derived operations*), analogues à des "macros", qui permettront d'écrire des expressions plus concises et plus lisibles. Par exemple, la *composition en série* des graphes à deux sources (désignées par les étiquettes 1 et 2) peut se définir ainsi :

$$G \bullet H = \mathbf{fg}_3(\mathbf{ren}_{2\longleftrightarrow 3}(G)//\mathbf{ren}_{1\longleftrightarrow 3}(H))$$

pour G et H de type {1,2}. Une opération dérivée est définie par un terme où aucune variable n'a plus d'une occurrence. La composition en série est définie par le terme $\mathbf{fg}_3(\mathbf{ren}_{2\longleftrightarrow 3}(x_1)//\mathbf{ren}_{1\longleftrightarrow 3}(x_2))$. L'opération qui à G associe $G \bullet G = \mathbf{fg}_3(\mathbf{ren}_{2\longleftrightarrow 3}(G)//\mathbf{ren}_{1\longleftrightarrow 3}(G))$, définie par le terme $\mathbf{fg}_3(\mathbf{ren}_{2\longleftrightarrow 3}(x_1)//\mathbf{ren}_{1\longleftrightarrow 3}(x_1))$ n'est pas une opération dérivée, car x_1 a deux occurrences dans ce terme.

On utilisera $[n] = \{1, ..., n\}$ comme ensemble d'étiquettes de sources.

Lemme 3.1 : Tout graphe à n sommets est défini par un terme de $T(\mathcal{F}_{[n]})$.

Preuve : On identifie les sommets du graphe donné G supposé sans sources aux entiers $1, ..., n$. On utilise une étiquette x pour chaque sommet x de G, et on définit G par le terme

$$t = \mathbf{fg}_{[n]}(\mathbf{x}_1\mathbf{y}_1//\mathbf{x}_2\mathbf{y}_2//...//\mathbf{x}_n\mathbf{y}_n//\mathbf{z}_1//\mathbf{z}_2//... //\mathbf{z}_n//\mathbf{w}_1^{bcl}//\mathbf{w}_2^{bcl}//...)$$

où les arêtes sont les $x_i - y_i$, les sommets isolés sont les z_i, et les sommets avec des boucles sont les w_i. Si G est un graphe avec sources, la construction est semblable avec un choix approprié des étiquettes de sources et une modification adéquate de $\mathbf{fg}_{[n]}$. □

Cette construction n'est pas très intéressante. Elle est équivaut à définir le graphe par une liste d'arêtes et de sommets, et ne comporte aucune structuration. Elle représenterait le graphe de la figure 1 au moyen de 16 étiquettes, alors que la décomposition de ce graphe définie par les figures 2 et 3 permet, comme on va le démontrer de définir une expression avec 4 étiquettes.

Une *décomposition arborescente d'un graphe avec sources*, est une décomposition arborescente de ce graphe dont une boîte contient toutes les sources. La notion de largeur arborescente (encore notée *twd*) en résulte. Dans le cas d'une décomposition arborescente *linéaire*, on demande en plus que les sources soient toutes dans l'une des boîtes situées aux extrémités du chemin. Ces définitions s'appliquent de manière évidente à des graphes abstraits.

Théorème 3.1.1 : Un graphe avec sources est de largeur arborescente au plus $k-1$ si et seulement si il est la valeur d'un terme $t \in T(\mathcal{F}_C)$ pour C de cardinalité k.

Preuve : Soit G un graphe défini par un terme $t \in T(\mathcal{F}_C)$ pour C de cardinalité k. On va définir par induction sur la structure de t une décomposition arborescente orientée (T, f) de G de largeur $k-1$, telle la boîte associée à sa racine est l'ensemble des sources (éventuellement l'ensemble vide). Les boîtes vides seront supprimées si nécessaire dans une étape finale de la construction (cf. Définition 2.1.1).

Si $t = \mathbf{a}, \mathbf{a}^{bcl}$ ou $\mathbf{ab}$, on prend T réduit à un unique sommet qui est donc la racine, et la boîte correspondante contient le ou les sommets et, dans les deux derniers cas, l'arête de G.

Si $t = \mathbf{ren}_h(t')$ et (T', f') a été construite pour le graphe G' défini par t' conformément à l'hypothèse d'induction, on prend $(T, f) = (T', f')$. Noter que G et G' ont les mêmes sources, mais étiquettées différemment.

Si $t = \mathbf{fg}_A(t')$ et (T', f') a été construite pour le graphe G' défini par t', on construit (T, f) en ajoutant un nouveau sommet r qui sera la racine de T, un arc de r vers la racine r' de T', $f(n) = f'(n)$ si n est un noeud de T et $f(r)$ est l'ensemble des sources de G. Noter que : $f(r) \subseteq f(r')$.

Si $t = t_1//t_2$ définit G, on note G_1 et G_2 les sous-graphes de G définis par t_1 et t_2 et d'après l'hypothèse d'induction on peut supposées construites des décompositions (T_1, f_1) et (T_2, f_2) de G_1 et G_2 respectivement. On supposera T_1 et T_2 disjoints : si ils ne le sont pas, on remplace l'un d'eux par une copie isomorphe. On construit T en ajoutant à l'union de T_1 et T_2 un nouveau sommet r qui sera la racine de T et des arcs de r vers les racines r_1 et r_2 de T_1 et T_2. On définit $f(r)$ comme la réunion de $f_1(r_1)$ et de $f_2(r_2)$. C'est bien l'ensemble des sources de G.

On vérifie par induction sur la structure de t que (T, f) est une décomposition arborescente de G, et que chacune de ses boîtes, qui est l'ensemble des sources d'un graphe défini par un sous-terme de t, comporte au plus k sommets, puisqu'il n'y a que k étiquettes de sources.

Réciproquement, soit $G = (X, E)$ un graphe avec sources est de largeur arborescente au plus $k-1$, dont on connait une décomposition orientée (T, f). Les sources de G sont toutes dans la boîte associée à la racine. D'après le Corollaire 2.1, il existe un coloriage $c : X \longrightarrow [k]$ tel que deux sommets d'une même boîte ont des couleurs différentes. On va utiliser $[k]$ comme ensemble d'étiquettes de sources. La preuve se fait par récurrence sur le nombre de noeuds de T.

Si T est réduit à un seul noeud, le graphe G a au plus k sommets, et on construit un terme de $T(\mathcal{F}_{[k]})$ qui le définit en utilisant le Lemme 3.1.

Sinon, soit r la racine et $T_1, ..., T_p$ les sous-arbres de T issus de ses fils $n_1, ..., n_p$. Soient $G_1, ..., G_p$ les graphes définis par les décompositions associés aux sous-arbres $T_1, ..., T_p$. (G_i est l'union des graphes associées aux boîtes de T_i). Les sources de ces graphes sont les sommets du graphe (défini par) $f(n_i)$. Plus précisément, x est la j-source de $f(n_i)$ si et seulement si $c(x) = j$. Par hypothèse de récurrence, on peut supposer construits des termes $t_1, ..., t_p$ pour ces graphes. Pour chaque i on note A_i l'ensemble des étiquettes des sources de G_i qui ne sont pas dans la boîte racine de T. On définit G'_i comme le graphe $\mathbf{fg}_{A_i}(G_i)$.

Le graphe G est la composition parallèle des graphes $G'_1, ..., G'_p$ augmentée des sommets et des arêtes *propres à* $f(r)$ c'est à dire qui sont dans $f(r)$ mais non dans $G'_1//...//G'_p$. Il en résulte que G est défini par le terme :

$$\mathbf{fg}_{A_1}(t_1)//...//\mathbf{fg}_{A_p}(t_p)//\mathbf{x}_1\mathbf{y}_1//...//\mathbf{x}_n\mathbf{y}_n//\mathbf{z}_1//... \ //\mathbf{z}_m//\mathbf{w}_1^{bcl}//\mathbf{w}_2^{bcl}//...$$

où $z_1, z_2, ... , z_n$ sont les sommets isolés propres à $f(r)$, où $\mathbf{w}_1^{bcl}, \mathbf{w}_2^{bcl}, ...$ définissent les boucles propres à $f(r)$, et $x_1 - y_1, x_2 - y_2, ..., x_n - y_n$ sont les arêtes propres à $f(r)$.□

Remarque : Il résulte de cette preuve que l'on peut construire un graphe de largeur arborescente au plus $k-1$ avec k étiquettes, sans utiliser l'opération de renommage de sources. Cette opération est néanmoins très utile. Sans elle, on ne peut pas définir par un terme la composition en série des graphes de type $\{1, 2\}$ comme le montre l'exemple suivant.

Soit G le graphe défini par le terme $t = \mathbf{12}//\mathbf{fg}_3(\mathbf{13}//\mathbf{32})$. Soit H le graphe $G \bullet G$, défini par le terme $\mathbf{fg}_3(\mathbf{ren}_{2\longleftrightarrow 3}(t)//\mathbf{ren}_{1\longleftrightarrow 3}(t))$. Si l'on s'interdit l'opération de renommage, on peut définir H par le terme $\mathbf{fg}_3(t_1//t_2)$ où t_1 définit $\mathbf{ren}_{2\longleftrightarrow 3}(G)$ et t_2 définit $\mathbf{ren}_{1\longleftrightarrow 3}(G)$, en prenant $t_1 = \mathbf{13}//\mathbf{fg}_2(\mathbf{12}//\mathbf{23})$ et $t_2 = \mathbf{32}//\mathbf{fg}_1(\mathbf{31}//\mathbf{12})$.

L'inconvénient est que l'on ne peut pas écrire un terme définissant $G \bullet G'$ en utilisant "tels quels" des termes t et t' qui définissent respectivement G et G'. On est obligé de modifier ces termes. D'autre part, dans l'exemple de $H = G \bullet G$,

la syntaxe du terme $\mathbf{fg}_3(t_1//t_2)$ masque le fait intéressant que le même graphe G figure deux fois comme "facteur" de H.

Corollaire 3.1.1 : Un graphe avec sources est de largeur arborescente linéaire au plus $k-1$ si et seulement si il est la valeur d'un terme $t \in T(\mathcal{F}_C)$ pour C de cardinalité k qui est tel que dans tout sous-terme de t de la forme $t_1//.../t_q$ au plus un des termes $t_1, ..., t_q$ n'est pas une constante.

Preuve : Soit G défini par un terme t qui satisfaisait la condition de l'énoncé. Comme pour la preuve du Théorème 3.1.1, on utilise une induction sur la structure de t. On construit ainsi une décomposition arborescente linéaire et orientée (T, f) du graphe G dont la racine est de degré sortant 1, c'est une extrémité du chemin T, et les sommets de la boîte correspondante sont les sources de G. On a les cas suivants.

Si t est une constante, $\mathbf{ren}_h(t')$ ou $\mathbf{fg}_A(t')$, la construction est la même que pour la preuve du Théorème 3.1.1.

Autrement $t = t_1//.../t_q$ et $t_2, ..., t_q$ sont des constantes. Utilisant l'hypothèse d'induction, on suppose (T', f') construite pour le graphe G' défini par t_1. On construit (T, f) en ajoutant un nouveau sommet r qui sera la racine de T, un arc de r vers la racine r' de T', $f(n) = f'(n)$ si n est un noeud de T et $f(r)$ est l'ensemble des sources de G et des arêtes définies par les constantes $t_2, ..., t_q$. Les extrémités de ces arêtes sont des sources de G et $f(r) \supseteq f(r')$. On obtient une décomposition arborescente linéaire orientée de G dont la racine est de degré sortant 1 et les sommets de la boîte correspondante sont les sources. Donc $pwd(G) \leq k-1$, puisque dans chaque boîte, les sommets qui sont tous des sources sont en nombre au plus k.

Réciproquement, soit (T, f) une décomposition arborescente linéaire de G. On lui applique la construction de la preuve du Théorème 3.1.1 et l'on a $p = 0$ ou 1. On obtient donc un terme de la forme souhaitée.□

Pour engendrer des graphes orientés, on utilisera la constante $\overrightarrow{\mathbf{ab}}$ qui désigne un arc d'une a-source vers une b-source. Tous les résultats s'étendent de façon immédiate.

3.2 Grammaires

La grammaire d'une langue définit des types de mots et de groupes de mots en fonction desquels on peut décomposer une phrase et lui donner une structure arborescente qui est liée à son sens. Cette structure permet de formuler des règles dites « d'accord », et elle constitue une représentation « pivot » pour un processus de traduction automatique. Ainsi la traduction d'une langue dans une autre peut être vue comme composée de deux phases, une phase d'analyse syntaxique partant de la phrase donnée et aboutissant à la construction de son arbre syntaxique (avec les problèmes d'ambiguïté inhérents aux langues

naturelles), et d'une phase de transformation de cet arbre en une phrase de la langue visée. L'approche grammaticale formalisée des langues naturelles a été introduite par N. Chomsky dans les années 50.

Les langages de programmation ont des *grammaires*, conçues pour éviter toute ambiguïté, et le *compilateur* a pour tâche la traduction d'un programme écrit dans un language de programmation en un programme écrit en "langage-machine" directement exécutable par l'ordinateur, contrairement au premier.

Une *grammaire* peut être formalisée comme un ensemble de définitions simultanées d'un ensemble de termes. Prenons l'exemple très simple des termes écrits avec un unique symbole de fonction binaire f et deux constantes a et b. Quelques exemples de termes en commençant par les plus petits sont :

$$a, b, f(a,b), f(a,a), f(a,f(a,b)), f(f(b,b),a), \text{ etc...}$$

On peut les caractériser par des *règles de formation*. Dans des ouvrages de mathématiques qui ne s'attardent pas à des questions formelles on trouve des définitions de ce genre :

1) a et b sont des termes,

2) si s et t sont des termes alors $f(s,t)$ est un terme,

3) rien n'est un terme que par application des règles qui précèdent.

Pour être correct, il faut préciser qu'un terme est un *mot* c'est à dire une suite finie de symboles. Cette hypothèse de finitude est nécessaire pour exclure le terme infini $f(f(.,.),f(.,.))$ solution de l'équation $T = f(T,T)$ (sorte de série formelle généralisée). En utilisant le vocabulaire de la théorie des langages formels, on écrirait que l'ensemble des termes est le langage engendré par la grammaire composée des règles de réécriture :

$< terme > \longrightarrow a$,

$< terme > \longrightarrow b$,

$< terme > \longrightarrow f(< terme >, < terme >)$.

Une manière plus élégante de formuler cela est de dire que l'ensemble des termes est le plus petit ensemble de mots L qui contient a, b et inclut l'ensemble de mots $f(L,L)$ (où $f(L,L)$ est l'ensemble des mots de la forme $f(s,t)$ pour tous s et t dans L.)

L'analyse syntaxique des langages de programmation s'appuie sur la notion de grammaire où, au lieu d'une unique catégorie d'expressions, on en utilise plusieurs, par exemple <instruction>, <déclaration>, <expression booléenne>, <expression arithmétique>, <appel de procédure>.

La notion de grammaire peut également s'appliquer à la description d'autres objets tels que les graphes. On va prendre un l'exemple simple des *cographes* . Ce sont les graphes non orientés, simples, sans bouclesdéfinissables à partir de deux opérations : l'*union disjointe* et le *produit complet*, notées respectivement $\oplus$ et $\otimes$, à partir des graphes rédits à un seul sommet.

Le graphe $G = H \oplus K$ est défini à partir de deux copies disjointes des graphes H et K : on en prend l'union des sommets et l'union des ensembles

d'arcs. Le graphe $G = H \otimes K$ est défini comme $H \oplus K$ augmenté de tous les arcs possibles entre les sommets de H et ceux de K. Ces graphes ne sont définis qu'à isomorphisme près puisque, à chaque étape, on prend des copies disjointes. Notons $\mathbf{1}$ le graphe réduit à un sommet isolé. Les *cographes* sont les graphes définis par les termes finis écrits avec $\mathbf{1}$, $\oplus$ et $\otimes$.

On peut également écrire que l'ensemble des cographes L est l'unique solution, parmi les ensembles de graphes finis, de l'équation :

$$L = (L \oplus L) \cup (L \otimes L) \cup \{\mathbf{1}\}.$$

Chacun des graphes engendrés possède une description sous la forme d'un terme. Le graphe $K_{3,3}$ peut donc être défini par le terme $(\mathbf{1}\oplus\mathbf{1}\oplus\mathbf{1})\otimes(\mathbf{1}\oplus\mathbf{1}\oplus\mathbf{1})$, le graphe K_n par $(\mathbf{1}\otimes\mathbf{1}\otimes...\otimes\mathbf{1})$, avec n occurrences de $\mathbf{1}$. Le graphe P_4 ne peut pas être représenté par un terme sur ces opérations, la vérification est facile.

Les arbres peuvent être définis au moyen des deux équations qui suivent :
<arbre> = $\mathbf{fg}_r$(<arbre avec racine>)
<arbre avec racine> = (<arbre avec racine>//<arbre avec racine>) $\cup$
$\mathbf{fg}_n(\mathbf{nr}//r\mathbf{en}_{n\longleftarrow r}(<$ arbre avec racine $>)) \cup \{\mathbf{r}\}$.

On les construit comme arbres avec racine et puis on "oublie les racines". On utilise deux étiquettes de sources, r pour désigner la racine courante et n pour désigner une "nouvelle racine", lors d'une étape intermédiaire de construction.

3.3 Grammaires comme systèmes de définitions récursives.

Les opérations définies dans la Section 3.1 munissent l'ensemble des graphes abstraits avec sources d'une structure d'*algèbre universelle* dite encore de *magma* [Cou97]. (Ce terme a été introduit par Bourbaki [Bou] pour désigner une structure dotée d'une unique opération binaire sans aucun axiome (nommée aussi *groupoïde*), mais n'a pas été communément adopté. C'est dommage car le terme "algèbre" est trop utilisé. Il désigne les anneaux qui sont aussi des espaces vectoriels les familles d'ensembles fermées par des opérations booléennes infinitaires et d'autres structures comme les algèbres de Boole). Conformément à la pratique courante, nous utiliserons le terme de *F-algèbre*, où F désigne l'ensemble des opérations internes. Quelques préliminaires d'Algèbre Universelle peuvent sembler loin des graphes mais faciliteront considérablement les définitions.

Définition 3.3.1 : F-algèbres.

Une *signature fonctionnelle* est un couple formé d'un ensemble fini ou dénombrable F de *symboles de fonctions* et d'une fonction $\rho : F \longrightarrow \mathbf{N}$ qui associe à chacun d'eux une *arité*, c'est à dire un entier positif ou nul qui définira son nombre d'arguments. Un symbole d'arité 0 est une *constante*. Pour simplifier on dira simplement que F est une signature, en supposant fixée la fonction arité.

On notera $T(F)$ l'ensemble des termes finis écrits avec ces symboles et bien formés relativement à la fonction arité. La notation usuelle sera avec des parenthèses et des virgules, comme ci-dessus. Le terme $f(a, b, f(a, c))$ ne peut pas être bien formé, car f devrait avoir l'arité 2 et 3 en même temps ce qui est exclus. Si F ne contient aucune constante, $T(F)$ est vide.

Soit X un ensemble fini ou dénombrable. On notera $T(F, X)$ l'ensemble des termes finis bien formés écrits avec ces symboles et avec les variables de X. Un exemple de terme est $f(a, g(a, x, y))$ qui appartient à $T(\{f, g, a, b, c\}, \{x, y, z\})$.

Une *F-algèbre* (où F est une signature) est un objet $\mathbf{M} = (M, (f_{\mathbf{M}})_{f \in F})$, dont M est le domaine et dont, pour chaque $f \in F$, la composante f_M est une fonction totale $M^{\rho(f)} \longrightarrow M$. Chaque constante désigne un élément de M. Un monoïde, un groupe, un anneau, un corps sont des F-algèbres, pour des signatures bien connues. Tout terme $t \in T(F)$ a une valeur dans M, notée $t_{\mathbf{M}}$, obtenue en l'évaluant en utilisant les fonctions $f_{\mathbf{M}}$ comme interprétation des symboles f de fonctions et de constantes.

Définition 3.3.2 : Parties équationnelles d'une F-algèbre.

Afin de définir des fonctions sur $\mathcal{P}(M)$, on va introduire des polynômes. Un *polynôme* est un terme p de la forme $t_1 \cup t_2 \cup ... \cup t_k$ où $t_1, ..., t_k \in T(F, X)$, ou bien de la forme $\varnothing$ (où $\varnothing$ est une constante qui va désigner l'ensemble vide $\emptyset$). On note $Pol(F, X)$ l'ensemble des polynômes sur F et X, ensemble de variables.

A un terme t de $T(F, \{x_1, ..., x_n\})$ est associée une fonction $t_{\mathcal{P}(\mathbf{M})} : \mathcal{P}(M)^n \longrightarrow \mathcal{P}(M)$ ainsi définie, pour $D_1, ..., D_n \subseteq M$:
$t_{\mathcal{P}(\mathbf{M})}(D_1, ..., D_n) = D_i$ si $t = x_i$,
$t_{\mathcal{P}(\mathbf{M})}(D_1, ..., D_n) = \{c_{\mathbf{M}}\}$ si t est la constante c,
$t_{\mathcal{P}(\mathbf{M})}(D_1, ..., D_n) = \{f_{\mathbf{M}}(a_1, ..., a_k) \mid a_i \in t_{i\mathcal{P}(\mathbf{M})}(D_1, ..., D_n)), 1 \leq i \leq k\} s$ si $t = f(t_1, ..., t_k)$.

A un polynôme p de $Pol(F, \{x_1, ..., x_n\})$ est associée une fonction $p_{\mathcal{P}(\mathbf{M})} : \mathcal{P}(M)^n \longrightarrow \mathcal{P}(M)$ ainsi définie :
$p_{\mathcal{P}(\mathbf{M})}(D_1, ..., D_n) = \emptyset$ si $p = \varnothing$
$p_{\mathcal{P}(\mathbf{M})}(D_1, ..., D_n) = t_{1\mathcal{P}(\mathbf{M})}(D_1, ..., D_n) \cup ... \cup t_{k\mathcal{P}(\mathbf{M})}(D_1, ..., D_n)$ si $p = t_1 \cup t_2 \cup ... \cup t_k$.

Un *système d'équations* est un ensemble d'équations de la forme $S = \{u_1 = p_1, ..., u_n = p_n\}$ où $u_1, ..., u_n$ sont des variables et $p_1, ..., p_n \in Pol(F, \{u_1, ..., u_n\})$.
Une *solution* de S dans $\mathbf{M}$ est un n-uplet $(U_1, ..., U_n)$ de parties de M tel que $U_i = p_{i\mathcal{P}(\mathbf{M})}(U_1, ..., U_n)$ pour tout i.

On écrira cela de façon plus synthétique $\overrightarrow{U} = S_{\mathbf{M}}(\overrightarrow{U})$ en notant $\overrightarrow{U}$ un n-uplet $(U_1, ..., U_n)$ et $S_{\mathbf{M}}$ la fonction : $\mathcal{P}(M)^n \longrightarrow \mathcal{P}(M)^n$ qui, à $D_1, ..., D_n \subseteq M$ associe le n-uplet $(p_{1\mathcal{P}(\mathbf{M})}(D_1, ..., D_n), ..., p_{n\mathcal{P}(\mathbf{M})}(D_1, ..., D_n))$.

Les propriétes dont nous aurons besoin sont rassemblées dans la proposition qui suit et qui utilise les objets et notations ci-dessus.

Proposition 3.3.1 : (1) Dans toute F-algèbre $\mathbf{M}$, un système d'équation S a une plus petite solution qui est la borne supérieure des itérés de la fonction $S_{\mathbf{M}}$ à partir de $(\emptyset, ..., \emptyset)$.

(2) Cette solution est constituée des ensembles des valeurs dans M des termes des composantes de la plus petite solution du système S dans la F-algèbre des termes $\mathbf{T}(F)$.

(3) Si h est un homomorphisme de F-algèbres : $\mathbf{M} \longrightarrow \mathbf{N}$, alors la plus petite solution de S dans $\mathbf{N}$ est l'image par h de celle de S dans $\mathbf{M}$.

(4) Si la F-algèbre $\mathbf{M}$ est finie, la plus petite solution d'un système peut être obtenue au terme d'un nombre fini d'itérations.

Précisons le point (1) : la plus petite solution de S dans $\mathbf{M}$ est l'union des n-uplets d'ensembles $S^i_{\mathbf{M}}(\emptyset, ..., \emptyset)$, $i \geq 0$, où l'union des n-uplets est définie par :

$$(U_1, ..., U_n) \cup (V_1, ..., V_n) = (U_1 \cup V_1, ..., U_n \cup V_n).$$

On trouvera la preuve dans [Cou97], [CouB] et des exemples traités en exercice (Exercice 6).

Pour simplifier les énoncés, nous écrirons *la solution* d'un système S dans $\mathbf{M}$ pour désigner la plus petite solution, et nous dirons que la première composante de la solution est *l'ensemble défini par* S dans $\mathbf{M}$. Cela revient à prendre systématiquement la première inconnue comme inconnue principale, ou *axiome* dans la terminologie des grammaires algébriques. Un *ensemble équationnel de* $\mathbf{M}$ est un ensemble défini par un système d'équations S. Toute composante de la solution d'un système est donc équationnel, il suffit de placer en tête l'inconnue correspondante.

Exemple : Dans la F-algèbre des graphes non orientés $\mathbf{G}$, où $F = \{\oplus, \otimes, \mathbf{1}\}$ l'équation :

$$U = (U \oplus U) \cup (U \otimes (U \otimes U)) \cup \mathbf{1}$$

définit un sous-ensemble $\mathcal{U}$ de l'ensemble des cographes, (cf Section 3.2). Si l'on interprète cette équation comme définissant un ensemble de mots de A^*, où A est l'alphabet $\{\oplus, \otimes, \mathbf{1}, (,)\}$, on obtient un ensemble de termes dont les plus courts sont :

$\mathbf{1}, (\mathbf{1} \oplus \mathbf{1}), (\mathbf{1} \otimes (\mathbf{1} \otimes \mathbf{1})), (\mathbf{1} \oplus (\mathbf{1} \oplus \mathbf{1})), (\mathbf{1} \oplus (\mathbf{1} \otimes (\mathbf{1} \otimes \mathbf{1}))), ...,$

$((\mathbf{1} \otimes (\mathbf{1} \otimes \mathbf{1})) \oplus (\mathbf{1} \otimes (\mathbf{1} \otimes \mathbf{1}))), ...$

Les graphes définis par ces termes sont les éléments de $\mathcal{U}$. On applique ici la Proposition 3.3.1(2) avec pour h l'homomorphisme qui évalue un terme (écrit avec les symboles $\oplus, \otimes, \mathbf{1}$ et des parenthèses pour lever les ambiguïtés résultant de la notation infixée) en le graphe correspondant. Il est clair que $\mathcal{U}$ est un ensemble de cographes. Cet exemple est complété par l'exercice 7.

Proposition 3.3.2 : Soit $\mathbf{M}$ une F-algèbre, D un ensemble d'opérations dérivées et **M'** la $(F \cup D)$-algèbre ainsi obtenue, dont le domaine est également M. Les ensembles équationnels de $\mathbf{M}$ et de **M'** sont les mêmes.

Preuve : Tout ensemble équationnel relativement à $\mathbf{M}$ l'est aussi relativement à **M'**. Inversement, soit L défini par un système S' écrit avec des opérations de F et de D. En remplaçant ces opérations dérivées par les termes qui les définissent, on obtient un système S écrit avec les seules opérations de F. En utilisant la condition qu'aucune variable n'apparait deux fois dans un terme qui définit une opération dérivée, on montre que $S'_{\mathbf{M}'} = S_{\mathbf{M}}$. Donc S et S' ont les mêmes solutions et L est équationnel pour $\mathbf{M}$. □

3.4 Grammaires de graphes

Dans cette sous-section, on va appliquer la notion générale de système d'équations récursives ensemblistes aux graphes, en utilisant les opérations sur les graphes avec sources définies dans la Section 3.1. On utilisera l'ensemble $\mathcal{F}$ où l'on placera aussi les constantes $\overrightarrow{\mathbf{ab}}$ qui désignent des arcs orientés. On désignera par **HR** la $\mathcal{F}$-algèbre correspondante des graphes avec sources et qui peuvent avoir simultanément des arêtes (non orientées) et des arcs. Un ensemble de graphes est **HR**-*équationnel* si c'est un ensemble équationnel de **HR**. Il existe d'autres types de grammaires de graphes et d'autres définitions de celles-ci. L'ouvrage édité par G. Rozenberg [GraGraBook] est très complet pour ces questions. Le terme "grammaire de graphes" désignera toujours dans ce chapitre un système d'équations à résoudre dans **HR**.

On dira qu'un graphe est *défini par un système* S si c'est un élément de l'ensemble défini dans **HR** par ce système.

Proposition 3.4.1 : Soit S est un système d'équations sur $\mathcal{F}$, et C le plus petit sous-ensemble de $\mathcal{C}$ tel que $\mathcal{F}_C$ contient tous les symboles de S. Cet ensemble est fini. Les graphes définis par S sont définis par des termes de $T(\mathcal{F}_C)$. Leurs types sont des parties de C.

Preuve : Puisque que S est fini, il est écrit avec un nombre fini d'opérations, et donc C est fini. Sa solution dans l'ensemble des termes de $T(\mathcal{F})$ ne comporte que des termes de $T(\mathcal{F}_C)$. Le résultat est une conséquence de la Proposition 3.3.1(2).□

Dans la proposition qui suit, on considère des graphes pouvant comporter des arêtes (non orientées) et des arcs (orientés).

Proposition 3.4.2 : Les graphes d'un ensemble équationnel sont de largeur arborescente bornée. Pour tout k, les ensembles $TWD(< k)$ et $PWD(< k)$ sont des ensembles équationnels.

Preuve : La première assertion est une conséquence immédiate de la proposition qui précède et du Théorème 3.1.1.

Pour construire une équation qui définit $TWD(< k)$, prenons $C = [k]$ comme ensemble d'étiquettes et considérons l'équation :

$T = T//T \cup \mathbf{fg}_1(T) \cup ... \cup \mathbf{fg}_k(T) \cup$

$\cup \mathbf{ren}_{h_1}(T) \cup ... \cup \mathbf{ren}_{h_p}(T) \cup \mathbf{1} \cup \mathbf{1}^{bcl} \cup \mathbf{12} \cup \overrightarrow{\mathbf{12}}, \quad (4)$

où $h_1, ..., h_p$ sont les $k!$ permutations de C. Tout graphe défini par cette équation est de largeur arborescente au plus $k-1$ d'après la Proposition 3.4.1 et le Théorème 3.1.1.

Inversement, un graphe dans $TWD(< k)$ est la valeur d'un terme écrit avec les opérations $//, \mathbf{fg}_A$, $\mathbf{ren}_h$ et les constantes $\mathbf{i}, \mathbf{i}^{bcl} \cup \mathbf{ij} \cup \overrightarrow{\mathbf{ij}}$ pour $A \subseteq C$, $i, j \in C$, $i \neq j$. Mais $\mathbf{fg}_A$ est la composition des opérations $\mathbf{fg}_i$ pour $i \in A$, $\mathbf{ij}$ définit le même graphe que $ren_h(\mathbf{12})$, pour une permutation h appropriée et de même $\mathbf{i}, \mathbf{i}^{bcl}$, $\overrightarrow{\mathbf{ij}}$. Un tel graphe est défini par une composition des opérations et constantes qui figurent dans l'équation (4). Cette équation définit donc l'ensemble $TWD(< k)$.

Pour définir l'ensemble $PWD(< k)$ des graphes de largeur arborescente linéaire inférieure à k, on utilise le système d'équations :

$P = P//C \cup C \cup \mathbf{fg}_1(P) \cup ... \cup \mathbf{fg}_k(P) \cup \mathbf{ren}_{h_1}(P) \cup ... \cup \mathbf{ren}_{h_p}(P)$

$C = D \cup \mathbf{ren}_{h_1}(D) \cup ... \cup \mathbf{ren}_{h_p}(D)$

$D = \mathbf{1} \cup \mathbf{1}^{bcl} \cup \mathbf{12} \cup \overrightarrow{\mathbf{12}}$.

Avec le Corollaire 3.1.1 et un raisonnement analogue, on montre que $PWD(< k)$ est l'ensemble défini par ce système.□

Remarque : Les cographes forment un ensemble équationnel pour une algèbre de graphes différente. Voir [Cou97] pour une autre algèbre de graphes pour laquelle les cographes et bien d'autres ensembles sont équationnels.

En application de la Proposition 3.3.2, on peut écrire des systèmes d'équations en utilisant des opérations dérivées. Nous allons donner quelques exemples.

Exemple: Graphes de Halin

Les arbres avec racines sont définis au moyen de l'équation :

$A = A//A \cup ext(A) \cup \mathbf{r}$

écrite avec l'opération dérivée *ext* (pour *extension*) :

$ext(G) = \mathbf{fg}_n(\mathbf{nr}//\mathbf{ren}_{n \longleftrightarrow r}(G))$.

Les *graphes de Halin* sont les graphes planaires avec au moins 4 sommets, constitués d'un arbre représenté dans le plan, sans sommet de degré 2, et d'un cycle qui "entoure cet arbre" en reliant toutes ses feuilles. Nous allons définir une grammaire qui engendre un ensemble voisin de celui des graphes de Halin.

Les équations suivantes utilisent les étiquettes de sources r, n, g, d :

$D = B//\mathbf{gd}$
$B = Conc(B,B) \cup ext(B) \cup \mathbf{rg}//\mathbf{rd}//\mathbf{gd}, (**)$

où $Conc$ est l'opération de "concaténation" définie ainsi :
$Conc(G,H) = \mathbf{fg}_n([\mathbf{fg}_d(G//\mathbf{nd})]//\mathbf{ren}_{g\longleftrightarrow n}(H))$.

L'équation (**) définit des graphes planaires avec 3 sources désignées par r, g, d qui sont des arbres dont les feuilles sont reliées par un chemin.L'étiquette r désigne la racine, g la feuille la plus à gauche et d la feuille la plus à droite de l'arbre sous-jacent. Ces graphes sont schématisés en haut de la figure 4. La concaténation de deux de ces graphes produit le graphe en bas de la figure 4. Les graphes intermédiares montrent des étapes de la définition : $G' = \mathbf{fg}_d(G//\mathbf{nd})$ et $H' = \mathbf{ren}_{g\longleftrightarrow n}(H)$. Pour compléter le cycle qui relie les feuilles, on ajoute une arête entre les feuilles extrêmes, ce que réalise la première équation.

Les graphes engendrés par D sont, à quelques détails près des graphes de Halin. Tout graphe de Halin est mineur d'un de ces graphes et donc les graphes de Halin sont de largeur arborescente au plus 3. Cet exemple est complété dans l'exercice 11.

3.5 Grammaires : deux difficultés

La Proposition 3.4.2 montre que l'ensemble de tous les graphes finis n'est pas engendré par une unique grammaire. Il en va de même de l'ensemble des graphes planaires et des graphes de degré maximum d, pour tout $d \geq 3$. A l'inverse, l'ensemble de tous les mots sur un alphabet fini est engendré par une grammaire algébrique.

Une autre difficulté est liée à la complexité de l'analyse syntaxique. Alors que toute grammaire algébrique possède un algorithme polynomial d'analyse syntaxique, certains ensembles HR-équationnels de graphes sont NP-complets. Un exemple en est fourni par l'ensemble des graphes de *largeur de bande circulaire* au plus 2 (*cyclic bandwith*) [LVW] ; voir aussi l'exercice 11. Cet exemple est évidemment un peu particulier (on peut reconnaître en temps polynomial les graphes connexes de largeur de bande circulaire au plus 2) et beaucoup de grammaires de graphes ont des algorithmes d'analyse syntaxique polynomiaux et même linéraires. C'est le cas notamment des grammaires qui définissent $TWD(\leq k)$ d'après le résultat de Bodlaender présenté dans la Proposition 2.4.3, car la transformation d'une décomposition arborescente en un terme et son inverse sont réalisables en temps linéaire, et la grammaire de la Proposition 3.4.2 engendre tous les termes qui définissent des graphes de largeur arborescente au plus k. Il en va de même des grammaires construites par le Corollaire 4.3.3.

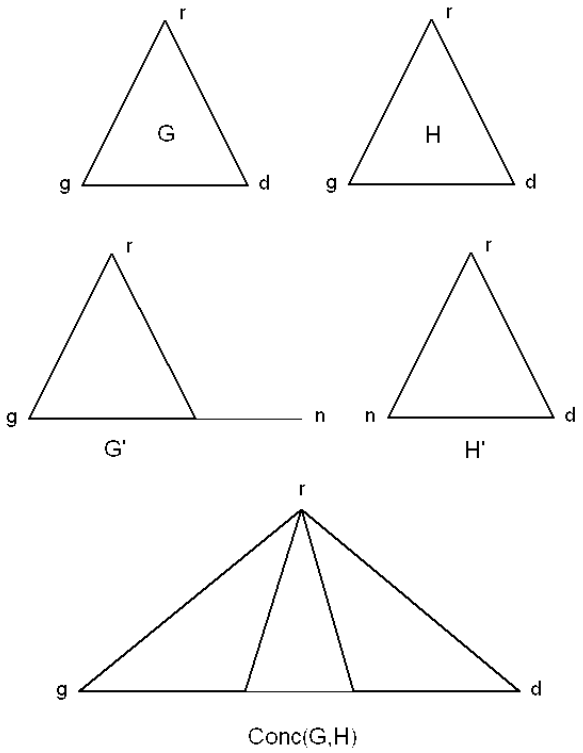

Figure 4: The operation *Conc*

4 Preuves et calculs inductifs

Cette section montre comment on peut utiliser la structuration des graphes définie par leurs représentations par des termes de $T(F)$ pour vérifier des propriétés ou effectuer des calculs sur ces graphes. Il en résultera des méthodes de construction de grammaires et d'algorithmes polynomiaux pour des classes de graphes structurés.

4.1 Exemples de preuves et de calculs inductifs

Propriétés des graphes série-parallèles.

Les graphes série-parallèles sont définis par l'équation : $S = \mathbf{12} \cup S//S \cup S \bullet S$, où $\bullet$ est la composition en série (voir section 3.1). Pour montrer que tout graphe série-parallèle est connexe, il suffit d'observer que $\mathbf{12}$ est connexe, et que si G et H sont des 2-graphes connexes, alors $G//H$ et $G \bullet H$ sont connexes. (Cela résulte de la Proposition 3.3.1(2)). Pour montrer que tout graphe série-parallèle est planaire, on peut tenter de faire de même, mais on rencontre une difficulté : il n'est pas vrai que si G et H sont des 2-graphes planaires, alors $G//H$ est planaire. Un contre-exemple est fourni par $\mathbf{12}$ et le graphe K_5 moins un arc, les extrémités de l'arc supprimé étant les sources 1 et 2. Il faut raisonner inductivement sur une propriété plus forte que la conclusion. On pourra prendre comme propriété :

"G possède un plongement dans le plan tel que les deux sources soient sur le bord d'une même face", ou encore

"$G//\mathbf{12}$ est planaire".

On est dans la situation classique où une preuve d'une propriété P se fait par récurrence sur une propriété de la forme $P \wedge Q$, où Q est une *propriété auxiliaire*. La *Logique du Second-Ordre Monadique* permet de systématiser cette observation et de générer automatiquement (au moins sur le plan théorique) la propriété auxiliaire nécessaire à la conduite d'une preuve.

Recherche d'une 3-coloration.

Prenons l'exemple du problème NP-complet de la recherche d'une 3-coloration d'un graphe. Soit G un graphe de largeur arborescente au plus $k-1$ (k est fixé), donné un terme de $T(\mathcal{F}_C)$ où C est un ensemble fixé de k étiquettes. Pour exploiter cette donnée, il suffit de savoir vérifier la 3-colorabilité d'un graphe de la forme $H//K$ ou $\mathbf{ren}_h(H)$ ou $\mathbf{fg}_A(H)$ en fonction de la 3-colorabilité et de propriétés auxiliaires pour H et K. On va considérer des graphes avec sources, de type inclus dans C, donc des graphes de largeur arborescente au plus $k-1$. On ne manipulera que des graphes ayant au moins une source, ce qui ne nuit pas à la généralité.

Pour tout graphe $G = (X, E)$, on note $Col(G)$ l'ensemble des 3-coloriages, c'est à dire des fonctions $c : X \longrightarrow \{1, 2, 3\}$ telles que $c(x) \neq c(y)$ pour x adjacent à y. On pourrait calculer $Col(G)$ par induction sur la structure d'un terme qui définit G, mais on aurait à manipuler des ensembles de taille éventuellement exponentielle en $s(G)$ et l'on n'obtiendrait pas ainsi un algorithme efficace. On va extraire de $Col(G)$ une information finie de taille bornée qui « passe à l'induction ».

À un coloriage c d'un graphe G, on associe sa restriction aux sources, c'est à dire l'application $c^{\#} = c \circ \mathbf{src}_G : \tau(G) \longrightarrow \{1, 2, 3\}$ qui associe à chaque étiquette de source la couleur du sommet correspondant. On note $Col^{\#}(G)$ l'ensemble de ces fonctions $c^{\#}$pour tous $c \in Col(G)$. Contrairement à $Col(G)$, l'ensemble $Col^{\#}(G)$ est partie d'un ensemble fini dont la taille (grande certes) ne dépend que de k.

Le point clé est que l'on peut déterminer $Col^{\#}(H//K)$, $Col^{\#}(\mathbf{fg}_A(H))$, et $Col^{\#}(\mathbf{ren}_h(H))$ en fonction de $Col^{\#}(H)$ et de $Col^{\#}(K)$. Pour un graphe de base B, on calcule directement $Col^{\#}(B)$. Une fois connu un terme t de $T(\mathcal{F}_C)$ qui définit G on peut calculer l'ensemble $Col^{\#}(H)$ pour tout graphe H défini par un sous-terme de t, et par induction montante dans t. Selon que l'ensemble $Col^{\#}(G)$ associé à la racine de l'arbre t est vide ou non, on obtient la réponse que G n'est pas ou est 3-coloriable respectivement. Et ceci par un algorithme linéaire en temps par rapport à la taille du terme t supposé connu, et pour un nombre d'étiquettes maximal k fixé.

En vue d'une application à d'autres problèmes, il importe de noter que les informations à calculer inductivement restent dans un ensemble fini, même très grand. Cette méthode s'applique à bien d'autres problèmes. En particulier à la recherche d'un cycle Hamiltonien optimal, et plus généralement à tout problème de graphe que l'on peut exprimer par une formule de la *logique du second-ordre monadique*, c'est à dire d'une formule logique écrite avec des quantifications existentielles et universelles sur des variables désignant des sommets, des arcs, des ensembles de sommets et des ensembles d'arcs. x Nous présenterons plus loin ce langage.

4.2 Ensembles inductifs de propriétés des éléments d'une F-algèbre.

Nous nous plaçons dans le cadre algébrique général de la section 3.3.

Définition 4.2.1 : **Ensembles inductifs de propriétés.**

Soit $\mathbf{M}$ une F-algèbre. Une *propriété des éléments de* $\mathbf{M}$ est une fonction totale $P : M \longrightarrow \{Vrai, Faux\}$. Soit $\mathcal{P} = \{P_1, ..., P_m\}$ un ensemble fini de propriétés des éléments de $\mathbf{M}$. Pour tout $d \in M$, on pose $\overrightarrow{\mathcal{P}}(d) = (P_1(d), ..., P_m(d)) \in \{Vrai, Faux\}^m$.

On dit que $\mathcal{P}$ est *inductif relativement à* F si pour toute fonction $f \in F$ d'arité k, il existe une fonction $f^{\#}$: $(\{Vrai, Faux\}^m)^k \longrightarrow \{Vrai, Faux\}^m$ telle que pour tous $d_1, .., d_k \in M$,

$$\overrightarrow{\mathcal{P}}(f_{\mathbf{M}}(d_1, .., d_k)) = f^{\#}(\overrightarrow{\mathcal{P}}(d_1), ..., \overrightarrow{\mathcal{P}}(d_k)).$$

Concrètement, cela équivaut à dire que, pour tous i et f, il existe une fonction booléenne $f_i^{\#}$ et des propriétés $P_{1,1}, P_{1,2}, ..., P_{2,1}, ..., P_{k,1}, P_{k,2}, ... \in \mathcal{P}$ telles que pour tous $d_1, .., d_k \in M$:

$$P_i(f_{\mathbf{M}}(d_1, .., d_k)) = f_i^{\#}(P_{1,1}(d_1), P_{1,2}(d_1), ..., P_{2,1}(d_2), ..., P_{k,1}(d_k), P_{k,2}(d_k), ...).$$

Proposition 4.2.1 [Cou97, CouB] : Soit $\mathbf{M}$ une F-algèbre et $\mathcal{P}$ un ensemble de propriétés inductif relativement à F. Si L est un ensemble $\mathbf{M}$-équationnel, et $P \in \mathcal{P}$, alors $L \cap \{d \in M \mid P(d) = Vrai\}$ est $\mathbf{M}$-équationnel.

L'idée de la preuve est la suivante. Si L est défini par un système dont les inconnues sont $U_1, ..., U_n$, et la solution est $(L_1, ..., L_n)$, alors en utilisant les fonctions $f^{\#}$ (c'est à dire les fonctions $f_i^{\#}$) on construit un système S' dont les inconnues sont $U_{i,w}$, pour tous $i = 1, ..., n$ et $w \in \{Vrai, Faux\}^m$, tel que la composante de sa solution associée à la variable $U_{i,w}$ soit $\{d \in L_i \mid \overrightarrow{\mathcal{P}}(d) = w\}$. On complète la définition de S' en ajoutant comme première équation $W = ... \cup U_{1,w} \cup ...$ où l'union est étendue à tous les vecteurs w dont la j-ème composante est $Vrai$, j étant le rang de P dans la liste $\mathcal{P}$. Des exercices illustrent cette méthode. (Exercice 11). On rappelle que $t_{\mathbf{M}}$ désigne la valeur dans $\mathbf{M}$ d'un terme t.

Proposition 4.2.2 [Cou97, CouB] : Soit $\mathbf{M}$ une F-algèbre et $\mathcal{P}$ un ensemble de propriétés inductif relativement à F. Pour tout terme $t \in T(F)$, on peut calculer $\overrightarrow{\mathcal{P}}(t_{\mathbf{M}})$ en temps $O(\mid t \mid)$.

Preuve : On utilise un automate fini déterministe montant (voir pour les automates [DF] ou le livre en ligne [TATA]) qui calcule pour tout noeud u de t le vecteur $\overrightarrow{\mathcal{P}}((t/u)_{\mathbf{M}})$ où t/u est le sous-terme de t issu de u.□

L'exemple traité du 3-coloriage peut être formulé ainsi en termes de propriétés $\mathcal{F}_C$-inductives. On considère l'ensemble des propriétés $P_{D,\gamma}$ où D est une partie non vide de C, γ est un ensemble de fonctions : $T \longrightarrow \{1, 2, 3\}$. On définit $P_{D,\gamma}(G)$ comme $Vrai$ si et seulement si $\tau(G) = D$ et $\gamma \in Col^{\#}(G)$. L'ensemble des propriétés $P_{D,\gamma}$ est fini et il résulte des remarques formulées dans la section 4.1 qu'il est $\mathcal{F}_C$-inductif, et que G est 3-coloriable ssi $P_{\tau(G),\gamma}(G) = Vrai$ pour quelque γ.

4.3 Utilisation de la logique du second-ordre monadique

Un graphe avec sources $G = (Y, E)$ orienté et de type $\{a, b, ..., d\}$, peut être décrit sans ambiguïté par la structure logique :

$$\|G\| =< Y \cup E, inc_1, inc_2, a, b, ..., d >$$

où inc_1 et inc_2 sont les relations binaires *d'incidence*, c'est à dire les ensembles de couples (e, x) (resp.(e, y)) pour tous $e \in E$ avec $e : x \longrightarrow y$; les $a, b, ..., d$ sont des constantes qui désignent les sources, d'étiquettes respectives $a, b, ..., d$. Noter que inc_1 et inc_2 sont fonctionnelles. Pour un graphe non orienté, on utilisera :

$$\|G\| =< Y \cup E, inc, a, b, ..., d >$$

avec inc non fonctionnelle, définie comme l'ensemble des couples (e, x) pour tous $e \in E, y \in Y$ tels que $e : x - y$.

La *Logique du Second-Ordre Monadique* est l'extension de la *Logique du Premier Ordre* au moyen de variables désignant des sous-ensembles des domaines des structures considérées. (Pour la logique du premier ordre, on pourra consulter les livres [CL] ou[LR] ; pour la LSOM, voir [Cou97]). On se contentera de donner deux exemples significatifs de propriétés de graphes exprimables en LSOM.

Tout d'abord on définit une formule auxiliaire du premier ordre $Adj(u, v)$ qui exprime que u et v sont deux sommets sont distincts et adjacents :

$u \neq v \wedge \exists w[(inc_1(w, u) \wedge inc_2(w, v)) \vee (inc_1(w, v) \wedge inc_2(w, u))]$

Cette formule est prévue pour le cas d'un graphe orienté. Pour un graphe non orienté, on utilisera la formule suivante, toujours notée $Adj(u, v)$:

$u \neq v \wedge \exists w[inc(w, u) \wedge inc(w, v)]$

Le domaine de la structure $\|G\|$ mélange les sommets et les arcs ou arêtes. Un élément du domaine est un arc (resp. une arête) si et seulement si il satisfait la formule :

$\exists v(inc_1(u, v))$ (resp. $\exists v(inc(u, v))$). On notera $Som(u)$ sa négation, qui caractérise les sommets (éventuellement isolés).

La 3-coloration d'un graphe s'écrit au moyen de la formule γ_3 (où l'on remplace $Adj(u, v)$ par sa définition) :

$\exists X_1, X_2, X_3[\forall x(x \in X_1 \vee x \in X_2 \vee x \in X_3) \wedge$

$\forall x(\neg(x \in X_1 \wedge x \in X_2) \wedge \neg(x \in X_2 \wedge x \in X_3) \wedge \neg(x \in X_1 \wedge x \in X_3))$

$\wedge \forall u, v(Adj(u, v) \Longrightarrow \neg(u \in X_1 \wedge v \in X_1) \wedge \neg(u \in X_2 \wedge v \in X_2) \wedge \neg(u \in X_3 \wedge v \in X_3))]$.

Il est clair que qu'un graphe G est 3-coloriable si et seulement $\|G\| \models \gamma_3$. Pour chaque entier k, on peut écrire une formule γ_k qui exprime la k-colorabilité.

L'existence d'un chemin non orienté entre deux sommets distincts x et y est exprimée par la formule $\delta(x, y)$:

$\forall X[(\forall u, v\{(Adj(u, v) \wedge u \in X) \Longrightarrow v \in X\} \wedge x \in X) \Longrightarrow y \in X]$.

On en déduit une formule exprimant la connexité :

$\forall x, y[Som(x) \wedge Som(y) \Longrightarrow x = y \vee \delta(x, y)]$.

Des propriétés telles que le fait d'être un arbre, la forte connexité ou plus généralement les propriétés basées sur une fermeture transitive sont exprimables de cette façon.

L'existence d'un circuit Hamiltonien dans un graphe orienté s'exprime à l'aide de la formule $\exists X.\varphi(X)$ suivante :

où $\exists!$ est une abréviation de "il existe un et un seul" :

$\forall x(\exists! u, u \in X.inc_1(u, x)) \wedge (\exists! u, u \in X.inc_2(u, x))$.

La formule $\varphi(X)$ exprime, en tenant compte de ce que l'on ne considère que des graphes finis, que les arcs de l'ensemble X forment un ou plusieurs circuits disjoints qui couvrent tous les sommets du graphe. Pour écrire que X ne définit qu'un seul circuit, on utilise la formule $\psi(X)$ qui exprime que le sous-graphe induit par les arcs de X est connexe. La formule cherchée est alors :

$\exists X[\varphi(X) \wedge \psi(X)]$.

On peut démontrer que cette propriété n'est pas exprimable en LSOM sans quantifications sur les ensembles d'arcs. ([Cou97]). Le sigle MS2 indique la possibilité de quantification sur les ensembles d'arcs ou d'arêtes.

Lemme 4.3.1 ([Cou97]) : Pour tout graphe non orienté H, il existe une formule MS2 qui exprime qu'un graphe ne contient pas H comme mineur. Si H est un graphe simple, on peut construire une formule de ce type dans MS1.

(Voir aussi exercice 12).

Il en résulte que toute classe de graphes fermée par minoration est définissable en MS2, et même en MS1 si elle est fermée par fusion d'arêtes parallèles. Cela s'applique aux graphes planaires, aux graphes représentables sans croisements d'arcs sur des surfaces orientables ou non, aux classes $TWD(\leq k)$ et $PWD(\leq k)$ et à d'autres (voir [DF]).

Soit C un ensemble fini d'étiquettes et $\mathcal{F}_C$ l'ensemble fini des opérations et constantes définies avec les seules étiquettes de C. Soit $\mathbf{HR}_C$ la $\mathcal{F}_C$-algèbre dont le domaine est l'ensemble des graphes avec sources de type inclus dans C et les opérations sont celles de $\mathcal{F}_C$. Le domaine de $\mathbf{HR}_C$ contient les graphes définis par les termes de $T(\mathcal{F}_C)$ et d'autres, de largeur arborescente non bornée.

Théorème 4.3.1 [Cou97] : Soit C un ensemble fini d'étiquettes et P une propriété MS2 des graphes de type inclus dans C. On peut construire un ensemble fini de propriétés qui contient P et qui est $T(\mathcal{F}_C)$-inductif sur $\mathbf{HR}_C$.

Ce théorème associé aux propositions 4.2.1 et 4.2.2 admet le corollaire suivant, dont la preuve est immédiate :

Corollaire 4.3.2 : Soit L un ensemble **HR**-équationnel et P une propriété MS2. L'ensemble des éléments de L qui satisfont P est **HR**-équationnel, et l'on peut construire un système qui définit cet ensemble à partir de la formule qui définit P et du système qui définit L.

Corollaire 4.3.3 : Soit k un entier et P une propriété MS2. L'ensemble L des graphes de largeur arborescente au plus k qui satisfont P est **HR**-équationnel. On peut construire un systeme qui définit L dont le problème de l'analyse syntaxique est soluble en temps linéaire par rapport à la taille du graphe donné.

Preuve : On rappelle que l'analyse syntaxique pour $TWD(\leq k)$ est possible en temps linéaire (Proposition 2.4.3). L'analyse syntaxique pour une grammaire construite en application de 4.3.2 se fait en 2 temps : vérifier si le graphe est de largeur arborescente au plus k et construire un terme qui en témoigne, et ensuite, utiliser l'automate de la Proposition 4.2.2 sur ce terme pour vérifier si la propriété P est vraie. Il n'y a pas unicité du terme t, définissant G donné, mais il suffit de faire fonctionner l'automate sur un terme t quelconque. Le résultat obtenu sera le bon.□

L'applications aux graphes de largeur arborescente linéaire au plus k grâce aux mineurs interdits, notion exprimable en MS1 d'après le lemme 4.3.1.

4.4 Mise en oeuvre et applications

La mise en oeuvre pratique : ????????? A voir. Citer Klarlund ???

Concernant les applications, on citera l'article de Thorup [Tho] qui montre que des graphes des programmes sans GOTOs écrits dans les langages Pascal, Modula-2 et C ont une largeur arborescente au plus 6. Il en déduit des algorithmes polynomiaux d'allocation de registres qui fournissent des allocations utilisant au pire 4 fois le nombre optimum de registres. C'est un beau résultat si l'on considère que les problèmes sous-jacents sont NP-complets.
Autre exemple : Allocation de fréquences. A PRECISER.

D'autres applications sont présentées dans Bodlaender [Bod1].

5 Extensions de cette approche

Ce chapitre n'est qu'une introduction à un domaine de la Théorie des graphes en constante expansion. Parmi les nombreuses directions de recherche on pourra citer sans prétendre être exhaustif, la recherche des valeurs les plus précises

possible de la largeur arborescente pour des graphes dotés de propriétés particulières ou construits de telle ou telle façon, la comparaison avec des notions de structuration proches telle que la *largeur de branche* (*branch-width*) [RS-GM13], l'extension des résultats aux matroïdes, qui peuvent eux aussi être décomposés de manière arborescente, l'algorithmique, séquentielle et parallèle associée, les raffinements de la notion de décomposition arborescentes, dont on a vu quelques exemples dans la section 2.4. La notion de largeur arborescente a des formulations équivalentes en termes de *stratégies d'exploration des graphes* : voir Bodlaender [Bod1], [Fra]?? (A préciser).

Il existe une autre mesure de complexité des graphes, liée à un autre type de structuration (arborescente) et qui donne lieu à des algorithmes polynomiaux pour des problèmes exprimés en Logique du Second-ordre monadique (mais sans quantifications sur les ensembles d'arcs et d'arêtes). Il s'agit de la *largeur de clique* (*clique-width*) définie et étudiée dans [CouOla], [CMR]. Cette notion est plus puissante en ce sens que la largeur arborescente en ce sens que tout ensemble de graphes de largeur arborescente bornée est de largeur de clique bornée, alors que l'inverse n'est pas vrai (les cographes sont de largeur de clique 2 et de largeur arborescente non bornée.) Les résultats récents de Oum et Seymour [OumSey] montrent que le corollaire 4.3.3 a une formulation analogue pour ces graphes, en remplaçant "linéaire" par "polynomial".

Les applications algorithmiques dépassent dépassent la simple vérification et s'étendent aux problèmes de dénombrement, d'optimisation, de calcul de fiabilité dans les réseaux et d'énumération. Voir en particulier [CMR2].

6 Exercices

1) **Majorations de largeurs arborescentes.**

a) En fonction de quels paramètres (degré, largeur arborescente, diamètre,...) relatifs à un graphe G peut-on majorer la largeur arborescente de $Line(G)$, son *graphe des incidences entre les arêtes* (*line graph*) ?

b) Soit H un graphe obtenu à partir d'un graphe G par subdivisions itérées d'arêtes. Peut-on majorer la largeur arborescente de H en fonction de celle de G ?

2) **Graphes de fonctions**

Le graphe $G(f)$ d'une fonction partielle $f : V \longrightarrow V$ où V est un ensemble fini a pour sommets les éléments de V et pour arcs les couples $x \longrightarrow f(x)$ pour tous $x \in V$. Montrer que ces graphes ont une largeur arborescente bornée indépendante de f et V. Préciser son maximum. Montrez que l'ensemble des graphes $G(f)$ est **HR**-équationnel.

3) **Propriété de Helly.**

La *Propriété de Helly* pour les arbres (graphes non orientés, connexes, sans cycles) énonce que si k sous-arbres d'un arbre sont 2 à 2 d'intersection non vide, alors l'intersection de ces k sous-arbres est non vide. Montrer que si l'intersection de 2 sous-arbres est non vide, c'est un arbre. Montrer que si 3 sous-arbres sont 2 à 2 d'intersection non vide, alors leur intersection est un sous-arbre non vide. Montrer la propriété de Helly par récurrence sur k.

4) **Sur les mineurs.**

a) Montrer la transitivité de la notion de mineur (Définition 2.3.1). [On montrera que si G est sous-graphe de $H \setminus F$, alors $G = H' \setminus F'$ où H' est un sous-graphe de H.]

b) Montrer que tout graphe planaire G est mineur d'une grille carrée $G_{n\times n}$ suffisamment grande. [On montrera que G est un mineur d'un graphe planaire H de degré maximum 3 et on utilisera un arbre recouvrant en profondeur de H.]

5) **Construction du graphe valeur d'un terme**

Le but est de définir une construction concrète du graphe $val(t)$ pour $t \in T(\mathcal{F}_C)$, C fini.

Une occurrence d'un symbole dans t peut être définie comme un entier qui représente sa position, le terme t étant écrit comme un mot. A t on associe l'ensemble $E(t)$ des couples $(u, 0)$ où u est une occurrence de $\mathbf{a}^{bcl}$ ou de $\mathbf{ab}$ ou de $\overrightarrow{\mathbf{ab}}$. Ces couples seront les arêtes du graphe à construire. On définit $V(t)$ comme l'ensemble des couples $(u, 1)$ tels que u est une occurrence de $\mathbf{a}$ ou de $\mathbf{a}^{bcl}$, et des couples $(u, 1), (u, 2)$ tels que u est une occurrence de $\mathbf{ab}$ ou de $\overrightarrow{\mathbf{ab}}$. Ces couples seront les sommets du graphe. Ainsi, $(u, 1)$ et $(u, 2)$ seront la a-source et la b-source respectivement de l'arête définie par l'occurrence u de la constante $\mathbf{ab}$ (ou de $\overrightarrow{\mathbf{ab}}$). À cause des opérateurs de composition parallèle qui identifient des sources de mêmes étiquettes, un même sommet du graphe $G(t)$ sera le plus souvent défini par plusieurs éléments de $V(t)$. On définira donc une relation d'équivalence $\approx$ sur $V(t)$ qui signifie que deux couples représentent le même sommet. Les sommets de G comme des classes d'équivalence de cette relation.

i) Dans le cas particulier du terme $t = \overrightarrow{\mathbf{cd}}//ren_{a\longleftrightarrow c}(\overrightarrow{\mathbf{ab}}//\mathbf{fg}_b(\overrightarrow{\mathbf{ab}}//\overrightarrow{\mathbf{bc}}))$, définir les ensembles $E(t), V(t)$ et l'équivalence $\approx$.

ii) Montrer que l'équivalence $(u, i) \approx (v, j)$ ne dépend que des constantes et des opérations unaires correspondant aux noeuds du chemin qui relie u à v dans l'arbre syntaxique du terme t.

iii) Définir précisément cette équivalence en termes de l'appartenance à des langages rationnels des mots reliant u et v à leur plus grand ancêtre commun dans t.

[On commencera par traiter les questions 2 et 3 dans le cas des termes sans renommages de sources].

6) **Systèmes d'équations**

A) *Etude d'une grammaire algébrique.*

On considère le système d'équation $S =< U = aUT \cup b\ ;\ T = bUUa \cup cTc >$ associé à la grammaire algébrique $G =< U \longrightarrow aUT\ ;\ U \longrightarrow b\ \ ;T \longrightarrow bUUa\ ;\ T \longrightarrow b >$ qui définit le langage $L(G,U) \subseteq \{a,b,c\}^*$. Soit $\mathbf{M} =< \{a,b,c\}^*, *, \varepsilon, a, b, c >$. Démontrer que le système $S_{\mathbf{M}}$ a une plus petite solution. Calculer les itérés $S^i_{\mathbf{M}}(\varnothing,\varnothing)$ pour $i = 0, 1, ..., 5$. Caractériser, pour tout i les mots de $S^i_{\mathbf{M}}(\varnothing,\varnothing)$ en termes de leurs dérivations relativement à la grammaire G.

A tout mot u de $\{a,b,c\}^*$ on associe le triplet $h(u) = (\alpha,\beta,\gamma) \in \{0,1\}^3$où α est la parité du nombre d'occurrences dans u de la lettre a, et β,γ de même pour b et c.

Définir une algèbre finie $\mathbf{N}$ telle que h soit un homomorphisme. Déterminer la solution de S dans $\mathbf{N}$ (cf. la Proposition 3.3.1(4)). Comment cette solution est reliée à la solution de $S_{\mathbf{M}}$ dans $\mathbf{M}$? (cf. la Proposition 3.3.1(4)).

B) *Preuve de la Proposition 3.3.1.*

Pour montrer (1) on montrera d'abord que si $U_i \subseteq V_i$, alors $S_{\mathbf{M}}(U_1,..,U_n) \subseteq S_{\mathbf{M}}(V_1,..,V_n)$ ($S_{\mathbf{M}}$ est une fonction croissante), et ensuite que :

$\bigcup_i S_{\mathbf{M}}(U_1^{(i)},..,U_n^{(i)}) = S_{\mathbf{M}}(\bigcup_i U_1^{(i)},..,\bigcup_i U_n^{(i)})$ si $U_j^{(i)} \subseteq U_j^{(i+1)}$) pour tous i et j ($S_{\mathbf{M}}$ est une fonction continue pour l'ordre d'inclusion).

On pourra en déduire (4). On pourra ensuite montrer (3) en montrant d'abord que :

$h(S_{\mathbf{M}}(U_1,...,U_n)) \subseteq S_{\mathbf{N}}(h(U_1,),...,h(U_n))$.

Pour montrer (2), on montrera que tous les mots définis par la solution de S dans la F-algèbre des termes ... sont des termes bien formés. Définir h non ambiguïté etc...

C) *Calcul des types des graphes d'un ensemble équationnel.*

Montrer qu'il existe un algorithme qui permet de calculer, pour tout ensemble de graphes **HR**-équationnel donné par un système d'équations l'ensemble des types des graphes de cet ensemble.

[On définira une algèbre finie $\mathbf{N}$ telle que la fonction τ est un homomorphisme de la $\mathcal{F}_C$-algèbre $\mathbf{HR}_C$. dans $\mathbf{N}$. On utilisera la Proposition 3.3.1].

7) **Cographes**

Construire une grammaire qui engendre les cographes dont le nombre de sommets est un multiple de 3.

Démontrer que les systèmes :

$$\{X = X \otimes (X \oplus X) + \mathbf{1}\ \} \text{ et } \{X = Y \otimes X + \mathbf{1}\ ;\ Y = X \oplus X\}$$

définissent le même ensemble X de cographes.

Finir exemple Section grammaires

8) **Grammaires de graphes qui engendrent des mots.**

Tout mot peut être considéré comme un graphe orienté dont les arcs sont étiquetés par des lettres. Le mot vide correspond à un sommet isolé. Tout langage

algébrique $\subseteq \{a,b,c\}^*$ est un ensemble de graphes HR-équationnel. Montrer que les langages non algébriques $\{a^nb^nc^n/n>0\}$ et $\{ww/w\in\{a,b,c\}^*\}$ sont HR-équationnels.

9) **Nombres de Strahler**

Soit f d'arité 2 et a,b des constantes. On définit le *nombre de Strahler* de $t\in T(\{f,a,b\})$ par l'induction suivante :

$NS(a)=NS(b)=1$

$NS(f(t_1,t_2))=NS(t_1)+1$ si $NS(t_1)=NS(t_2)$,

$NS(f(t_1,t_2))=Max\{NS(t_1),NS(t_2)\}$ si $NS(t_1)\neq NS(t_2)$.

Construire un système d'équations Σ_k qui définit l'ensemble des arbres de nombre de Strahler au plus k.

On donnera une construction telle la taille de Σ_k soit $O(k)$. (La *taille d'un système* est la somme des longueurs de ses équations, tous les symboles étant comptés pour 1, même dans le cas de symboles x_k, pour k grand, que l'on pourrait compter comme de longueur $log(k)$.)

10) **Propriétés inductives**

Déterminer les propriétés inductives permettant de tester sur les arbres syntaxiques des graphes de largeur arborescente au plus 3 si ces graphes :

- sont connexes,
- sont sans circuit,
- ont un circuit Hamiltonien (qui passe par chaque sommet une fois et une seule ; un 8 n'est pas Hamiltonien).

11) **Constructions de grammaires**

a) Construire une grammaire qui engendre les graphes série-parallèles 2-coloriables.

b) En quoi l'ensemble des graphes engendrés par la grammaire de l exemple ??? diffère-t-il de l'ensemble des graphes de Halin ? Modifier la construction donnée pour engendrer exactement les graphes de Halin.

c) Dans ce qui suit on dit que G est un *sous-graphe* de H si les ensembles de sommets sont les mêmes et tout arc ou arête de G est un arc ou une arête de H. Montrer que si L est un ensemble de graphes **HR**-équationnel alors l'ensemble M de ses sous-graphes est aussi **HR**-équationnel, défini par un système que l'on construira à partir d'un système qui définit L.

d) La *largeur de bande* (*bandwidth*) d'un graphe G est le plus petit entier k tel que l'on peut ordonner les sommets de G en une suite $x_1,...,x_n$ telle que si $i<j$ et x_i-x_j alors $j-i\leq k$. Construire un système d'équations qui définit l'ensemble des graphes simples non orientés de largeur de bande au plus k.

e) La *largeur de bande circulaire* (*cyclic bandwidth*) d'un graphe G est le plus petit entier k tel que l'on peut ordonner les sommets de G en une suite $x_1,...,x_n$ telle que si $i<j$ et x_i-x_j alors $Min\{j-i,n-(j-i)\}\leq k$. Construire un système d'équations qui définit l'ensemble des graphes simples non orientés de largeur de bande circulaire au plus k.

[Pour répondre aux questions d) et e) on pourra commencer par caractériser les ensembles de graphes largeur de bande (circulaire) au plus k qui sont maximaux pour la relation de sous-graphe. On construira des systèmes pour ces ensembles et on utilisera ensuite le résultat de la question c).]

12) **Mineurs**
Démontrer le Lemme 4.3.1. [Indications : à faire]

7 Bibliographie

[Bod1] H. Bodlaender, A Partial k-Arboretum of Graphs with Bounded Treewidth. *Theoret. Comput. Sci.* **209** (1998) 1-45

[Bod2] H.Bodlaender, A Tourist Guide through Treewidth, *Acta Cybernetica* **11** (1993) 1-22

[BodHag] H. Bodlaender, T. Hagerup: Parallel Algorithms with Optimal Speedup for Bounded Treewidth. *SIAM J. Comput.* **27** (1998) 1725-1746

[Bou] N. Bourbaki, *Algèbre*, Hermann, Paris, 1970

[BLS] A. Brandstädt, V. Lê, J. Spinrad, *Graph Classes: A Survey*, SIAM, Philadelphia, 1999.

[CL] R. Cori, D. Lascar, *Logique mathématique : cours et exercices, volumes 1 et 2*, Collection : Axiomes, Masson, Paris, 1993

[CouB] B. Courcelle, Basic notions of Universal Algebra for Language Theory and Graph Grammars, *Theoretical Computer Science* **163** (1996) 1-54.

[Cou97] B. Courcelle, The expression of graph properties and graph transformations in monadic second-order logic, Chapter 5 of the "Handbook of graph grammars and computing by graph transformations, Vol. 1 : Foundations", G. Rozenberg ed., World Scientific (New-Jersey, London), 1997, pp. 313-400.

[CouXV] B. Courcelle,The monadic second-order logic of graphs XV : On a conjecture by D. Seese. A paraître dans le *Journal of Applied Logic*, 2005.

[CouOum] B. Courcelle, S. Oum, Vertex-minors, monadic second-order logic and a conjecture by Seese, soumis, July 2004.

[CMR] B. Courcelle, J. Makowsky, U. Rotics. Linear time solvable optimization problems on certain structured graph families, *Theory of Computer Systems* **33** (2000) 125-150.

[CMR2] B. Courcelle, J.Makowsky, U. Rotics, On the Fixed Parameter Complexity of Graph Enumeration Problems Definable in Monadic Second Order Logic, *Discrete Applied Mathematics* **108** (2001) 23-52.

[CouVan] B. Courcelle, R. Vanicat, Query efficient implementations of graphs of bounded clique-width, *Discrete Applied Mathematics* **131** (2003) 129-150

[CouOla] B. Courcelle, S. Olariu, Upper bounds to the clique-width of graphs, *Discrete Applied Mathematics* **101** (2000) 77-114.

[Dje] S. Djelloul, Version definitive ICGT

[DF] R. Downey et M. Fellows, *Parameterized Complexity*, Springer-Verlag, 1999

[Fra] P. Fraignaud, ?????????????

[GraGraBook] G. Rozenberg, ed., *Handbook of graph grammars and computing by graph transformations*, Vol. 1 : Foundations, World Scientific (New-Jersey, London), 1997.

[Hedet] S. Hedetniemi, References on partial k-trees, *Discrete Applied Maths* **54** (1994) 281-290.

[Lag] J. Lagergren, Upper bounds on the size of obstructions and intertwines, *Journal of Combinatorial Theory Series B*, **73** (1998) 7–40

[LVW], J.Leung, O.Vornberger, J. Witthoff, On some. variants of the bandwidth minimization problem, *SIAM J. Comput.* **13** (1984) 650-667.

[Die] R. Diestel, *Graph Theory*, Second Edition, Springer-Verlag, 2000. Disponible gratuitement en lecture seule sur :
http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/

[LR] R.Lassaigne, M. de Rougemont, *Logique et complexité*, Hermes, Paris, 1996

[MT] B. Mohar, C.Thomassen, *Graphs on surfaces*, John Hopkins University Press, Baltimore, 2001.

[Rec] A. Recski: *Matroid theory and its applications in electric network theory and in statics*, Algorithms and Combinatorics Vol. **6**. Springer-Verlag, 1989

[RS-GM13] N. Robertson, P. Seymour, Graph Minors .XIII. The Disjoint Paths Problem, *Journal of Combinatorial Theory Series B* **63** (1995) 65-110.

[RS-GM20] N. Robertson, P. Seymour, Graph Minors .XX. Wagner's conjecture, *Journal of Combinatorial Theory, Series B*, **92**,(2004) 325-357

[RST] N. Robertson, P. Seymour, R. Thomas, Quickly Excluding a Planar Graph, *Journal of Combinatorial Theory, Series B*, **62** (1994) 323-348

[Seese] D. Seese,The structure of the models of decidable monadic theories of graphs, *Annals of Pure and Applied Logic*, **53** (1991) 169-195

[Spin] J. Spinrad, *Efficient graph representations*, Fields Institute Monographs, American Mathematical Society, 2003

[TATA] Comon et al., *Tree Automata Techniques and Applications*, Livre en ligne, Consultable gratuitement,
http://www.grappa.univ-lille3.fr/tata/

[Tho] M. Thorup: All Structured Programs have Small Tree-Width and Good Register Allocation. *Inf. Comput.* **142** (1998) 159-181

[TUK] A. Takahashi, S. Ueno and Y. Kajitani, Minimal acyclic forbidden minors for the family of graphs with bounded path-width, *Discrete Mathematics* **127** (1994) 293-304

Références supplémentaires (A supprimer ?)

Ton Kloks: Treewidth of Circle Graphs. Int. J. Found. Comput. Sci. 7(2): 111-120 (1996)

Ton Kloks, Hans L. Bodlaender, Haiko Müller, Dieter Kratsch: Computing Treewidth and Minimum Fill-In: All You Need are the Minimal Separators. ESA 1993: 260-271Ton Kloks, Hans L. Bodlaender, Haiko Müller, Dieter Kratsch: Erratum: Computing Treewidth and Minimum Fill-In: All You Need are the Minimal Separators. ESA 1994: 508

Hans L. Bodlaender, Rolf H. Möhring: The Pathwidth and Treewidth of Cographs. SIAM J. Discrete Math. 6(2): 181-188 (1993)

Jens Gustedt: On the Pathwidth of Chordal Graphs. Discrete Applied Mathematics 45(3): 233-248 (1993)