

Propositionnalisation formelle et fouille de graphes

M. Liquière¹

¹ LIRMM

161 Rue ADA
34392 Montpellier Cedex 5
Liquiere@lirmm.fr

Résumé

Dans le cadre d'un mécanisme d'apprentissage relationnel, la propositionnalisation formelle a pour but de transformer la description structurelle des exemples en une description propositionnelle équivalente. Cette équivalence n'est pas basée sur l'expressivité ou la syntaxe mais sur l'équivalence des possibilités de discrimination du langage propositionnel par rapport au langage structurel et ce pour les exemples utilisés. Cette formalisation permet une caractérisation précise des motifs à sélectionner. Dans ce but nous utilisons une méthode performante de fouille de graphes pour l'extraction de motifs qui sont dans ce cas des chemins et des arbres répétés dans un ensemble de graphes. Ces motifs sont alors fournis à une méthode incrémentale, au niveau de l'ajout des motifs, qui sélectionne les motifs pertinents. Une expérimentation sur un jeu de données est ensuite présentée pour valider l'approche d'un point de vue pratique.

Mots Clef

Propositionnalisation, analyse de concepts formels, fouille de graphes.

Abstract

In the model of relational machine learning problem, the goal of a formal propositionnalisation method is to transform the relational problem into an equivalent propositional problem. This equivalence is not based on expressiveness but in the equivalence discrimination capacity of the two languages. This formalisation gives a precise definition of the non-redundant features. We use a graph mining method for feature extractions. The features are repeated paths and trees which occurs in a set of graphs. These features are then given to an incremental method which selects the important one. An experimental validation of the approach conclude the paper.

Keywords

Propositionnalisation, formal concept analysis, graph mining.

1 Introduction

Notre étude se situe dans le cadre de l'apprentissage structurel non supervisé. Pour notre présentation et expérimentation, les exemples seront décrits via des graphes étiquetés, toutefois la méthodologie présentée peut être utilisée dans le cas d'autres langages de description des exemples. Il est bien connu que l'apprentissage de concepts, dans ce cadre, doit gérer une complexité due à la taille de l'espace de recherche ainsi qu'à la complexité des opérations de généralisations et de projections utilisées. Dans ce but, les méthodes issues de la programmation logique inductive utilisent des techniques pour gérer la taille de l'espace de recherche soit en limitant la taille de cet espace (biais sur le langage) soit en utilisant des mesures locales (biais sur le parcours) [17]. Une alternative consiste à reformuler le problème initial en un problème booléen, ce qui permet ensuite d'appliquer des méthodes propositionnelles robustes. Ce changement de représentation est nommé *propositionnalisation* [14]. Dans un premier temps un ensemble de motifs est recherché, puis on reformule chacun des exemples en créant une relation binaire entre l'ensemble des exemples et l'ensemble de motifs trouvés. Cette relation indique, pour chaque motif, s'il est présent ou non dans l'exemple, ce qui donne la nouvelle description des exemples. L'avantage principal de la propositionnalisation est de permettre d'utiliser des méthodes robustes sur la nouvelle description des exemples. Les résultats (classification, discrimination), qui existaient dans le cadre de la description structurelle, peuvent alors être rapidement trouvés sur les exemples mais il faut pour cela que la nouvelle description préserve les regroupements que l'on cherchait et que l'ensemble des motifs sélectionnés soit de taille non exponentielle en fonction du nombre d'exemples.

Pour la recherche de motifs, les méthodes de fouille de données [1] et plus précisément de graphes [18], [8],[9],[22] se révèlent être très adaptées. Ces méthodes utilisent une structure par niveau de l'espace des descriptions pour accélérer la recherche de motifs. Ces motifs sont sélectionnés sur leurs nombre d'apparition dans l'ensemble des exemples (paramètre de validation). Les avantages principaux de ces méthodes sont leurs caractéristiques de quasi exhaustivité (borné par le paramètre de validation) et leur rapidité. Toutefois le nombre des motifs extraits et leurs redondances rendent ces résultats difficiles à exploiter.

Pour réduire la taille de cet ensemble deux techniques principales sont utilisées : la sélection de motifs et la réduction de l'ensemble des motifs.

1. La sélection de motifs a pour but la recherche d'un sous-ensemble de motifs suffisants pour la recherche de l'hypothèse cible. Ce problème a été largement exploré en apprentissage, principalement dans le but d'améliorer la fonction de décision. Ces méthodes utilisent principalement des mesures ou des biais sur le langage de description [13],[3].
2. La réduction de l'ensemble de motifs a pour but de réduire l'ensemble des motifs en éliminant des éléments logiquement redondants. Pour cela une définition de la redondance est donnée.

Dans le travail de [5] un motif booléen x est considéré comme non redondant pour un concept cible c s'il existe une paire d'exemples A et B tel que A et B diffèrent seulement dans leur assignement pour x (x est vrai pour l'un et faux pour l'autre) et $c(A) \neq c(B)$. (ou $c(A)$, $c(B)$ indique la classe associée à chacun des exemples).

Dans les papiers [16],[2] les auteurs prouvent que l'élimination de leurs motifs redondants ne compromet pas l'existence et la complétude d'une théorie consistante pour l'hypothèse cible dans le cas d'un problème de discrimination. Dans ce travail, un motif f est défini comme redondant face un autre motif g si g est vrai pour au moins les mêmes exemples que f et faux pour au moins les mêmes contre-exemples que f . En utilisant cette propriété l'algorithme Reduce [2] procède à une comparaison paire par paire des motifs. Dans ces deux cas il s'agit donc d'équivalence basée sur la qualité de discrimination d'un motif. La méthode proposée dans [27] utilise une propriété sur la corrélation entre un motif et un ensemble de motifs. Pour cela il utilise une procédure d'élimination nommée "Markov blanket filtering" et utilise des approximations pour gérer la complexité du calcul des corrélations. La propriété d'incrémentalité apportée par cette procédure de réduction sont très proches de celles que nous utilisons.

La méthode que nous présentons est une méthode de réduction mais se place dans le cadre de l'apprentissage non supervisé. Nous ne pouvons donc pas uti-

liser la même définition de la redondance que dans les deux premières approches. Dans notre cas un motif sera redondant par rapport à un ensemble de motifs existants, s'il ne permet pas de créer de nouvelles classes d'exemples. En utilisant des résultats de l'analyse de concepts formels, nous prouvons l'existence d'un ensemble minimal de motifs non redondants et proposons une méthode incrémentale (sur l'ajout de nouveaux motifs). Notre approche est donc algébrique plutôt que probabiliste comme dans le papier [27] et elle utilise des résultats de l'analyse de concepts formels (ACF) [11] que nous présentons dans le paragraphe suivant.

2 Cadre formel

L'analyse des concepts formels [26] repose sur la notion de concepts d'une relation binaire. Ces concepts lient entre eux des exemples et des attributs. Un concept sera donné par deux éléments : son extension (une sous partie de l'ensemble des exemples) et sa description ou intension (une sous partie de l'ensemble des attributs dans le cas propositionnel). Les exemples de la partie extension vérifient tous les attributs de la partie intension. Toutefois la définition de concept impose d'autres contraintes. L'idée est la suivante : si l'on met dans une même classe des exemples en considérant que ces exemples vérifient une propriété particulière : pourquoi ne pas mettre aussi dans cette classe tous les exemples vérifiant cette propriété ? Donc si l'on met dans la partie extension d'un concept un ensemble d'exemples vérifiant la partie description, il faut y mettre TOUS les exemples (de l'ensemble des exemples) qui vérifient cette description sinon on peut se poser la question "qu'elle est la raison qui permet de rejeter cet exemple de ce groupe d'exemples".

De manière duale, si l'on a une propriété P vérifiée par un ensemble d'exemples et que ces exemples vérifient tous une autre propriété P' non directement impliquée par P : pourquoi ne pas créer un nouvelle propriété $P \wedge P'$ pour la description de cette classe d'exemples ? Donc si l'on met dans la partie intension d'un concept un ensemble d'attributs vérifiés par la partie extension, il faut y mettre aussi TOUS les attributs (de l'ensemble des attributs) qui sont vrais pour ces exemples, sinon on peut se poser la question "Pourquoi ne pas ajouter la propriétés P' car elle est vérifiée par l'ensemble de ces exemples ?".

Ces notions de "bon sens" sont formellement définies par les définitions suivantes [11] :

Un *contexte* est un triplet $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ où \mathcal{E} est un ensemble, l'ensemble des exemples, \mathcal{A} un ensemble de propriétés (attributs) et \mathcal{R} une relation binaire entre \mathcal{E} et \mathcal{A} .

Pour un contexte $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ avec $e \in \mathcal{E}$, $a \in \mathcal{A}$, $e \mathcal{R} a$ s'interprète comme "l'exemple e vérifie l'attribut a " et "l'attribut a est vrai pour l'exemple e ".

On note : Pour $A \subseteq \mathcal{A}$ et $E \subseteq \mathcal{E}$.

$e(A) = \{e \in \mathcal{E} \mid e \mathcal{R} a, \forall a \in A\}$ (extension de A)

$d(E) = \{a \in \mathcal{A} \mid e \mathcal{R} a, \forall e \in E\}$. (description de E)

Un couple (E,A) avec $E \subseteq \mathcal{E}$ et $A \subseteq \mathcal{A}$ est un *concept* $\Leftrightarrow d(E)=A$ et $e(A)=E$.

Ce qui indique que pour un concept (E,A) , E contient tous les exemples $e \in \mathcal{E}$ vérifiant tous les attributs de A et A contient tous les attributs $a \in \mathcal{A}$ vrai pour les exemples de E . E est la partie extension du concept et A la partie intension du concept.

Il existe une relation d'ordre partiel entre les concepts. Pour un contexte $(\mathcal{E},\mathcal{A},\mathcal{R})$, la structure $\mathcal{L}(\mathcal{E},\mathcal{A},\mathcal{R})=\{(E,A) \mid (E,A) \text{ est un concept de } (\mathcal{E},\mathcal{A},\mathcal{R})\}$ ou les concepts sont ordonnés par $(E_1, A_1) \geq_{\mathcal{L}} (E_2, A_2) \Leftrightarrow E_2 \subseteq E_1$ ($\Leftrightarrow A_1 \subseteq A_2$) est un treillis nommé *treillis des concepts* [11] où treillis de Galois [21].

Rappel : Un ordre partiel $\mathcal{L}:(L, \leq_L, \vee, \wedge)$ est un treillis si chaque paire d'éléments (x,y) dans L a un unique infimum $x \wedge y$ et a un unique supremum $x \vee y$ dans L . \mathcal{L} est un treillis complet, ssi ¹ le supremum $\vee X$ et l'infimum $\wedge X$ existent pour tous sous-ensemble X de L [11]. Tout treillis fini et non vide est complet.

Dans le cas du treillis de Galois \mathcal{L} , chaque élément du treillis est un concept et ce treillis est complet. Pour deux concepts $(E_1, A_1), (E_2, A_2)$ de \mathcal{L} , les opérations \wedge et \vee sont :

$(E_1, A_1) \vee (E_2, A_2) = (E, A)$ avec $E=e(A_1 \cap A_2)$ et $A=d(E)$

$(E_1, A_1) \wedge (E_2, A_2) = (E, A)$ avec $A=d(E_1 \cap E_2)$ et $E=e(A)$.

L'opération \wedge est donc définie à partir de l'intersection des extensions et l'opération \vee est définie à partir de l'intersection des intensions.

Il s'agit en fait de mettre en relation deux ordres partiels (ici des treillis). L'espace des classements (regroupements) possibles des exemples qui est de manière générale le treillis des parties sur l'ensemble des exemples et l'espace des descriptions possibles qui est dans le cas d'une description par attribut, le treillis des parties sur l'ensemble des attributs. Comme exemple, le treillis de Galois de la relation \mathcal{R} de la figure 1 est présenté dans la figure 2.

Si l'on suit la définition de concepts donnée ci-dessus, on voit qu'il existe des parties de l'ensemble des exemples qui ne sont pas égales à une des parties extensions d'un concept. Il en va de même avec les partitions des attributs (voir figure 2). Comme cette relation est faite par l'intermédiaire de deux applications, la taille du treillis de Galois associé à une relation binaire est bornée par $\min(2^{|\mathcal{E}|}, 2^{|\mathcal{A}|})$. Comme nous le verrons plus loin, dans le cas structurel, nous pouvons avoir le même type de relation entre deux espaces. Toutefois dans ce cas la taille de l'espace de description est en général bien plus grande que la taille du treillis des parties sur les exemples.

On peut associer à chaque concept du treillis un rectangle maximal (au sens de l'inclusion) de 1 dans la relation \mathcal{R} , ceci en changeant l'ordre de lignes et des colonnes de \mathcal{R} si nécessaire.

	a_0	a_1	a_2	a_3	a_4	a_5	a_6
e_0	1	0	0	0	1	1	1
e_1	0	0	1	0	1	1	1
e_2	0	0	0	1	1	1	0
e_3	0	1	0	0	1	0	1
e_4	0	0	1	1	0	1	1

FIG. 1 – Relation \mathcal{R}

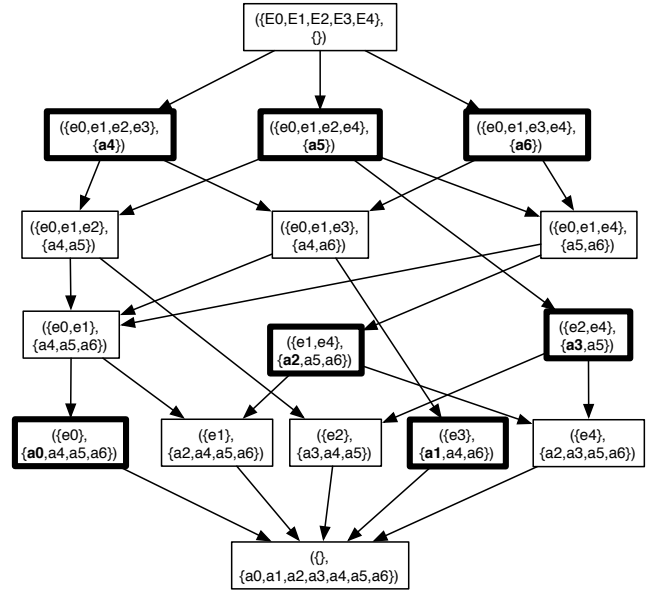


FIG. 2 – Le treillis de Galois de la relation \mathcal{R} (diagramme de Hasse)

3 Equivalence entre langages

Pour une relation binaire donnée, la construction du treillis de Galois associé permet d'avoir l'ensemble des concepts que l'on peut trouver sur cette relation. Il s'agit, si l'on considère les parties extensions des concepts, de l'ensemble de tous les regroupements des exemples possibles sans redondance (maximalité des parties extensions/intensions)

Pour un ensemble d'exemples \mathcal{E} les descriptions de ces exemples via un ensemble d'attributs \mathcal{A}_1 ou via un ensemble d'attributs \mathcal{A}_2 sont équivalentes si on peut obtenir exactement les mêmes regroupement (classification) des exemples en utilisant l'un ou l'autre des ensemble $\mathcal{A}_1, \mathcal{A}_2$ (et les relations \mathcal{R}_1 et \mathcal{R}_2 associées).

Dans le cadre formel des treillis de Galois, on définira que deux ensembles d'attributs $\mathcal{A}_1, \mathcal{A}_2$ utilisés dans les contextes $(\mathcal{E}, \mathcal{A}_1, \mathcal{R}_1)$ et $(\mathcal{E}, \mathcal{A}_2, \mathcal{R}_2)$ sont équivalents si $\mathcal{L}(\mathcal{E}, \mathcal{A}_1, \mathcal{R}_1) = \mathcal{L}(\mathcal{E}, \mathcal{A}_2, \mathcal{R}_2)$ ou $=$ indique l'isomorphisme entre les deux treillis. Un des cas simple ou cela se produit est quand deux attributs a,b sont vrais pour exactement le

¹si et seulement si

même ensemble d'exemple ($e(a)=e(b)$). Dans ce cas il n'est pas nécessaire de garder ces deux attributs pour obtenir le même treillis. Ce cas est un sous-cas d'une propriété générale basée sur une notion classique des treillis : les irréductibles.

Il y a différentes propriétés caractéristiques des éléments irréductibles d'un treillis. Nous donnons tout d'abord une présentation intuitive puis nous formaliserons cette notion. Les éléments \vee -irréductibles (resp. \wedge) dans un treillis sont les éléments ne pouvant pas être "recalculés" par une opération \vee (resp. \wedge) entre éléments du treillis. Il s'agit donc d'éléments essentiels pour le treillis dans le sens qu'ils permettent de reconstruire les autres éléments du treillis grâce aux opérations \vee et \wedge . Ces éléments vont être au coeur de notre recherche.

Une caractérisation plus formelle de ces éléments est la suivante :

Si l'on oriente le treillis suivant la relation d'ordre partiel (des éléments les plus grands vers les éléments les plus petits), les éléments \wedge -irréductibles (resp. \vee) dans un treillis sont les éléments ayant un et un seul prédécesseur direct (resp. successeur direct). Cette propriété permet de caractériser rapidement les éléments irréductibles si l'on a, comme donnée, un treillis. Dans le cas des treillis de Galois, à chaque concept \wedge -irréductibles (resp. \vee -irréductibles) est associé un attribut (exemple) tel que la fermeture de cet attribut donne la partie intension (extension) du concept. Dans le livre [11], une propriété permet de caractériser ces attributs de manière plus intéressante car elle permet d'avoir une construction de l'ensemble des attributs \wedge -irréductibles.

Propriété 1 (attribut \wedge -irréductible)

Dans un contexte $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ où $\forall a_1, a_2, a_1 \neq a_2 \in \mathcal{A}, e(a_1) \neq e(a_2), a \in \mathcal{A}$ est un attribut \wedge -irréductible $\Leftrightarrow \exists e \in \mathcal{E} \mid \neg (e \mathcal{R} a)$ et $\forall a' \neq a \in d(e(a))$ (fermeture de a), $e \mathcal{R} a'$. Pour un attribut \wedge -irréductible a nous notons $irre(a)$ l'ensemble, non vide, de tous les exemples vérifiant cette dernière propriété.

Il existe une propriété duale pour les exemples \vee -irréductibles.

On voit ici qu'un attribut \wedge -irréductible a est caractérisé par un ensemble d'exemples ($irre(a)$) pour lequel il n'est pas vrai, mais où ces exemples vérifient l'ensemble des attributs qui sont "proches" de a (proche dans le sens appartenant à la fermeture de a). C'est à dire les attributs qui sont vrais quand a est vrai (donc les attributs impliqués par a). En fait l'attribut a permet de séparer (et donc de caractériser) les exemples de $irre(a)$.

Propriété 2 (attribut et élément \wedge -irréductible)

Pour un contexte $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ où $\forall a_1, a_2, a_1 \neq a_2 \in \mathcal{A}, e(a_1) \neq e(a_2)$, pour chaque élément \wedge -irréductible (E, A) du treillis de Galois $\mathcal{L}(\mathcal{E}, \mathcal{A}, \mathcal{R})$, il existe un unique attribut \wedge -irréductible a tel que $E=e(a)$ et $A=d(e(a))$.

Dans la figure 2 les éléments \wedge -irréductibles sont indiqués par un cadre sombre, et l'attribut \wedge -irréductible de chacun de ces éléments est indiqué en gras. Par exemple le concept $(\{e_1, e_2\}, \{a_2, a_5, a_6\})$ est un élément \wedge -irréductible du treillis et l'attribut a_2 est l'attribut \wedge -irréductible de ce concept.

La propriété attribut \wedge -irréductible permet d'avoir un algorithme polynomial pour la recherche des attributs \wedge -irréductibles pour un contexte $(\mathcal{E}, \mathcal{A}, \mathcal{R})$. Toutefois dans le cadre de notre mécanisme de propositionnalisation le nombre d'attributs à traiter peut être immenses et le stockage et le traitement de l'ensemble de la relation \mathcal{R} peut être irréaliste. De plus, dans de nombreux mécanismes d'apprentissages, comme par exemple dans l'apprentissage par renforcement [25], les motifs à traiter n'arrivent qu'au fur et à mesure, c'est pourquoi nous avons étudié un usage incrémental (sur l'ajout d'attribut) de la propriété ci-dessus.

Definition (Contexte \wedge -réduit)

Un contexte $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ est dit \wedge -réduit ssi tous les attributs $a \in \mathcal{A}$ sont \wedge -irréductibles.

Propriété 3 (Contexte \wedge -réduit et ajout d'attribut)

Soit un contexte $(\mathcal{E}, \mathcal{A}_n, \mathcal{R}_n)$ \wedge -réduit.

Soit nouveau contexte $(\mathcal{E}, \mathcal{A}_{n+1}, \mathcal{R}_{n+1})$ formé à partir d'un nouvel attribut a_{n+1} tel que $\mathcal{A}_{n+1}=\mathcal{A}_n \cup a_{n+1}$ et \mathcal{R}_{n+1} est la relation binaire liant \mathcal{E} et \mathcal{A}_{n+1} .

Le contexte $(\mathcal{E}, \mathcal{A}_{n+1}, \mathcal{R}_{n+1})$ est \wedge -réduit ssi les trois conditions suivantes sont vérifiées.

1. $\nexists a_k \in \mathcal{A}_n \mid e(a_k)=e(a_{n+1})$
2. a_{n+1} est un attribut \wedge -irréductible pour le contexte $(\mathcal{E}, \mathcal{A}_{n+1}, \mathcal{R}_{n+1})$
3. Pour tout $a_k \in \mathcal{A}_n \mid e(a_k) \subset e(a_{n+1}) \Rightarrow irre(a_k) \cap e(a_{n+1}) \neq \emptyset$

Preuve :

\Rightarrow

1. si $\exists a_k \mid e(a_k) = e(a_{n+1})$ alors $a_k \in d(e(a_{n+1}))$ donc $irre(a_{n+1})=\emptyset$. Ce qui entraîne que $(\mathcal{E}, \mathcal{A}_{n+1}, \mathcal{R}_{n+1})$ n'est pas \wedge -réduit.
2. Si a_{n+1} n'est pas \wedge -irréductible alors $(\mathcal{E}, \mathcal{A}_{n+1}, \mathcal{R}_{n+1})$ n'est pas \wedge -réduit par définition
3. Si $\exists a_k \in \mathcal{A}_{n+1} \mid e(a_k) \subset e(a_{n+1})$ alors $a_{n+1} \in d(e(a_k))$ et si $irre(a_k) \cap e(a_{n+1}) = \emptyset$ cela veut dire que a_k n'est pas un \wedge -irréductible donc $(\mathcal{E}, \mathcal{A}_{n+1}, \mathcal{R}_{n+1})$ n'est pas \wedge -réduit.

\Leftarrow

Pour que $(\mathcal{E}, \mathcal{A}_{n+1}, \mathcal{R}_{n+1})$ ne soit pas \wedge -réduit il faut soit que a_{n+1} ne soit pas \wedge -irréductible dans le contexte $(\mathcal{E}, \mathcal{A}_{n+1}, \mathcal{R}_{n+1})$ ce qui est impossible car les propriétés (1) et (2) assurent que a_{n+1} est \wedge -irréductible.

Soit qu'un autre \wedge -irréductible de \mathcal{A}_{n+1} ne soit plus \wedge -irréductible, pour cela il faudrait que son ensemble

$irre(a_k)$ devienne vide, ce qui est impossible par la propriété 3. En effet $irre(a_k)$ n'était pas vide pour le contexte $(\mathcal{E}, \mathcal{A}_n, \mathcal{R}_n)$. C'est à dire pour a_k avec tous les attributs de \mathcal{A}_n . Si $e(a_k) \not\subset e(a_{n+1})$ alors $irre(a_k)$ n'est pas modifiée, si $e(a_k) \subset e(a_{n+1})$ la condition (3) assure que a_k reste un \wedge -irréductible.

De plus si un attribut b est non \wedge -irréductible l'ajout de tout nouveau attribut ne permet pas de le rendre \wedge -irréductible, car son ensemble $irre(b)$ reste vide. L'ajout d'un attribut ne permet que de réduire les ensembles $irre(a_i)$ associés à chaque \wedge -irréductible a_i , pas de l'augmenter car cet ensemble est produit par une intersection

Pour le contexte de la figure 1 les attributs sont tous \wedge -irréductibles. Si on essaye d'ajouter à ce contexte un nouvel attribut a_7 avec $e(a_7) = \{e_0, e_1\}$, cet attribut ne sera pas ajouté car l'ensemble A des attributs a du contexte courant tel que $a \succ a_7$ est $A : \{a_4, a_5, a_6\}$. On a $(e(a_4) \cap e(a_5) \cap e(a_6)) - e(a_7) = \{e_0, e_1\} - \{e_0, e_1\} = \emptyset$. Donc a_7 n'est pas irréductible. On peut voir sur le treillis de la figure 2 que cet élément existe déjà, il est donc complètement caractérisé par les attributs \wedge -irréductibles déjà connus.

De plus chacun des attributs irréductibles a est caractérisé par un ensemble $irre(a)$ non vide. Par exemple, $irre(a_2) = \{e_0\}$ et $irre(a_6) = \{e_2\}$.

L'intérêt des irréductibles pour notre étude, provient des propriétés suivantes : dans un treillis fini, non vide tout élément est construit comme un \vee d'éléments \vee -irréductibles et comme un \wedge d'éléments \wedge -irréductibles. Ce qui signifie que chaque élément d'un treillis fini est complètement caractérisé par les éléments irréductibles du treillis. Cette propriété est formellement décrite par le résultat suivant :

Théorème 1

Si l'on note $J(L)$ (resp. $M(L)$) l'ensemble des éléments \vee -irréductibles (resp. \wedge -irréductibles) d'un treillis complet $\mathcal{L} : (L, \leq_L, \vee, \wedge)$ et $B(L)$ une relation binaire construite entre $J(L)$ et $M(L)$ avec pour $j_a \in J(L)$ et $m_b \in M(L)$, $B(j_a, m_b) = 1 \Leftrightarrow j_a \leq_L m_b$.

Le treillis de Galois construit à partir du contexte $(J(L), M(L), B(L))$ est isomorphe au treillis \mathcal{L} .

Preuve : Ce résultat est connu depuis 1965 [4]. Il s'agit aussi d'un théorème fondamental de l'ACF [11]. Ce théorème est vrai pour tout treillis complet, il est donc aussi vrai pour un treillis de Galois.

Il n'existe pas de plus petite relation binaire ayant cette propriété. Donc un contexte construit sur deux ensembles irréductibles est le plus petit élément d'une classe de contextes équivalents dans le sens où ces contextes produisent la même structure (à un isomorphisme prêt) pour le treillis de Galois associé. Dans le cas d'un treillis de Galois L obtenu à partir d'un contexte $(\mathcal{E}, \mathcal{A}, \mathcal{R})$, le treillis de Galois L_\wedge obtenu à partir du contexte \wedge -réduit sera isomorphe au treillis L [20]. De plus la partie extension de chaque élément de L_\wedge sera égale à la partie extension de l'élément de L as-

socié (via l'isomorphisme). Ce qui signifie que les possibilités de classement des exemples en utilisant le langage \mathcal{A} ou le langage réduit aux attributs \wedge -irréductibles seront les mêmes. Ce résultat est aussi vrai si $J(L)$ (resp. $M(L)$) est un sous-ensemble des éléments de L qui inclut l'ensemble des éléments \vee -irréductibles (resp. \wedge -irréductibles). Il s'agit de cas où le contexte n'est pas totalement réduit.

Pour illustrer notre propos, si l'on prend le treillis de Galois de la figure 2, la relation binaire que l'on construit, entre les éléments \wedge -irréductibles et les éléments \vee -irréductibles, est la relation \mathcal{R} de la figure 1. C'est aussi le cas pour le treillis de la figure 3, cette considération est la base des résultats que nous montrerons dans le prochain paragraphe et qui généralise cette approche au cas structurel.

3.1 Cas structurel

Dans l'article [19] il est montré que l'analyse des concepts formels peut être étendue au cas structurel. Il est donc possible de considérer une équivalence entre les langages de description pour des cas plus complexes qu'une description par attribut. Pour cela il faut tout d'abord préciser la notion de langage de description et de contexte structurel. Nous définissons un langage de description comme un ensemble d'expressions bien formées, ordonnées par une relation d'ordre partiel que nous noterons $\geq_{\mathcal{L}}$ (voir l'annexe jointe pour plus de précisions).

Un contexte structurel est un triplet $(\mathcal{E}, \mathcal{L}, d)$ où \mathcal{E} est un ensemble d'exemples, \mathcal{L} est un langage de description et $d : \mathcal{E} \rightarrow \mathcal{L}$ une application qui associe à chaque exemple sa description. Dans l'article [19] un langage de description possède un opérateur de généralisation (opérateur produit) permettant de structurer l'espace des descriptions sous la forme d'un treillis. Dans ce cas, la mise en relation entre le treillis de parties de l'ensemble des exemples et le treillis des descriptions se fait naturellement et permet d'obtenir un treillis de Galois dans le cas structurel (figure 3). En effet, il est alors possible d'associer à chaque sous partie de \mathcal{E} une unique description dans \mathcal{L} , ce qui permet de construire l'opérateur de fermeture associé.

En utilisant le théorème 1 nous savons qu'il existe une description propositionnelle des exemples qui permet d'avoir un treillis de Galois isomorphe à celui de la figure 3. Cette nouvelle description utilise les éléments \wedge -irréductibles qui sont, dans la figure 3, matérialisés par un trait plus épais. Ceci nous donne comme une nouvelle description des exemples via une relation binaire. Cette relation est celle de la figure 1. On peut vérifier que le treillis de la figure 2 est bien isomorphe au treillis de la figure 3.

Cependant la recherche des éléments \wedge -irréductibles n'est à priori pas réaliste, si on utilise le principe de parcours du treillis indiqué dans [19]. En effet, le parcours se fait en généralisation (du plus spécifique vers le plus général) soit du bas vers le haut pour la figure 3. Comme les éléments irréductibles sont "statistiquement" plus dans le haut du treillis

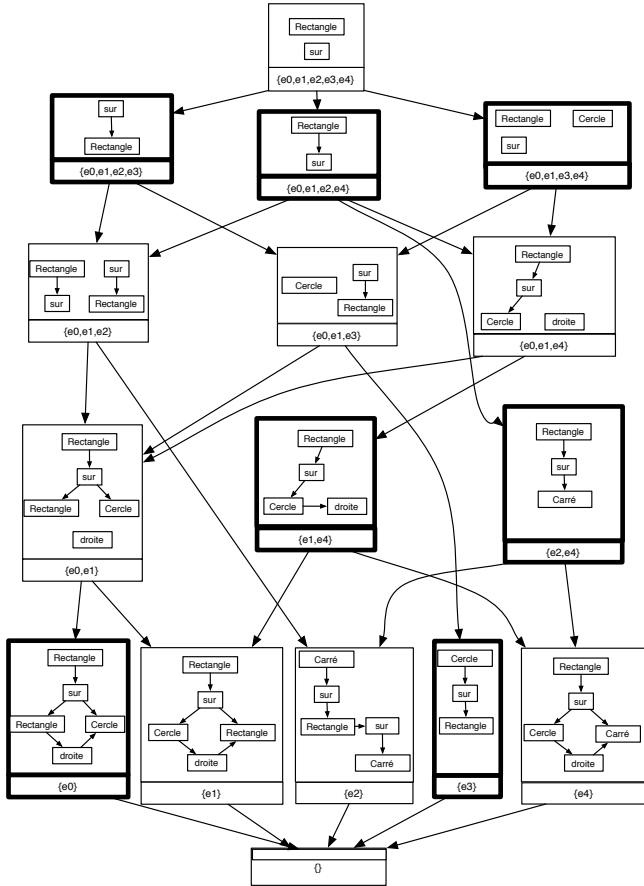


FIG. 3 – Treillis de Galois généralisé : Exemple

(proche du sup) que dans le bas, cette recherche (outre sa complexité) se révèle inadéquate car il faut construire l'ensemble du treillis.

Un principe de parcours en spécialisation semble plus approprié. Toutefois, pour faire ce parcours il faut disposer d'un premier ensemble de descriptions que l'on va "fusionner". Or c'est une partie de ces descriptions que nous cherchons. Dans le cadre de la description propositionnelle, le problème ne se pose pas car le partitionnement par l'intermédiaire des attributs fournit ce premier ensemble de descriptions.

Pour sortir de ce problème on peut utiliser une astuce [12] : nous remplaçons chaque description structurale (dans notre cas des graphes) par l'ensemble de ses sous-descriptions (dans notre cas sous-graphes), en incluant la description elle-même. L'extraction de ces sous-graphes pour un ensemble de graphes est, à-priori, un problème complexe mais des techniques récentes de fouille de graphes permettent une telle recherche en proposant des méthodes efficaces pour la recherche de tous les sous-graphes présents au moins k fois sur n exemples. Pour des raisons d'efficacité, des restrictions au type des graphes recherchés

sont aussi utilisées. Le papier [12] montre que l'espace des concepts est toujours dans ce cas un Treillis de Galois.

4 Fouille de graphes et propositionnalisation

Les techniques de fouille de graphes [8] exploitent les structures particulières de l'espace des descriptions pour accélérer la recherche de structures. Depuis [18] ces techniques ce sont précisées, dernièrement des méthodes optimisées et plus générales ont été développées et ont montré leur efficacité (par exemple [9], [22]).

Pour notre démarche, toutes ces méthodes sont utilisables, notre but étant de réduire l'ensemble des motifs fournis par une de ces méthodes au seul motif nécessaire pour le regroupement et la séparation des exemples. La propriété 3 permet d'envisager la sélection de motifs au fur et à mesure de leur découverte par une méthode du type data mining. L'idée est d'enlever les motifs qui, à l'étape n , ne sont pas informants pour la classification des exemples, par rapport au motif déjà rencontrés. Notre modèle prouve que seul les motifs \wedge -irréductibles sont à conserver. En utilisant ces propriétés nous obtenons l'algorithme suivant :

Procédure : estIrreductible

Données : $C_n : (\mathcal{E}, \mathcal{A}_n, \mathcal{R}_n)$

Données : attribut a

Résultat : vrai si a est \wedge -irréductible sinon faux;

On retourne aussi indirectement $irre(a)$ si a est \wedge -irréductible

$F \leftarrow \{a_k \in \mathcal{A}_n \mid e(a) \subseteq e(a_k)\}$;

si $F = \emptyset$ **alors**

Il est possible que a soit un irréductible situé juste sous le sommet (top) du treillis;

$irre(a) \leftarrow \mathcal{E} - e(a)$

sinon

// Dans le cas ou il y a des attributs irréductibles $> a$;

// On recherche si $irre(a)$ est vide ou non;

$irre(a) \leftarrow e(a_j) - e(a)$; *// ou a_j est un élément quelconque de F*

$F \leftarrow F - a_j$; *// a_j est traité*

pour a_k *in* F **faire**

si $e(a_k) = e(a)$ **alors retourner** faux;

$irre(a) \leftarrow irre(a) \cap e(a_k)$;

si $irre(a) = \emptyset$ **alors retourner** faux;

fin

fin

retourner $irre(a) \neq \emptyset$;

Procédure : mettreAJour

Données : $C_n : (\mathcal{E}, \mathcal{A}_n, \mathcal{R}_n)$

Données : attribut a

Résultat : C_n modifié

// Modification des $irre(a_k)$ pour chaque attribut $a_k < a$;

$F \leftarrow \{ a_k \in \mathcal{A}_n \mid e(a_k) \subseteq e(a) \}$;

pour a_k in F **faire**

$irre(a_k) \leftarrow irre(a_k) \cap e(a)$;

si $irre(a_k) = \emptyset$ **alors** // non \wedge -irréductible ;

$C_n \leftarrow C_n - a_k$;

fin

return C_n ;

Algorithme : APropos : (A propositionalisation Algorithm)

Données : $C_n : (\mathcal{E}, \mathcal{A}_n, \mathcal{R}_n)$

Données : attribut a

// Il s'agit d'un algorithme incrémental (online), à chaque étape on essaye d'ajouter un nouvel attribut;

Résultat : C_{n+1}

si $estIrreductible(C_n, a)$ **alors**

$C_{n+1} \leftarrow mettreAJour(C_n, a)$;

sinon

$C_{n+1} \leftarrow C_n$;

fin

retourner C_{n+1} ;

Algorithme 1 : Propositionalisation formelle incrémentale

Complétude : L'algorithme suit précisément les propriétés données précédemment, la preuve de ces propriétés prouvent l'algorithme.

Complexité en temps et espace : La complexité de l'ajout d'un attribut dans un contexte $(\mathcal{E}, \mathcal{A}_n, \mathcal{R}_n)$ est polynomiale en fonction de la taille de \mathcal{E} et de \mathcal{A}_n . Il faut principalement faire $|\mathcal{A}_n|$ intersection des extensions entre $e(a)$ et $e(a_k)$ avec $a_k \in \mathcal{A}_n$. Comme les listes $e(a)$ peuvent être triées en donnant un ordre total sur les exemples, chaque intersection ou test d'égalité prend au pire en temps égal à $|e(a)| + |e(a_k)|$. Soit au pire $2 \times |E|$. Donc la complexité en proportionnelle à $(|\mathcal{A}_n| \times |E|)$. La complexité en espace est directement fonction de la taille de \mathcal{R}_n .

5 Exemple et Expérimentation

Pour appliquer l'algorithme *APropos* nous utilisons un générateur de motif qui a pour but de fournir des expressions du langage de description, répétées sur les exemples. Dans ce but nous avons utilisé des algorithmes de fouille de graphes [8] qui extraient des sous graphes répétés dans un ensemble de graphes. Les algorithmes choisis sont "l'ancêtre" Inne [18] sur notre petit ensemble d'exemples et un algorithme récent et optimisé nommé Gaston [22] sur des plus grands ensembles d'exemples.

Dans le cas des exemples de la figure 3, nous avons limité la recherche des motifs à des chemins de longueur maximale 2 et vu au moins une fois sur les exemples. Avec cette limitation on trouve la relation binaire de la figure 4.

Dans la figure 4 nous présentons un ensemble de 7 at-

	e_0	e_1	e_2	e_3	e_4
$\{droite \rightarrow Cercle\}$	1	0	0	0	0
$\{Cercle \rightarrow sur\}$	0	0	0	1	0
$\{Cercle \rightarrow droite\}$	0	1	0	0	1
$\{Carre\}$	0	0	1	0	1
$\{sur \rightarrow Rectangle\}$	1	1	1	1	0
$\{Rectangle \rightarrow sur\}$	1	1	1	0	1
$\{Cercle\}$	1	1	0	1	1

FIG. 4 – Matrice \mathcal{R} sur les \wedge -irréductibles chemins.

tributs \wedge -irréductibles. A chacun de ces \wedge -irréductibles peut être associé un ensemble de chemins. Il s'agit d'un sous ensemble des chemins examinés via la méthode de fouille de graphes. Ces chemins étant présents sur exactement le même ensemble d'exemples que l'attribut \wedge -irréductible a . Par exemple, si on a limité la recherche à des chemins de longueurs 2, pour l'attribut \wedge -irréductible associé à l'ensemble d'exemples $\{e_1, e_4\}$ on a $\{Cercle \rightarrow droite, sur \rightarrow Cercle \rightarrow droite\}$. Suivant l'utilisation que l'on veut faire des attributs \wedge -irréductibles on peut soit choisir un élément quelconque de cet ensemble ou choisir le sous ensemble des éléments les plus généraux ou, enfin, le sous-ensemble des éléments les plus spécifiques (pour la relation d'ordre choisie sur le langage de description). Si de plus le langage original possède un opérateur de généralisation (opérateur produit) alors il est possible d'associer à l'attribut \wedge -irréductibles l'unique description associée à l'ensemble des exemples vérifiant l'attribut \wedge -irréductible [19]. Pour notre exemple il s'agit des graphes décrits dans les pavés du treillis de Galois de la figure 3 et pour l'attribut \wedge -irréductible associé à l'extension $\{e_1, e_4\}$ on trouve le graphe : Rectangle \rightarrow sur \rightarrow Cercle \rightarrow droite.

Sur le tableau (figure 4) nous n'avons porté que les éléments les plus généraux et non l'ensemble des motifs ayant même extension. Toutefois, ces informations peuvent être intéressantes pour un post-traitement (recherche de règles d'associations par exemple) et pourront donc être conservées lors de la recherche des éléments \wedge -irréductibles. On peut voir que cette matrice est équivalente (en échangeant les lignes avec les colonnes) avec la matrice de la figure 1. Donc le treillis que nous obtiendrons sera isomorphe au treillis de la figure 2. En fait, pour cet exemple, cette matrice n'évoluera plus, même si l'on spécialisait les motifs chemins, car on a trouvé l'ensemble des \wedge -irréductibles que l'on aurait pu obtenir en construisant l'ensemble du treillis. On voit donc ici que le langage des chemins de longueur 2 offre les mêmes capacités de discrimination que le langage de graphes utilisé dans la figure 2. Toutefois le fait que l'ensemble des \wedge -irréductibles ne change pas entre deux étapes n'assure pas que l'on ait trouvé tous les \wedge -irréductibles. Actuellement, il est nécessaire de construire l'ensemble du treillis structurel si l'on veut s'assurer de ce résultat.

Pour une expérimentation plus conséquente nous avons utilisé le fichier de 340 graphes représentant des molécules fournis avec le logiciel Gaston et issu du NCI (National Cancer Institute). Nous avons fait varier le critère de sélection des motifs entre 15 et 4. Les motifs recherchés étant des chemins ou des arbres dont la taille n'est pas limitée. Nous avons axé notre expérimentation sur le nombre d'éléments irréductibles trouvés. En effet, en tenant compte du théorème 1 nous sommes certains de l'équivalence entre le langage réduit et le langage initial.

Pour vérifier précisément les capacités de réduction de notre algorithme nous avons fait un pré-traitement sur l'ensemble des motifs fournis par Gaston pour éliminer les motifs à extension égale, c'est à dire ceux vus sur exactement sur les mêmes exemples. Nous n'avons gardé qu'un seul motif, comme représentant d'un ensemble de motifs égaux sur les extensions (c.a.d tel que $e(a)=e(b)$). Bien que notre algorithme traite ce cas il nous a semblé nécessaire de bien montrer la réduction qui n'est due qu'à la sélection des \wedge -irréductibles. C'est pourquoi les chiffres du nombre de motifs peuvent paraître petits. Si l'on considère l'ensemble complet des motifs, par exemple dans le cas où l'on recherche les motifs répétés au moins 7 fois sur les 340 exemples, Gaston fournit 120000 motifs, mais seulement 2457 d'entre eux se différencient sur les extensions. Dans le cas où le seuil de sélection est plus bas le nombre de motifs croît de façon exponentielle, par exemple pour un seuil de 4 nous obtenons des millions de motifs. De plus, cette approche permet de comparer nos résultats avec la sélection des motifs fermés [7]. En effet, à chacun des motifs ainsi sélectionné correspond à un fermé sur l'ensemble des motifs observés.

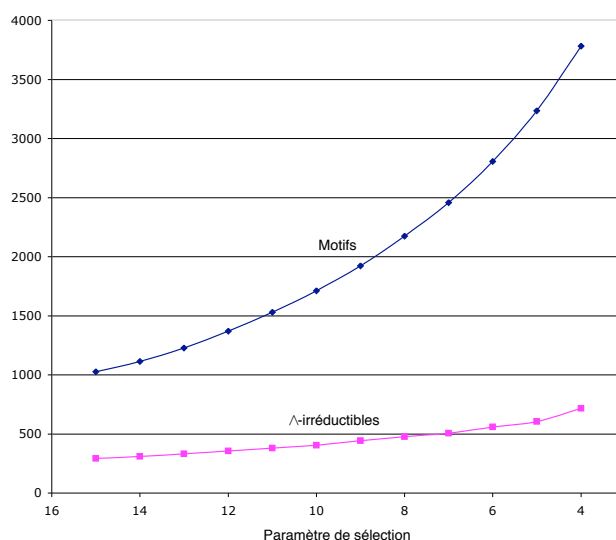


FIG. 5 – Expérimentation molécules

L'axe vertical indique le nombre de motifs (non répétés) extraits par Gaston. L'axe horizontal indique le para-

mètre de sélection fourni à Gaston, ce paramètre est le nombre minimal d'apparition d'un motif pour qu'il soit sélectionné. Plus ce nombre est petit plus le nombre de motifs extraits par Gaston est grand. La courbe supérieure (motifs) indique la croissance du nombre de motifs extraits par Gaston quand on fait varier le paramètre de sélection. La courbe inférieure (\wedge -irréductibles) indique le nombre d'irréductibles trouvés. Nous n'avons pas placé les courbes sur le temps de calcul et l'espace occupé. Ces courbes suivent directement la courbe sur le nombre d'irréductibles, ceci est dû au caractère incrémental de l'algorithme. L'espace mémoire utilisé ne dépasse pas 600k, le temps de calcul varie de 2s à 6s sur un Imac G5 1.8mhz.

Nous avons mené une seconde expérimentation sur les données du problème Mutagenesis utilisée par [8] en ne prenant pas en compte les charges sur les atomes et la classification en exemples/contre-exemples. Dans ce cas, pour limiter la complexité de la recherche de Gaston, nous avons dû utiliser un second paramètre de Gaston en limitant la taille des motifs obtenus à 7 sommets (figure 6).

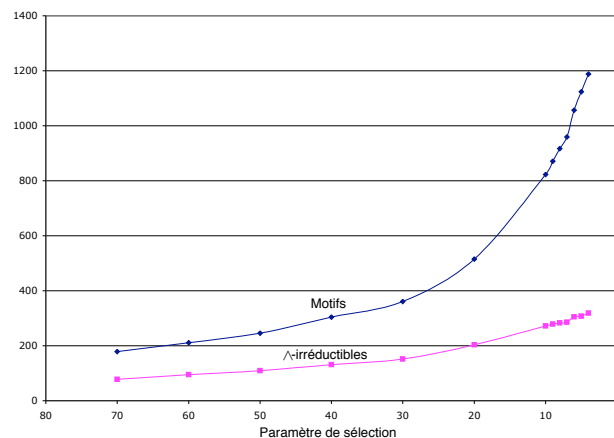


FIG. 6 – Expérimentation Mutagenesis

Pour la première expérimentation le gain obtenu est d'un facteur de 3.5 pour 1027 et de 5.2 pour 3781 motifs. Dans le cas de la seconde expérimentation le gain obtenu est d'un facteur de 2,3 pour 179 mais il est de 3,7 pour 1188 motifs. On voit donc, sur cette expérimentation, qu'il y a un effet d'amortissement de la croissance exponentielle. On peut penser que ce effet provient du positionnement des \wedge -irréductibles dans le treillis. De manière empirique, l'on voit que ceux ci se trouve plutôt dans le haut du treillis [10]. En effet plus il y a d'éléments \wedge -irréductibles plus ceux ci permettent d'éliminer de nouveaux éléments non \wedge -irréductibles.

6 Conclusion

Pourquoi propositionaliser ? Les résultats obtenus dans le cadre de ILP ou Pattern mining semble montrer la non nécessité de ce changement de représentation. Le théorème

montrant qu'il existe une représentation propositionnelle permettant d'obtenir le même espace de recherche que dans le cas structurel est un premier point en faveur de cette transformation. Mais l'existence de ce langage n'est pas tout. En effet, on sait depuis longtemps qu'il existe des passages entre logique du premier ordre et logique propositionnelle mais cette transformation se paie en une taille exponentielle du langage propositionnel. Pour répondre à cette dernière question, nous donnons, dans cet article, un nouvel algorithme permettant, étape après étape, d'approcher ce langage propositionnel. Cet algorithme est basé sur des résultats théoriques anciens et nouveaux. L'expérimentation actuelle montre que la réduction attendue est bien là mais surtout que la croissance exponentielle des motifs n'entraîne pas, au moins sur les cas traités, une croissance exponentielle du langage propositionnel réduit. Ce dernier résultat, très encourageant, est à relier à des précédents résultats de complexité dans le domaine de l'analyse de concepts formels montrant une croissance polynomiale de la taille de l'espace de recherche [21]. En fait, l'utilisation de la relation formelle entre les deux espaces (treillis des parties et treillis des descriptions) permet une réduction importante de l'ensemble des motifs informants que nous retrouvons dans nos résultats. La seconde partie de la réduction obtenue provenant des propriétés des éléments \wedge -irréductibles permettant la réduction de la relation binaire à une forme minimale. Dans le cadre de notre expérimentation, la croissance exponentielle du nombre de motifs est contrôlée par la réduction proposée. La question est de savoir si l'on peut toujours espérer un tel amortissement de la croissance exponentielle ? A cela nous n'avons actuellement pas de réponse.

Remerciements : Je remercie les deux lecteurs anonymes de ce papier pour leurs commentaires constructifs.

Références

- [1] Agrawal.R, Imelinsky.T, Swami.A, Mining association rules between sets of items in large database in proc. ACM SIGMOD'93 conf. (1993) 207-216
- [2] Annalisa Appice and Michelangelo Ceci and Simon Rawles and Peter Flach, Redundant feature elimination for multi-class problems, *Proceedings of the 21st International Conference on Machine Learning (ICML'2004)*, ISBN :1-58113-838-5, ACM, 33-40, 2004.
- [3] Alphonse, E. , Rouveirol C. 2000 Lazy propositionalization for Relational Learning In Horn W., editor, 14th European Conference on Artificial Intelligence, (ECAI'00) Berlin, Allemagne , pages 256-260, IOS Press, 2000.
- [4] Barbut.M, Monjardet.B., "Ordre et classification," *Hachette*, Paris, 1970.
- [5] A.L.Blum, P.Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*,97 :245-271,1997
- [6] L.Chaudron,N.Maille, "Generalized Formal Concept Analysis," *ICCS'2000 Lecture notes in Artificial intelligence*, 1867, pp. 357-370, 2000.
- [7] Chi, Y., Yang, Y., Xia, Y., Muntz, R. R. : CMTreeMiner : Mining Both Closed and Maximal Frequent Subtrees, *The Eighth Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04)*, May 2004
- [8] D.J.Cook, L.B.Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2) :32-41,2000
- [9] A.Inokuchi,T.Washio,H.Motoda. Complete mining of Frequent patterns from Graphs : Mining Graph Data. *Journal of Machine Learning*,Kluwer Academic publishers, 50,321-354,2003.
- [10] Duquenne .V 1999 "Latticial structure in data analysis", *TCS*,217 :407-436,1999
- [11] B.Ganter, R.Wille, « Formal concept analysis, mathematical foundation », 1998, Springer Verlag Ed.] R.Godin, R.Missaoui 1994, An incremental concept formation approach for learning from databases *Theoretical computer science* 133 (1994) 387-419
- [12] Ganter.B., Kuznetsov.S., "Pattern Structures and Their Projections," *ICCS'2002 Lecture notes in Artificial intelligence*, 2002.
- [13] D.Koller, M.Sahami. Toward optimal feature selection. In *proceedings of the thirteenth international conference on machine learning*, pages 284-292, 1996.
- [14] Kramer.S 1999 Relational Learning vs. Propositionalization : Investigations in Inductive Logic Programming and Propositional Machine Learning. PhD thesis, Vienna University of Technology, Vienna, Austria, 1999.
- [15] Lloyd. J.W., 2000 " A Logical Setting for the Unification of Attribute-Value and Relational Learning", *ICML2000 Workshop on "Attribute-Value and Relational Learning"*
- [16] Lavrac, N., Gamberger,D. , Javonoski.V. A Study of relevance learning in deductive databases. *J.Logic.programming*, 16,215-249.1999.
- [17] N.Lavrac, P.Flach, An extended transformation approach to Inductive Logic programming. Mars 2000. Rapport de l'universite de Bristol. Departement of computer sciences.CSTR-00-002
- [18] Liquière.M , J Sallantin, 1989 "INNE : A structural learning algorithm for noisy Data", *European Working Session on Learning*, Decembre 1989 pp 111-123
- [19] Liquiere.M, Sallantin.J 1998 "Structural machine learning with Galois lattice and Graphs.", *Machine Learning : Proceedings of the 1998 International Conferences*,Morgan Kaufmann ed (ICML 98) PP 305-313

- [20] M.Liquière 2002 “Recherche du noyau ontologique” Conférence d’apprentissage, Orléans, Juin 2002, C.Vrain editor, Presses universitaire de Grenoble, pp 13-24.
- [21] Mephu Nguifo E. 1994 “Galois lattice, A framework for concept learning- design, evaluation and refinement”,pp 461-467, Tool with AI, 1994
- [22] Siegfried Nijssen and Joost Kok. A Quickstart in Frequent Structure Mining Can Make a Difference. Proceedings of the SIGKDD, 2004. <http://www.liacs.nl/home/snijssen/gaston/>
- [23] L.Nourine, O.Raynaud 1999, A fast algorithm for building lattices, Information processing letters, 71 (1999) 199-204 Norris E.M., 1974 An algorithm for computing the maximal rectangles of a binary relation. Journal of ACM, 21 :356-366, 1974
- [24] G.D.Plotkin, “A further note on inductive generalization,” in : B.Meltzer and D.Michie (eds), *Machine intelligence*, vol 6, Edinburgh University press, Edinburgh, 101-124, 1971.
- [25] M.Ricordeau, Q-Concept-Learning : Généralisation à l’aide de treillis de Galois dans l’apprentissage par renforcement RFIA, Toulouse, janvier 2004
- [26] Wille.R, 1989.”Knowledge acquisition by methods of formal concept analysis”, in E.Diday, editor, *Data analysis, Learning Symbolic and numeric knowledge*, pages 365-380.1989.
- [27] Lei Yu and Huan Liu, Efficiently handling feature redundancy in high-dimensional data, *KDD '03 : Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003,685–690,Washington, D.C.,ACM Press.

Annexe :

Dans le cadre de la classification d’un ensemble d’exemples \mathcal{E} , un treillis de Galois peut être vu comme la mise en relation de deux treillis \mathcal{X} et \mathcal{D} .

Le treillis \mathcal{X} est le treillis des parties de \mathcal{E} et est défini sur $\prod(\mathcal{E})$ l’ensemble des parties de \mathcal{E} . Dans notre formalisme nous nommons ce treillis : *Treillis des extensions*.

Le treillis \mathcal{D} nommé *Treillis des descriptions* provient d’un langage de description \mathcal{L} qui est un ensemble d’expressions bien formées avec :

1) $\geq_{\mathcal{L}}$: un ordre partiel entre les expressions de \mathcal{L} . Par exemple pour un langage \mathcal{L} où les expressions sont des graphes, l’ordre partiel utilisé peut être l’inclusion de graphes (isomorphisme de sous-graphe), et l’égalité associé est l’isomorphisme de graphes.

Toutefois d’autres classes de graphes existent (chemins, arbres, graphes irredondants) et on peut considérer d’autres relations entre ces graphes comme par exemple l’homomorphisme [19].

2) Un opérateur produit $\otimes_{\mathcal{L}} : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$ avec pour $D_1 \in \mathcal{L}$ et $D_2 \in \mathcal{L}$, $D = D_1 \otimes_{\mathcal{L}} D_2 \in \mathcal{L} \mid D \geq_{\mathcal{L}} D_1, D \geq_{\mathcal{L}} D_2$ and $\forall D' \mid (D' \geq_{\mathcal{L}} D_1) \text{ et } (D' \geq_{\mathcal{L}} D_2) \Rightarrow D' \geq_{\mathcal{L}} D$.

Il s’agit d’un classique opérateur de généralisation [24]. Dans le cas d’un langage de description utilisant comme relation d’ordre partiel l’inclusion, il n’y a pas d’opérateurs produit car à partir de deux graphes on n’obtient pas un unique graphe généralisant ces deux graphes. Toutefois, si l’on change de langage en considérant que la description d’un concept est un ensemble de graphes alors il existe bien un opérateur produit. En fait, on associe à chaque graphe l’ensemble des sous-graphes inclus ou égal à ce graphe. L’intersection de deux graphes est alors l’intersection des deux ensembles associés.

La relation entre ces deux treillis est donné par les deux applications $d : \mathcal{X} \rightarrow \mathcal{D}$ et $e : \mathcal{D} \rightarrow \mathcal{X}$ (voir figure 7).

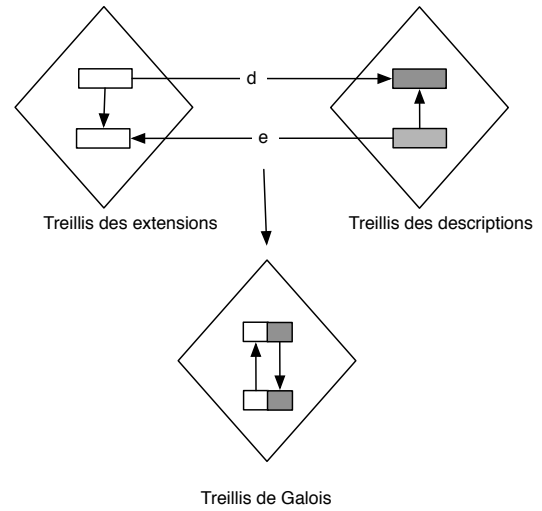


FIG. 7 – Relations entre les deux treillis

Nous généralisons alors la notion de contexte en analyse de concepts formels. Un *contexte structural* [19],[6] est un triplet $(\mathcal{E}, \mathcal{L}, d)$ où \mathcal{E} est un ensemble (d’exemples), \mathcal{L} un langage de description et $d : \prod(\mathcal{E}) \rightarrow \mathcal{L}$ une application. On peut transformer chaque contexte $(\mathcal{E}, \mathcal{A}, \mathcal{R})$ en un contexte structural $(\mathcal{E}, \mathcal{L}, d)$, où \mathcal{L} est l’ensemble des parties de \mathcal{A} .

Pour un contexte structural $(\mathcal{E}, \mathcal{L}, d)$ un concept est une paire (E, D) avec :

- $D \in \mathcal{L} \mid D = \otimes_{\mathcal{L}} d(\{e\})$ pour $e \in E$.
- $E \in \prod(\mathcal{E}) \mid E = \{e_i \in \mathcal{E} \mid D \geq_{\mathcal{L}} d(e_i)\}$.

Pour deux concepts $(E_1, D_1), (E_2, D_2)$ nous avons la relation d’ordre partiel suivante : $(E_1, D_1) \geq_c (E_2, D_2) \Leftrightarrow E_1 \supseteq E_2 (\Leftrightarrow D_1 \geq_{\mathcal{L}} D_2)$.

Pour un contexte structural $(\mathcal{E}, \mathcal{L}, d)$ l’ensemble de tous les concepts de ce contexte ordonnés par la relation \geq_c , est un treillis de Galois [19].

Pour ce treillis les opérations \wedge (inf) et \vee (sup) sont les suivantes :

$(E_1, D_1) \vee (E_2, D_2) = (E, D)$ avec $D = (D_1 \otimes_{\mathcal{L}} D_2)$ et $E = \{e \in \mathcal{E} \mid d(\{e\}) \leq_{\mathcal{L}} D\}$.

$(E_1, D_1) \wedge (E_2, D_2) = (E, D)$ avec $E = E_1 \cap E_2$ et $D = \otimes_{\mathcal{L}} d(e_i), \forall e_i \in E$.