Arc Consistency Projection: A New Generalization Relation for Graphs

Michel Liquière

Arc Consistency Projection: A New Generalization Relation for Graphs

Michel Liquiere

LIRMM,
161 Rue ada,
34392 Montpellier cedex 5,
France
Liquiere@lirmm.fr

Abstract. The projection problem (conceptual graph projection, homomorphism, injective morphism, θ-subsumption, OI-subsumption) is crucial to the efficiency of relational learning systems. How to manage this complexity has motivated numerous studies on learning biases, restricting the size and/or the number of hypotheses explored. The approach suggested in this paper advocates a projection operator based on the classical arc consistency algorithm used in constraint satisfaction problems. This projection method has the required properties : polynomiality, local validation, parallelization, structural interpretation. Using the arc consistency projection, we found a generalization operator between labeled graphs. Such an operator gives the structure of the classification space which is a concept lattice.

1 Introduction

The complexity of the computation of the generality relation between two relational descriptions, is a crucial problem. For conceptual graphs [1] this operation is named projection. Such an operation is linked to a classical problem in the graph community: the search for an homomorphism between two graphs. As stated in [2], "the elementary reasoning operation, projection is a kind of graph homomorphism that preserves the partial order defined on labels". The search for an homomorphism between a tree and a graph is polynomial but between general graphs, the problem is NP complete [3]. In conceptual graph community, different algorithms are proposed for the projection problem [4,5,6].

From another point of view, Inductive Logic Programming systems (ILP) commonly used a generality relation, a decidable restriction of logical implication named θ-subsumption. The homomorphism is also directly linked to the θ-subsumption operation [7]. In machine learning, the complexity of this operation has motivated the use of learning biases: syntactic biases (trees [8], specific graph [9,10]), efficient implementation [7] and approximation of θ-subsumption [11].

Finally, the homomorphism is also linked to the classical constraint programming resolution (CSP) [12]. This final link gives an interesting algorithmic point

U. Priss, S. Polovina, and R. Hill (Eds.): ICCS 2007, LNAI 4604, pp. 333–346, 2007.

of view since CSP community has many results which improve the resolution algorithm.

In this paper, we propose to use a part of these three domains for a classical unsupervised machine learning problem. We represent each example by a labeled graph. We use a new generality relation named AC-projection based on the arc consistency algorithm [12] and we prove that the search space, for a relational machine learning classification problem, is a concept lattice [13].

2 A New Projection: AC-Projection

Any constraint satisfaction problem can be viewed as a "network" of variables and constraints. In this network each variable is connected to the constraints that involve it and each constraint is connected to the variables it involves. Among backtracking based algorithms for constraint satisfaction problems, algorithm employing constraint propagation, like forward checking and arc consistency [12], have had the most practical impact. These algorithms use constraint propagation (arc consistency) during a search to prune inconsistent values from the domains of the uninstantiated variables.

2.1 AC-Projection and Arc Consistency

In this paragraph we present the arc consistency using a graph notation. For other presentations see the books [3,12].

Notation. For a labelled directed graph, named *digraph* in this paper, G, we note $V(G)$ the set of vertices of G, $A(G)$ the set of arcs of G, $L(G)$ the set of labels of G. For a vertex $x \in V(G)$ we note $l(x) \in L(G)$ the label of x, $N(x)$ the set of all the neighbors of x, $P(x) \subseteq N(x)$ the predecessors of x and $S(x) \subseteq N(x)$ the successors of x.

For a finite set S we note 2^S the set of all subsets of S (power set). In this paragraph we study some important properties of the arc consistency.

Definition 1 (labeling). *Let G_1 and G_2 be two digraphs. We named labeling from G_1 into G_2 a mapping $\mathcal{I}$:V(G_1) $\rightarrow$ $2^{V(G_2)}$ | $\forall$ $x \in$ V(G_1), $\forall$ $y \in \mathcal{I}(x)$, $l(x)=l(y)$.*

Thus for a vertex $x \in \mathrm{V}(G_1)$, $\mathcal{I}$(x) is a set of vertices of G_2 with the same label $l(x)$. We can think of $\mathcal{I}(x)$ as the set of "possible images" of the vertex x in G_2. This first labeling is trivial but can be refined using the neighborhood relations between vertices.

Definition 2 ($\sim\succ$). *Let G be a digraph, $V_1 \subseteq$ V(G), $V_2 \subseteq$ V(G).*
We note $V_1 \sim\succ V_2$ iff
1) $\forall x_k \in V_1$ $\exists y_p \in V_2$ | $(x_k,y_p) \in A(G)$
2) $\forall y_q \in V_2$ $\exists x_m \in V_1$ | $(x_m,y_q) \in A(G)$.

In this definition we give a direct relation between two sets of vertices V_1 and V_2. So for each vertex x_k of V_1 there is at least one vertex y_p of V_2 which is a

neighbor of x_k:$((x_k, y_p) \in \mathrm{A}(G))$ and all vertices of V_2 are a neighbor of, at least, one vertex of V_1 (oriented condition). This is not a one to one relation like the subgraph isomorphism.

Definition 3 (Consistency for one arc). *Let G_1 and G_2 be two digraphs. We say that a labeling $\mathcal{I}$:$V(G_1) \rightarrow 2^{V(G_2)}$ is* consistent with *an* arc $(x, y) \in A(G_1)$, *iff $\mathcal{I}(x) \sim\!\!\succ \mathcal{I}(y)$.*

In the example of Figure 1, a vertex is designated by a letter and a number: the letter is the label of the vertex and the number is only an identification number. In this example the labeling $\mathcal{I}$:$\mathcal{I}(a_0)=\{a_4, a_{10}\}$ and $\mathcal{I}(b_1)=\{b_5, b_9\}$, is consistent with the edge (a_0,b_1) since $\mathcal{I}(a_0) \sim\!\!\succ \mathcal{I}(b_1)$.

Definition 4 (AC-projection $\rightharpoonup$). *Let G_1 and G_2 be two digraphs. A labeling $\mathcal{I}$ from G_1 into G_2 is an AC-projection iff $\mathcal{I}$ is consistent with all the arcs $e \in A(G_1)$. We note it $G_1 \rightharpoonup G_2$*

The name "AC-projection" comes from the classical AC (arc consistency) used in [12].

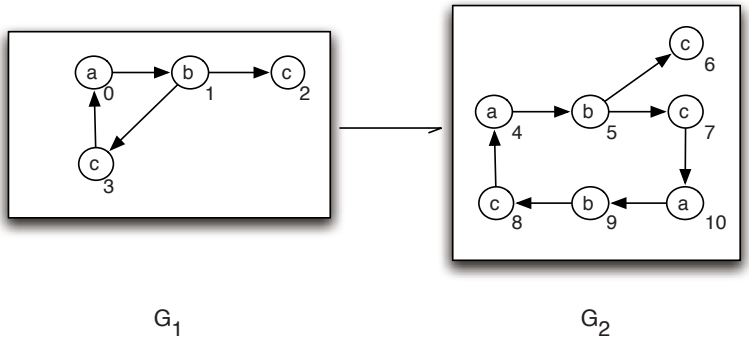

Fig. 1. AC-Projection: Example

Consider the labelling $\mathcal{I}(a_0)$:$\{a_4,a_{10}\}$, $\mathcal{I}(b_1)$:$\{b_5,b_9\}$, $\mathcal{I}(c_2)$:$\{c_6, c_7,c_8\}$,$\mathcal{I}(c_3)$:$\{c_7,c_8\}$. We verify $\mathcal{I}(a_0) \sim\!\!\succ \mathcal{I}(b_1)$, $\mathcal{I}(b_1) \sim\!\!\succ \mathcal{I}(c_2)$, $\mathcal{I}(b_1) \sim\!\!\succ \mathcal{I}(c_3)$, $\mathcal{I}(c_3) \sim\!\!\succ \mathcal{I}(a_0)$. Then $\mathcal{I}$ is an AC-projection from G_1 into G_2 since $\mathcal{I}$ is a labelling consistent with all arcs of G_1.

2.2 AC-Projection Properties

We have defined a new mapping relation between graphs. In this paragraph we study the properties of this relation (complexity, interpretation).

We recall the classical homomorphism definition for digraphs.

Definition 5 (homomorphism $\mapsto$). *A* homomorphism *of a digraph G_1 to a digraph G_2 is a mapping of the vertex sets f:V(G_1) $\rightarrow$ V(G_2) which preserves the arcs and labels, i.e such that (x,y) $\in$ A(G_1) $\Rightarrow$ (f(x),f(y)) $\in$ A(G_2) and $\forall x$, l(x)=l(f(x)).*
Notation: $G_1 \mapsto G_2$

Note that for digraph (f(x),f(y))$\in$A(G_2) implies that f(x) $\neq$ f(y), since each edge of A(G_2) consists of two distinct elements.

We have the following proposition which links the AC-projection to the Homomorphism.

Proposition 1. *For two digraphs G_1 and G_2, if $G_1 \mapsto G_2$ then $G_1 \rightharpoonup G_2$.*

Proof. See [3]

This proposition is the foundation of many CSP resolution methods. These methods are based on the classical arc consistency algorithm AC1 used in CSP, which has been improved (AC2 ... AC5), the actual minimal complexity is: O(ed^2) where e is the number of arcs and d the size of the largest domain [12].

In our case, the size of the largest domain is the size of the largest subset of nodes with the same label. So an AC-projection between two digraphs can be computed in polynomial time.

2.3 AC-Projection Algorithm

We give a simple AC-projection algorithm for digraphs (based on AC1 algorithm [12]).

This algorithm *Arc-Consistency* takes two digraphs G_1, G_2 and tests if there is an AC-projection from G_1 into G_2. It begins by the creation of a first rough labeling $\mathcal{I}$ and reduces, for each vertex x, the given lists $\mathcal{I}$(x) to consistent lists using the procedure *ReviseArc*.

The consistency check fails if some $\mathcal{I}$(x) becomes empty; otherwise the consistency check succeeds and the algorithm gives the labeling $\mathcal{I}$ which is an AC-projection $G_1 \rightharpoonup G_2$.

Procedure: ReviseArc
Data: An arc (x,y)$\in$ V(G_1)
Data: A labeling $\mathcal{I}$ from G_1 into G_2
Data: A digraph G_2
Result: A new labeling $\mathcal{I}'$ from G_1 into G_2
$\mathcal{I}' := \mathcal{I}$;
$\mathcal{I}'$(x):= $\mathcal{I}$(x) - {x' $\in$ V(G_2) | $\nexists$ y' $\in$ $\mathcal{I}$(y) with (x',y') $\in$ A(G_2)};
$\mathcal{I}'$(y):=$\mathcal{I}$(y) - {y' $\in$ V(G_2) | $\nexists$ x' $\in$ $\mathcal{I}$(x) with (x',y') $\in$ A(G_2)};
return $\mathcal{I}'$

Procedure: Arc-Consistency

Data: Two digraphs G_1 and G_2

Result: An AC-projection $\mathcal{I}$ from G_1 into G_2 if there is one else an empty set $\emptyset$

// Initialisation

for $x \in$ *V(G_1)* **do**

  $\mathcal{I}(x) = \{y \in V(G_2) \mid l(x) = l(y))\}$;

end

$S := A(G_1)$;

while $S \neq \emptyset$ **do**

  Choose an arc (x,y) from S; // In general the first element of S

  $\mathcal{I}'$:=ReviseArc((x,y),$\mathcal{I}$,G_2);

  //If for one vertex x $\in$ V(G_1) we have $\mathcal{I}'$(x)= $\emptyset$ then there is no arc consistency

  if $(\mathcal{I}'(x) = \emptyset)$ *or* $(\mathcal{I}'(y) = \emptyset)$ **then**

    return $\emptyset$;

  end

  // $\mathcal{I}'$ is consistent now with the arc (x, y); but it can be non-consistent with some other previously tested arcs so we have to verify and change (if necessary), the consistency of all these arcs.

  if $\mathcal{I}(x) \neq \mathcal{I}'(x)$ **then**

    $S := S \bigcup \{(x', y') \in V(G_1) \mid x' = x \text{ or } y' = x\}$;

  end

  if $\mathcal{I}(y) \neq \mathcal{I}'(y)$ **then**

    $S := S \bigcup \{(x', y') \in V(G_1) \mid x' = y \text{ or } y' = y\}$;

  end

  Remove (x,y) from S;

  $\mathcal{I}$:=$\mathcal{I}'$;

end

return $\mathcal{I}$;

The *Arc-Consistency* algorithm has a polynomial time complexity [3,12] and gives, if there is one, an AC-projection $\mathcal{I}$ from G_1 into G_2 verifying: for all AC-projection $\mathcal{I}'$ from G_1 into G_2, we have $\forall$ x $\in$ V(G_1), $\mathcal{I}'$(x) $\subseteq$ $\mathcal{I}$(x) [3].

3 AC-Projection and Machine Learning

In [10], we have studied the construction of a concept lattice, where the extension part is a subset of the set of example but where the intension part is described by a digraph. In the context of machine learning, the automatic bottom up construction of such a hierarchy can be viewed as an unsupervised conceptual classification method. In this paper the generalization partial order was based on homomorphism relation between digraph. To deal with the homomorphism complexity, we proposed a class of digraph with a polynomial homomorphism operation. This limit the generality of the description language.

In that paper we propose to put the bias on the projection operator. Since the complexity of the AC-projection is polynomial, our idea is to use the AC-projection algorithm instead of the homomorphism projection.In doing so, we need a structural interpretation of the results. In the case of the subgraph isomorphism relation between two graphs, there is no interpretation problem, because it is an "inclusion" relationship. For the homomorphism relation the interpretation is less natural since two vertices can get the same image. The structural interpretation of the AC-projection seems unnatural. For example, see Figure 1 and seek for the substructures which are in G_1 and G_2.

In fact, in the paper [3], the author gives the following proposition:

Proposition 2. *Let G_1 and G_2 be any labelled digraphs with $G_1 \rightharpoonup G_2$. If an directed labeled tree T satisfies $T \mapsto G_1$ then $T \mapsto G_2$.(recall $\mapsto$ is the homomorphism relation)*

A limited interpretation of the proposition 2 is: every subtree of G_1 has an homomorphic image in G_2. So all the covering trees of the digraph G_1 of the Figure 1, are homomorphic with G_2.

3.1 AC-Projection and Generalization

The basic building blocks of concept learning is the notion of example and description language. In our framework, each example, of the set of example, is described by one digraph. So we have a set of digraphs $\mathcal{E}$. We want to find a set of labeled digraphs which have an AC-projection with a subset of the labelled

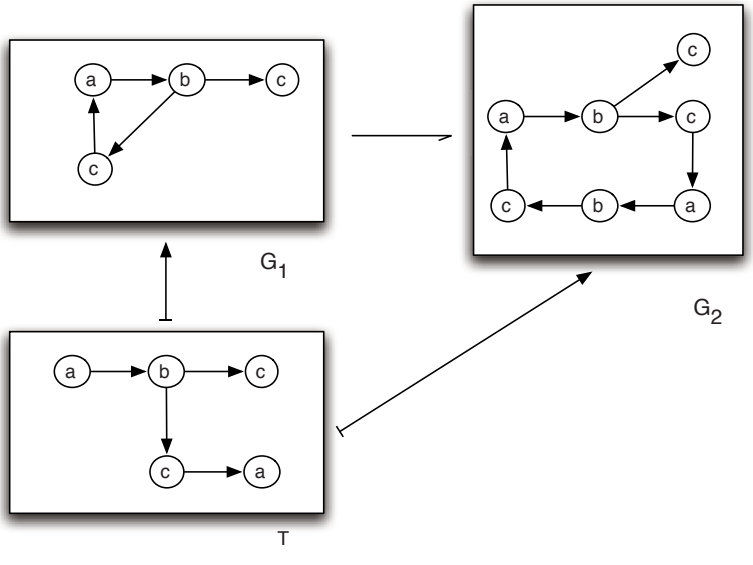

Fig. 2. AC-projection and interpretation

digraphs in $\mathcal{E}$. It is a classical unsupervised classification problem. A generalization algorithm uses a generalization operator: from two graphs we search for the more specific graph which generalizes two graphs (least general generalization [14]). Our generalization order will use the AC-projection relation.

First, we have to specify the generalisation relation between digraphs.

Definition 6 (Generalisation relation)
For two digraphs G_1,G_2, we consider that G_1 is more general than G_2 iff $G_1 \rightharpoondown G_2$.

This relation is only a pre-order because the antisymmetry property is not fulfilled.

The same problem occurs in Inductive Logic Programming. To get rid of this problem, Plotkin [14] defined equivalence classes of clauses, and showed that there is a unique representative of each clause, which he named 'the reduced clause'. For this purpose, we define the following equivalence relation between two digraphs.

Definition 7 (AC-equivalence graphs)
Two digraphs G_1 and G_2 are AC-equivalent, denoted by $G_1 \leftrightharpoons G_2$, iff both $G_1 \rightharpoondown G_2$ and $G_2 \rightharpoondown G_1$.

For example in the Figure 1 we have $G_1 \rightharpoondown G_2$ but we also have $G_2 \rightharpoondown G_1$ with the labelling $\mathcal{I}$: $\mathcal{I}(a_4) = \{a_0\}, \mathcal{I}(a_{10}) = \{a_0\}, \mathcal{I}(b_5) = \{b_1\}, \mathcal{I}(b_9) = \{b_1\}, \mathcal{I}(c_7) = \{c_3\}, \mathcal{I}(c_8) = \{c_3\}, \mathcal{I}(c_6) = \{c_2, c_3\}$.

Using this equivalence relation, we can define equivalence classes of digraph.

3.2 AC-Projection and Reduction

We have an equivalence relation between graphs using the AC-projection. In this paragraph we study the properties of this operation and search for a reduced element in an equivalence class of graphs. For this purpose, we define two reduction operators. Using these operators we construct an AC-equivalent digraph by removing (first operator) or merging (second operator) vertices.

Definition 8 (AC-redundant vertex)
For a digraph G, for a vertex $x \in V(G)$, if $G \leftrightharpoons G\text{-}x$ then x is an AC-redundant vertex. (With G-x = G' s.t V(G')=V(G) - {x} and A(G')=A(G)-{(y,z) | y=x or z=x}).

In the Figure 3 the node 1 labelled "c" is AC-redundant.

Definition 9 (AC-equivalent vertices)
For a digraph G, we say that x_1,$x_2 \in V(G)$ are AC-equivalent iff for the AC-projection $\mathcal{I}$: $G \rightharpoondown G$, $\mathcal{I}(x_1)=\mathcal{I}(x_2)$.

In the Figure 3 the nodes with same label are, in this case, AC-equivalent.

These two definitions give a reduction operator.

Procedure:$\mathcal{R}$

Data: a labelled digraph G
Result: a labelled digraph G' //with G' $\leftrightharpoons$ G
G':=G;
next := true;
while *next* **do**
  next:=false;
  if *(there is a AC-redundant vertex x ∈ V(G'))* **then**
    G':=G'-x;
    next:=true;
  end
  if *there is a set of AC-equivalent vertices E=*$\{x_1,\ldots x_n\}$*∈V(G') with* $|E|>1$ **then**
    // Merge of the AC-equivalent vertices of E
    add a vertice x to V(G') with P(x)= $\bigcup$ P(x_i) and S(x)=$\bigcup$ S(x_i);
    // remove all x_i ∈ E from G'
    G':=G'-E ;
    next:=true;
  end
end
return *G'*;

This $\mathcal{R}$ operation is polynomial because the AC-projection is polynomial.

Proposition 3 ($\mathcal{R}$ equivalence)
$\mathcal{R}(G) \leftrightharpoons G$

Proof.
1) If we remove an AC-redundant vertex x ∈ V(G), by definition G $\leftrightharpoons$ G-x.
2) if we merge a set of AC-equivalent vertices E=$\{x_1,\ldots x_n\}$ with $|E|>1$ in a vertex x, we obtain a new graph G'. We have to prove that G $\leftrightharpoons$ G'
We have a AC-projection $\mathcal{I}$ from G into G
a) G $\rightharpoonup$ G'
We construct a labeling $\mathcal{I}'$ from G into G' with
for x_i ∈ E, $\mathcal{I}'(x_i)$=x
for $y_j \notin$ E, if $\mathcal{I}(y_j) \bigcap$ E $\neq \emptyset$ then
$\mathcal{I}'(y_j)$= ($\mathcal{I}(y_j)$ - E) $\bigcup$ x
else
$\mathcal{I}'(y_j)=\mathcal{I}(y_j)$
We know that $\forall$ x_i ∈ E, and $\forall$ y_j∈S(x),$\mathcal{I}(x_i) \sim\!\!\succ \mathcal{I}(y_j)$. Since S(x)=$\bigcup$ S(x_i) | x_i ∈ E we have $\mathcal{I}'(x_i)$={x} $\sim\!\!\succ \mathcal{I}'(y_j)$ and reciprocally for the predecessors.
So $\mathcal{I}'(y_j)$ is an AC-projection.
b) G' $\rightharpoonup$ G
We construct a labeling $\mathcal{I}''$ from G' into G with
for x ∈ V(G'), $\mathcal{I}''(x)$=E
for y ∈ V(G') and y ≠ x, $\mathcal{I}''(y)=\mathcal{I}(y)$

We know that for all $x_i \in \mathrm{E}$ and $y_j \in \mathrm{S}(x_i)$, $\mathcal{I}(x_i) \sim\!\!\succ \mathcal{I}(y_j)$ (and reciprocally for the predecessors).

Since $\mathcal{I}''(x)=\mathcal{I}(x)$ and $\mathcal{I}''(y)=\mathcal{I}(y)$, we have $\mathcal{I}''(x_i) \sim\!\!\succ \mathcal{I}''(y_j)$ then $\mathcal{I}''$ is an AC-projection.

The Figure:3 shows the application of the $\mathcal{R}$ reduction operator on a digraph.

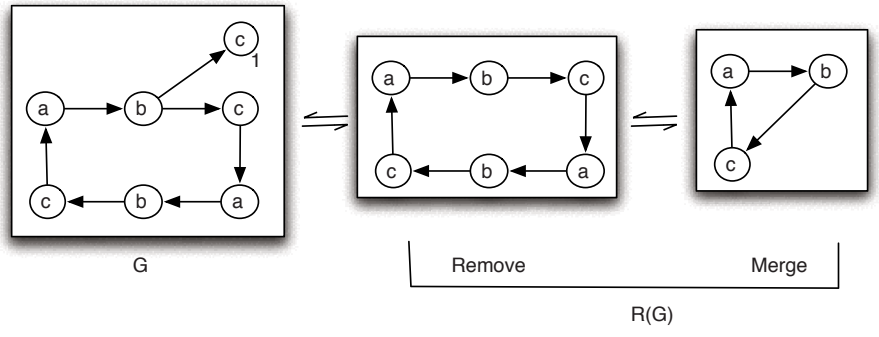

Fig. 3. $\mathcal{R}$(G)

3.3 AC-Projection and Generalization Operator

There are some pairs, (representation languages and generality relations), which have a least general generalization operator . For logic formula and θ-subsumption, this operator is the classical lgg (or rlgg) introduced by plotkin [14]. For graph and homomorphism this operator is the graph product [10]. In mathematics this kind of operator is defined as a product operator [15].

Definition 10 (Product operator). *A binary operator* $\bullet$ *is a product operator iff for a pre order (or a partial order)*$\geqslant$ *between element* E_i

- $E_1 \bullet E_2 \geqslant E_1$
- $E_1 \bullet E_2 \geqslant E_2$
- *if* $E \geqslant E_1$ *and* $E \geqslant E_2$ *then* $E \geqslant E_1 \bullet E_2$

For digraph we have the following product operator ($\otimes$) for the homomorphism pre-order [16,3,10].

Definition 11 (Product operator $\otimes$ for digraphs and homomorphism) *For two digraphs* G_1 *and* G_2 *We construct* $G=G_1 \otimes G_2$ *with*

- $L(G) = L(G_1) \bigcap L(G_2)$
- $V(G) \subseteq V(G_1) \times V(G_2)=\{x \mid x = (x_1, x_2)$ *with* $l(x) = l(x_1) = l(x_2)\}$
- $A(G)=\{(x,x') \mid x=(x_1,x_2),x'=(x'_1,x'_2)$ *and* $(x_1, x'_1) \in V(G_1)$, $(x_2,x'_2) \in V(G_2)$ $\} \subseteq A(G_1) \times A(G_2)$

For the AC-projection we have also a generalization operator.

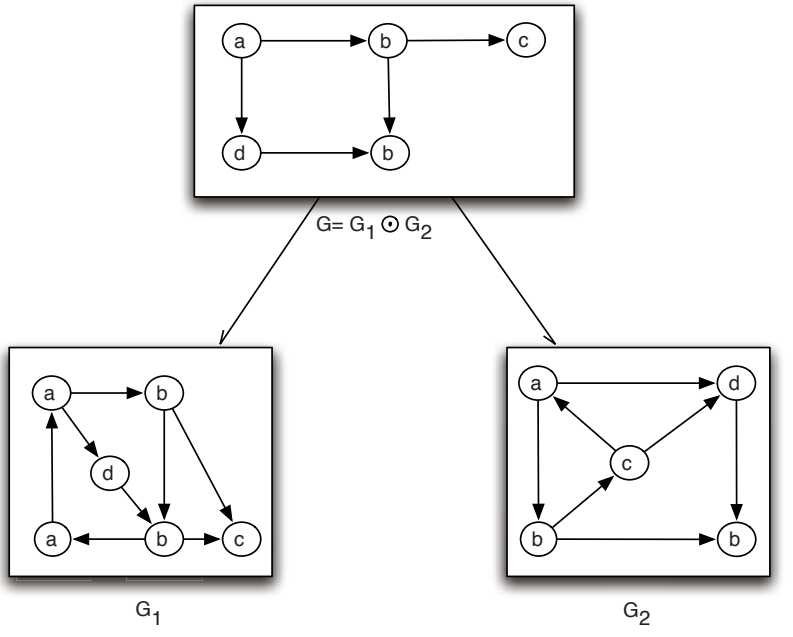

Fig. 4. Product $\odot$ of two digraphs

Proposition 4 (Product operator $\odot$ for digraphs and AC-projection) *For two digraphs G_1, G_2. The binary operator $G_1 \odot G_2 = \mathcal{R}(G_1 \otimes G_2)$ is a product operator.*
So for $G{=}G_1 \odot G_2$ we have:
1) $G \rightharpoonup G_1$, $G \rightharpoonup G_2$
2) for a digraph G' if $H \rightharpoonup G_1$ and $G' \rightharpoonup G_2 \Rightarrow G' \rightharpoonup G$

Proof
1) For two digraphs G_1, G_2, and G=$G_1 \otimes G_2$ we have G $\mapsto G_1$ and G $\mapsto G_2$ (by property of the digraph product operation [10]). And we know:

- $\mathcal{R}(G) \leftrightharpoons G$ (proposition 5).
- if $G \mapsto G_1 \Rightarrow G \rightharpoonup G_1$ (proposition 1)

2) We know that G' $\rightharpoonup G_1$ and G' $\rightharpoonup G_2$. So there is two labelings $\mathcal{I}_1$ and $\mathcal{I}_2$ with for each x $\in$ V(G'), $\mathcal{I}_1$(x)=$\{x_1^1, ..., x_1^n\} \subseteq$ V(G_1) and $\mathcal{I}_2$(x)=$\{x_2^1, ..., x_2^m\} \subseteq$ V(G_2).

In $G_1 \otimes G_2$ we have all the couples $(x_1^1, x_2^1), (x_1^1, x_2^2), ...(x_1^n, x_2^m)$ by construction. We define the following labelling $\mathcal{I}$: G' $\rightarrow G_1 \otimes G_2$ with $\mathcal{I}$(x)= $\mathcal{I}_1$(x)$\times\mathcal{I}_2$(x). If (x,y) $\in$ V(G') we have to prove that $\mathcal{I}(x) \sim\!\!\succ \mathcal{I}(x)$.

For each $(x_1^i, x_2^j) \in \mathcal{I}$(x) there is $(y_1^a, y_2^b) \in \mathcal{I}$(x) with $(x_1^i, y_1^a) \in$ V(G_1) and $(x_2^j, y_2^b) \in$ V(G_2). Because $x_1^i \in \mathcal{I}_1$(x) there is, at least, one $y_1^a \in \mathcal{I}_1$(y) with $(x_1^i, y_1^a) \in$ V(G_1). Since $x_2^j \in \mathcal{I}_2$(x) there is, at least, one $y_2^b \in \mathcal{I}_2$(y) with (x_2^j, y_2^b) $\in$ V(G_2). Since, the graph product $G_1 \otimes G_2$ builds all the couple (x_1^i, x_2^j) and

(y_1^a, y_2^b) (with same labels). $((x_1^i, x_2^j), (y_1^a, y_2^b)) \in$ V(G) iff $(x_1^i, y_1^a) \in$V(G_1) and $(x_2^j, y_2^b) \in$V(G_2). So for $(x_1^i, x_2^j) \in \mathcal{I}$(x) there is, by definition of $\mathcal{I}$, $(y_1^a, y_2^b) \in \mathcal{I}$(y) then $\mathcal{I}$(x)$\sim\succ\mathcal{I}$(y). $\mathcal{I}$ is an AC-projection from G' $\rightharpoonup G_1 \otimes G_2$. Since $\mathcal{R}$(G) $\leftrightharpoons$ G (by construction) then $\odot$ is a product operator.

In Figure:4 G is the product of G_1 and G_2. It represents all the different[1] subtrees common at G_1 and G_2.

4 Concept Lattice and AC-Projection

We have a generalization operator and a pre-order between digraphs. With this knowledge, we can define the notion of concept [13].

Definition 12 (concept, $\vee$, $\geq$)
For a set of examples $\mathcal{E}$, each example $e \in \mathcal{E}$ is described by a digraph d(e) $\in D$ (description space).

For a digraph G, we note α(G)=$\{e_i \in \mathcal{E} \mid G \rightharpoonup d(e_i)\}$.

For $E_1 \subseteq \mathcal{E}$, we note $\beta(E_1)=\odot_{e\in H} d()$.

A concept is a couple (E_1,G_1) with $E_1 \subseteq E$, G_1 a digraph with $\alpha(G_1)= E_1$ and $\beta(E_1) \leftrightharpoons G_1$.

For two concepts (E_1,G_1), (E_2,G_2):

(E_1,G_1) $\vee$ (E_2,G_2)=(α(G),G=$G_1 \odot G_2$)(E_1,G_1) $\geq$ (E_2,G_2) iff $G_1 \rightharpoonup G_2$

Proposition 5 (AC concept lattice)
For a set of examples $\mathcal{E}$, each example $e \in \mathcal{E}$ is described by a digraph d(e) $\in D$. The correspondance α,β defines a Galois connection between $2^{\mathcal{E}}$ and D.

Proof. see [10]

This proposition gives the structure of the search space (a concept semilattice). The size of the concept lattice is limited by the minimum of SD and SP where SD is the size of the description space and SP the size of the partition space. The size SD is very large for relational description but SP is limited by 2^n where n is the number of examples.

If we use our method on the set of examples of [10] we obtain the following join-semilattice Figure:5. In this concept semilattice, each node represent a concept with an extension part: a subset of the set of examples and an intension part :a digraph. The partial order between the elements of the lattice is based on AC-projection, then the digraph, intension part of a concept, can be interpreted as a compact description of a very large (potentially infinite) set of trees. But, thanks to AC-projection, we don't have to explore all the elements of this set as in classical tree mining method [8]. Using this example, we obtain a lattice which is isomorphic with the one given by Graal [10] but it is not always the case. This comes from the fact that, for this set of graphs, the set of included paths is enough to obtain this lattice.

[1] For the homomorphism relation.

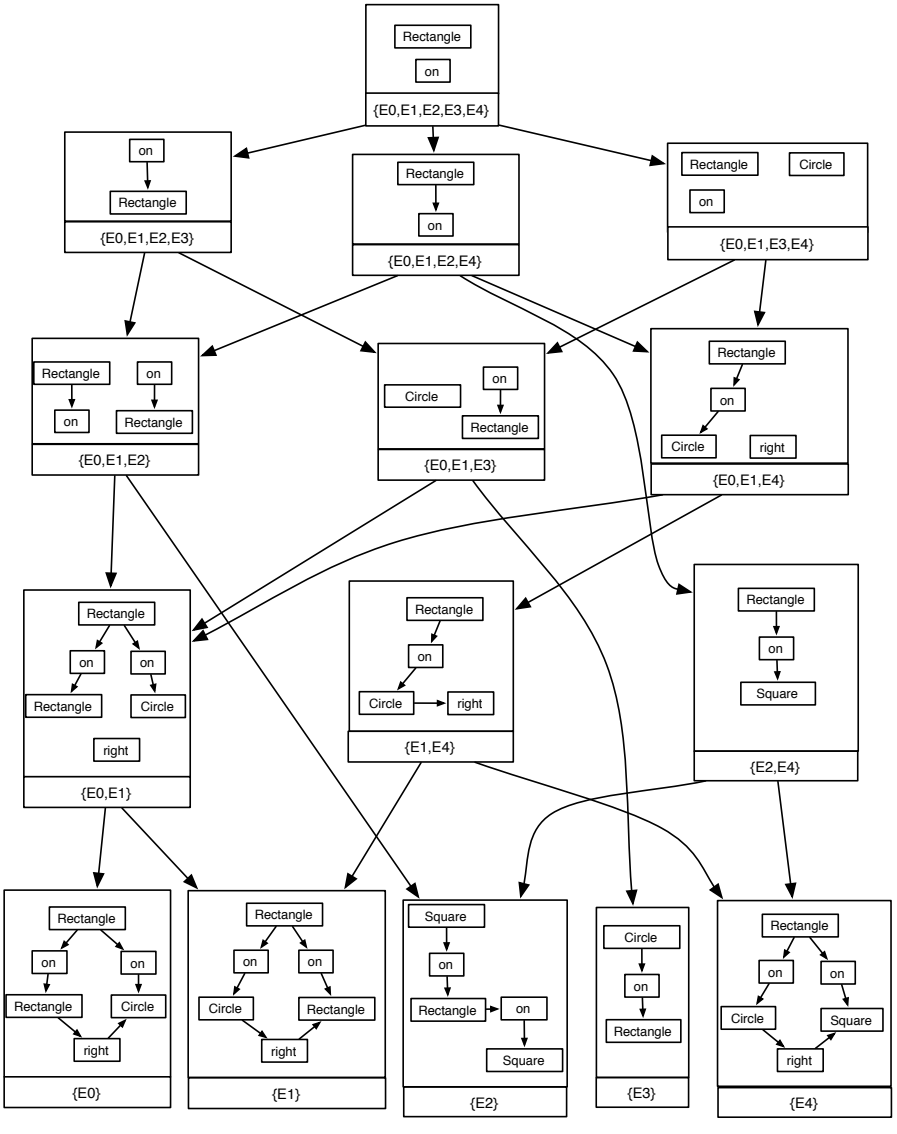

Fig. 5. A set of examples and the concept semilattice for AC-projection

5 Conclusion

This study has attempted to merge ideas from different communities: Graph, ILP, CSP. In these communities, the same problem is examined from different points of view: graph and homomorphism, logic and θ-subsumption, and constraint satisfaction problems and resolution.

Merging these knowledges we have obtained:

- the definition of a new generality relation between graphs with polynomial complexity.
- the definition of a least general generalization operator.

These results can be used for the construction of a concept lattice where the intention part of a concept is described by a digraph. The advantages of this approach compare with the approach [10] are:

- We use general digraph for the description of the examples. In the paper [10] we have a polynomial complexity only for a specific class of graphs.
- All the operations are polynomial (equivalence, reduction, product and concept order).
- We find graphs which express a large set of trees in a compact form.
- The size of the concept lattice is smaller.

But, since the AC projection is less precise, the classifications obtained are also less precise. However, we have a generalization operator, between two digraphs, which gives a digraph which represents all the homomorphic trees belonging at this two digraphs. So, a large part of their common structure is caught.

Acknowledgements. The author is grateful to C. Bessiere and F. Koriche for their constructive remarks.

References

1. Sowa, J.F.: Conceptual structures: information processing in mind and machine. Addison-Wesley Longman Publishing Co. Inc. Boston, MA, USA (1984)
2. Baget, J.F., Mugnier, M.L.: Extension of simple conceptual graphs: the complexity of rules and constraints. Journal of Artificial Intelligence Research 16, 425–465 (2002)
3. Hell, P., Nesetril, J.: Graphs and homomorphism. In: Oxford Lecture Series in Mathematics and its Application, vol. 28, Oxford University Press, Oxford (2004)
4. Mugnier, M., Chein, M.: Polynomial algorithms for projection and matching. In: ICCS. LNCS (LNAI), vol. 754, pp. 239–251. Springer, Heidelberg (1992)
5. Croitoru, M., Compatangelo, E.: A combinatorial approach to conceptual graph projection checking. In: 24th Int'l Conf. of the British Computer Society's Specialist Group on Art'l Intell, pp. 239–251. Springer, Heidelberg (2004)
6. Pfeiffer, H.: A comparison of different conceptual structures projection algorithms, Submitted to ICCS'07 (2007)
7. Maloberti, J., Sebag, M.: Fast theta-subsumption with constraint satisfaction algorithms. Machine Learning 55, 137–174 (2004)
8. Zaki, M.: Efficiently mining frequent trees in a forest. In: 8th Intl. Conf. knowledge discovery and data mining, pp. 71–80 (2002)
9. Cook, D.J., Holder, L.B.: Substructure discovery using minimum description length and background knowledge. Journal of Artificial Intelligence Research 1, 231–255 (1994)

10. Liquiere, M., Sallantin, J.: Structural machine learning with galois lattice and graphs. In: Shavlik, J.W. (ed.) ICML, pp. 305–313. Morgan Kaufmann, San Francisco (1998)
11. Sebag, M., Rouveirol, C.: Resource-bounded relational reasoning: induction and deduction through stochastic matching. Machine Learning 38, 41–62 (2000)
12. Bessiere, C.: Constraint propagation. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, Elsevier, Amsterdam (2006)
13. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer-Verlag, New York, Inc. Secaucus, NJ, USA, Translator-C. Franzke (1997)
14. Plotkin, G.: A note on inductive generalization. Machine Intelligence 5, 153–163 (1970)
15. Crole, R.: Categories for Types. Cambridge Mathematical Textbooks. Cambridge University Press, Cambridge (1993)
16. Weichsel, P.M.: The kronecker product of graphs. Proc.Am.Math.Soc. 13, 47–52 (1962)