A tree decomposition algorithm for Conceptual Graph projection

Madalina Croitoru and Ernesto Compatangelo
Department of Computing Science
University of Aberdeen (UK)

Abstract
This paper discusses combinatorial mechanisms for reasoning with conceptual graphs. We focus on the combinatorial aspects of a backtracking approach to the NP-hard problem of projection, which is the main tool for reasoning with conceptual graphs. We use effective graph-theoretical look-ahead procedures, based on a conceptual forest decomposition of the graph to be projected. We believe that our approach to projection can improve the practical applicability of exponential algorithms currently used to solve NP-hard problems.

Introduction
Conceptual graphs (CGs) are a visual knowledge representation formalism where information is encoded using a bipartite graph [11, 12], while their semantics can be mapped into that of first order logic [3]. Roughly speaking, a CG corresponds to the ABox component of a Declarative Knowledge Base (DKB) represented using a frame-based model (e.g., a Description Logic). The TBox equivalent of a DKB expressed using conceptual graphs, which is called the CG support, is managed separately. By exploiting their visual assets, conceptual graphs can be used for addressing real-world interoperability issues [5, 8]. Reasoning with conceptual graphs is based on projection [11]. This labelled graph homomorphism defines a generalisation-specialisation relation [4] such that there is a projection from a CG $G$ to a CG $H$ if $G$ is more general than $H$ ($G \geq H$).

Unfortunately, deciding whether, given two conceptual graphs $G$ and $H$, $G \geq H$ is an NP-complete problem [3, 1]. However, it has been shown that this problem is polynomially equivalent to other problems such as conjunctive query containment [2] and query output in databases [7], constraint satisfaction in combinatorial optimisation, and clause subsumption in knowledge representation and reasoning [7]. Consequently, algorithms of exponential complexity with fast execution time have been used in practical applications when the size of the graphs involved is not too large [10].

We have devised a new structure, called the matching graph, which exploits the combinatorial nature of graph homomorphism checking [6]. Following other approaches to projection [13], we relate its checking to the maximum clique problem. Our work is motivated by the belief that efficient graph theoretical techniques must be employed to enable the practical usability of projection as a reasoning mechanism. This paper details the combinatorial aspects of utilising effective graph-theoretical look-ahead procedures for projection algorithms.

The presentation is organised as follows. We introduce the main notions used throughout the paper and present the matching graph. The algorithm for the construction of reduced matching graphs, and the combination lemma follow. We then describe our novel ‘conceptual forest’ projection checking procedure. This algorithm is used to derive our method for the subsumption checking problem.

Conceptual graphs and projection
In this section, we briefly discuss and formalise a number of CG-related notions — namely background knowledge (support), simple CGs, and projection — which are needed to articulate the notion of matching graph in the next section. These definitions follow the notations introduced in [3].

Definition 1 (Bipartite graph) A graph $G = (V_G, E_G)$ with the nodes set $V_G = V_C \cup V_R$, where $V_C$ and $V_R$ are finite disjoint nonempty sets, and each edge $e \in E_G$ is a two element set $e = \{v_C, v_R\}$, where $v_C \in V_C$ and $v_R \in V_R$.

$V_C$ is the set of concept vertices of $G$ and $V_R$ is the set of relation vertices of $G$. $E_G$ is a multiset — multiple edges linking the same pair of vertices are allowed. Usually, a bipartite graph $G$ is denoted as $G = (V_C, V_R; E_G)$. The number of edges incident with a node $v \in V(G)$ is the degree $d_G(v)$ of the node $v$. An isolated vertex is a vertex $v$ with $d_G(v) = 0$.

Let $G = (V_C, V_R; E_G)$ be a bipartite graph. If, for each $v_R \in V_R$ there is a linear order $e_1 = \{v_R, v_1\}, \ldots, e_k = \{v_R, v_k\}$ on the set of edges incident to $v_R$ — where $k = d_G(v)$, then $G$ is called an ordered bipartite graph.

As a simple way of stating an ordered bipartite graph $G$, we can provide a labelling $l : E_G \rightarrow \{1, \ldots, |V_C|\}$, where $l(\{v_R, w\})$ is the index of the edge $\{v_R, w\}$ in the above ordering of the edges incident in $G$ to $v_R$. The label $l$ is called the order labelling of the edges of $G$.

An ordered bipartite graph is denoted by $G = (V_C, V_R; E_G, l)$. For a vertex $v \in V_C \cup V_R$, $N_G(v)$ de-
notes the neighbours set of \( v \), i.e. \( N_G(v) = \{ w \in V_G \cup V_R | (v, w) \in E_G \} \). Similarly, if \( A \subseteq V_R \cup V_C \), its neighbours set is denoted as \( N_G(A) = \bigcup_{v \in A} N_G(v) - A \). For each \( r \in V_R \), \( N_G^r(r) \) denotes the \( i \)-th neighbour of \( r \), i.e. \( v = N_G^r(r) \) if \( \{r, v\} \in E_G \) and \( l(\{r, v\}) = i \).

In this paper we restrict our focus to a particular type of subgraph of a bipartite graph: \( G^1 = (V_C^1, V_R^1, E_G^1) \) is a subgraph of \( G = (V_C, V_R; E_G) \) if \( V_R^1 \subseteq V_R \), \( V_C^1 = N_G(V_R^1) \) and \( E_G^1 = \{ \{v, w\} \in E_G | v \in V_C^1, w \in V_R \} \). In other words, we require that the (ordered) set of all edges incident in \( G \) to a vertex from \( V_R \) must appear in \( G^1 \). Therefore the subgraph \( G^1 \) is completely specified by its relation vertex set and will be denoted as \( [V_R^1]_G \). As we only consider bipartite graphs without isolated concept vertices, then \( G = [V_R]_G \). Moreover, if \( A, B \subseteq E_R \) then the usual union \( [A]_G \cup [B]_G \) of the graphs \( [A]_G \) and \( [B]_G \) is the subgraph \( [A \cup B]_G \). Hence, if \( V_R = \{r_1, \ldots, r_m\} \) and \( G_i \) is the (star) graph \( [\{r_i\}]_G \) \((1 \leq i \leq m)\), then \( G = G_1 \cup \ldots \cup G_m \).

Background knowledge (i.e. ontological knowledge consisting of entities organised as a partially ordered set) is encoded in a structure called support. The support is implicitly used in the representation of factual knowledge as an ordered bipartite labelled graph.

**Definition 2 (Support)**

A support is a 4-tuple \( S = (T_C, T_R, \mathcal{I}, \ast) \) where:

- \( T_C \) is a poset, i.e. a finite partially ordered set \((T_C, \leq)\) of concept types defining a type hierarchy. \( \forall x, y \in T_C : x \leq y \) means that \( x \) is a subtype of \( y \). The hierarchy has a topmost element \( \top \).
- \( T_R \) is a finite set of relation types partitioned into \( k \) posets \((T_{R_i}, \leq)\) of relation types with arity \( i \), \( 1 \leq i \leq k \), where \( k \) is the maximum arity of a relation type in \( T_R \). Each relation type \( r \in T_R \) of arity \( i \) has a signature \( \sigma(r) \in T_C \times \ldots \times T_C \), specifying the maximum concept type of each of its arguments. The notation \( r(x_1, \ldots, x_i) \) means that \( x_j \) is a concept with type \( \sigma(r)_j \) \((1 \leq j \leq i)\). The partial orders on relation types of the same arity must be signature compatible, i.e. \( \forall r_1, r_2 \in T_R : r_1 \leq r_2 \Rightarrow \sigma(r_1) \leq \sigma(r_2) \).
- \( \mathcal{I} \) is a countable set of individual markers used to refer to specific concepts, while \( \ast \) is the generic marker used to refer to an unspecified concept of a specified type. The four sets \( T_C, T_R, \mathcal{I}, \) and \( \ast \) are mutually disjoint and the union set \( T_C \cup \mathcal{I} \) is partially ordered by \( \leq \) if and only if \( x = y \) or \( y = \ast \).

A conceptual graph is a structure that depicts factual information about the background knowledge contained in its support. This information is presented in a visual manner as an ordered bipartite graph, whose nodes have been labelled with elements from the support.

**Definition 3 (Simple CGs)**

A simple conceptual graph is a 3-tuple \( SG = [S, G, \lambda] \) where:

- \( S = (T_C, T_R, \mathcal{I}, \ast) \) is a support;
- \( G = (V_C, V_R; E_G, I) \) is an ordered bipartite graph;
- \( \lambda \) is a labelling of the nodes in \( G \) with elements from the support \( S \), where \( \forall r \in V_R : \lambda(r) \in T_{R_d}(r) \) \( \forall c : c \in V_C \), \( \lambda(c) \in T_C \cup \{\ast\} \) such that \( c = N_G^r(r) \) \( \lambda(r) = t_c \), and \( \lambda(c) = (t_c, \text{ref}_c) \) then \( t_c \leq \sigma(r) \).

Therefore, each relation node \( r \) is labelled by a relation type \( \text{type}(c) \) with an associated signature of arity given by the degree of the node \( r \) in \( G \). Each concept node \( c \) is labelled by a couple \((\text{type}(c), \text{ref}(c))\), where \( \text{type}(c) \) is the type of the node \( c \) and \( \text{ref}(c) \) is the referent of \( c \). This referent either belongs to \( I \) (when \( c \) is said to be an individual concept node), or is the generic marker \( \ast \) (when \( c \) is said to be a generic concept node).

When the support is implicit, a SCG is denoted by the pair \( SG = (G, \lambda) \). If \( SG = (G, \lambda) \) is a SCG and \( G_1 \) is a bipartite subgraph of \( G \), then \( SG_1 = (G_1, \lambda_1) \), where \( \lambda_1 \) is the restriction of the labelling function \( \lambda \) to the vertices of \( G_1 \), is a conceptual subgraph of \( SG \).

The subsumption relation, which is called projection in the context of conceptual graphs, is the main tool for reasoning with simple CGs. It corresponds to deduction for the existential conjunctive and positive fragment of first order logic [2].

**Definition 4 (Projection)**

If \( SG = (G, \lambda_G) \) and \( SH = (H, \lambda_H) \) are two simple conceptual graphs defined on the same support \( S \), then a projection \( SG \rightarrow SH \) is a mapping \( \pi : V_C(G) \cup V_R(G) \rightarrow V_C(H) \cup V_R(H) \) where:

- \( \pi(V_C(G)) \subseteq V_C(H) \) and \( \pi(V_R(G)) \subseteq V_R(H) \);
- \( \forall c \in V_C(G) \) and \( \forall r \in V_R(G), if c = N_G^r(r) then \pi(c) = N_H^r(\pi(r)) \);
- \( \forall v \in V_C(G) \cup V_R(G), \lambda_G(v) \geq \lambda_H(\pi(v)) \).

The set of all projections from \( SG \) to \( SH \) is denoted as \( \Pi_{G \rightarrow H} = \{ \pi | \pi \text{ is a projection from } SG \text{ to } SH \} \). If \( \Pi_{G \rightarrow H} \neq \emptyset \) then \( SG \) is said to subsume \( SH \); symbolically, this is denoted as \( SG \geq SH \).

The subsumption relation is a pre-order on the set of all the SCGs defined on the same support. A semantics \( \Phi \) is provided in [11], which maps each SCG \( G \) based on a support \( S \) into a first order logic formula \( \Phi(G) \). If \( \Phi(S) \) is the set of formulas associated to \( S \), then for any two SCGs \( G \) and \( H \) defined on \( S \), if \( G \geq H \) then \( \Phi(S), \Phi(H) \models \Phi(G) \) (soundness). Conversely, Completeness (in formulæ, if \( \Phi(S), \Phi(H) \models \Phi(G) \) then \( G \geq H \) only holds if \( H \) is in normal form, i.e. if each individual marker appears at most once in concept node labels [9].

Although subsumption checking is an NP-complete problem [3, 1], engineering a backtracking algorithm could be computationally worthwhile. Hence, below we introduce our engineering approach to the problem.
The matching graph manoeuvre

Suppose that we want to test if $SG \geq SH$, where $SG = (G, \lambda_G)$ and $SH = (H, \lambda_H)$ are two simple conceptual graphs defined on the same support $S$. If $SG_1 = (G_1, \lambda_{G_1})$ is a conceptual subgraph of $SG$, then $\pi_{V|G_1}$ (the restriction of each $\pi \in \Pi_{G_1}$ to vertices of $G$) is a projection of $SG_1$ to $SH$.

Moreover, if $(A, B)$ is a partition of the set of relation vertices of $G$ and therefore $G = [A]_G \cup [B]_G$, then, for every $\pi \in \Pi_{G_{A-H}}$, we have $\pi|_{AU_{N_G}(A)} \in \Pi_{[A]_{G-H}}$ and $\pi|_{BU_{N_G}(B)} \in \Pi_{[B]_{G-H}}$. Clearly, if $v \in N_G(A) \cap N_G(B)$ then $\pi|_{AU_{N_G}(A)}(v) = \pi|_{BU_{N_G}(B)}(v) = \pi(v)$. The following lemma holds.

Lemma 1 (Combination lemma)

If $SG = (G, \lambda_G)$ and $SH = (H, \lambda_H)$ are two simple conceptual graphs defined on the same support $S$ and if $G = [A]_G \cup [B]_G$, where $(A, B)$ is a partition of the set of relation vertices of $G$, then $\Pi_{G_{A-H}} = \pi : V(G) \rightarrow V(H) \exists \pi_A \in \Pi_{[A]_{G-H}}, \exists \pi_B \in \Pi_{[B]_{G-H}}$ such that $\pi|_A = \pi_A, \pi|_B = \pi_B$ and $\forall v \in N_G(A) \cap N_G(B)$ then $\pi|_{AU_{N_G}(A)}(v) = \pi|_{BU_{N_G}(B)}(v) = \pi(v)$. Let $G = G_1 \cup \ldots \cup G_m$, where $G_i$ is the (star) graph $\{r_i\}_G \ (1 \leq i \leq m)$ and $V_{G_i} = \{r_1, \ldots, r_m\}$. By iteratively using the above lemma, we construct the set $\Pi_{G_{A-H}}$ by finding all the mutually compatible projections of the star graphs $G_i$ to $H$. A projection of $G_i$ to $H$ is completely determined by a star graph $H_i$ of $H, H_i = \{s_i\}_H$ with the property $\lambda_G(r_i) \geq \lambda_H(s_i)$ and $\forall j = 1, \lambda_i = d_G(r_i) = d_H(s_i) : \lambda_G(N_G^i(r_i)) \geq \lambda_H(N_H^i(s_i))$.

Let $(r_i, s_i)$ and $(r_j, s_j)$ describe individual projections of star graphs $G_i$ and $G_j$; these projections are compatible if $N_G^i(r_i) = N_G^i(r_j) \Rightarrow N_H^i(s_i) = N_H^i(s_j)$.

The matching graph $M_{G_{A-H}}$ is a conceptual tool for managing the mutual compatibilities between individual projections of the star graphs of $G$. The vertices of this graph are pairs $(r, s)$, where $r \in V_{G}(G)$ and $s \in V_{H}(H)$ represent the projection of the star graph $\{r\}_G$ into $\{s\}_H$.

The edges of $M_{G_{A-H}}$ are $(V, E)$ show that their extremities represent a pair of compatible projections. More precisely:

Definition 5 Let $SG = (G, \lambda_G)$ and $SH = (H, \lambda_H)$ be two simple conceptual graphs without isolated concept vertices defined on the same support $S$. The matching graph of $SG$ and $SH$ is the graph $M_{G_{A-H}} = (V, E)$ where:

1. $V \subseteq V_{G}(G) \times V_{H}(H)$ is the set of all pairs $(r, s)$ such that $r \in V_{G}(G)$, $s \in V_{H}(H)$, $\lambda_{G}(r) \geq \lambda_{H}(s)$ and for each $i \in \{1, \ldots, d_G(r)\}$, $\lambda_G(N_G^i(r)) \geq \lambda_H(N_H^i(s))$.
   For each $r \in V_{G}(G)$ let $V_r = \{(r, s) \in V(M_{G_{A-H}})\}$.

2. $E$ is the set of all 2-sets $\{r, s\} \cup \{r', s'\}$, where $r \neq r'$, $(r, s), (r', s') \in V$ and for each $i \in \{1, \ldots, d_G(r)\}$ and $j \in \{1, \ldots, d_G(r')\}$ such that $N_G^i(r) = N_G^i(r')$ the condition $N_H^j(s) = N_H^j(s')$ holds.

The sets $V_r$ are disjoint, their union is $V(M_{G_{A-H}})$, and no two vertices of the same $V_r$ are adjacent in $M_{G_{A-H}}$ (i.e. each $V_r$ is a stable set in this graph). This means that $(V_r)_{r \in V_{G}(G)}$ is a colouring of the matching graph $M_{G_{A-H}}$.

A $q$-clique in a graph $F$ is a set of $q$ pairwise adjacent vertices. The maximum positive number $q$ for which there is a $q$-clique in $F$ is denoted by $\omega(F)$. We denote by $C_G$ the family of all $\omega(F)$-cliques of the graph $F$.

Clearly $\omega(M_{G_{A-H}}) \leq |V_{H}(G)|$, since no clique of $M_{G_{A-H}}$ can have more than a vertex from a stable set $V_r$. From the above discussion it follows that if $\pi \in \Pi_{G_{A-H}}$ is a projection from $SG$ to $SH$, then $\{(r, \pi(r)) : r \in V_{G}(G)\}$ is a $|V_{H}(G)|$-clique in $M_{G_{A-H}}$. Conversely, if $\{(r_1, \pi(r_1)), \ldots, (r_m, \pi(r_m))\}$ is a $|V_{G}(G)|$-clique in $M_{G_{A-H}}$, then taking for each $i = 1, m$, $\pi(r_i) \leftarrow s_i$ and $\pi(N_{G}(r_i)) \leftarrow \pi(N_{H}(s_i))$ (for each $j$-neighbour of $r_i$ in $G$), we obtain a projection of $SG$ to $SH$. Hence, the following theorem holds.

Theorem 1 Let $SG = (G, \lambda_G)$ and $SH = (H, \lambda_H)$ are two SCGs without isolated concept vertices defined on the same support $S$ and let $M_{G_{A-H}} = (V, E)$ be their matching graph. The following two conditions hold:

1. $SG \geq SH$, that is $\Pi_{G_{A-H}} \neq \emptyset$, and if only if $\omega(M_{G_{A-H}}) = |V_{H}(G)|$

2. If $\Pi_{G_{A-H}} \neq \emptyset$, then there is a 1:1 mapping between $C_{G_{A-H}}$ and $\Pi_{G_{A-H}}$.

Consequently, the problem of deciding whether $SG \geq SH$ is reduced to the problem of testing whether the matching graph $M_{G_{A-H}}$ has a clique with exactly one vertex in each stable set $V_r$. If any $V_r$ is empty (or if there is a vertex $(r, s) \in V_r$ with no neighbour in some $V_{r'}$ with $r \neq r'$), then there is no $|V_{G}(G)|$-clique in $M_{G_{A-H}}$ that contains $(r, s)$.

The reduced matching graph (RMG) of $SG$ and $SH$ is the graph obtained from $M_{G_{A-H}}$ by eventually deleting a vertex with no neighbour in some nonempty stable set $V_r$ that does not contain that vertex. The RMG is denoted as $RM_{G_{A-H}}$. It follows that $\omega(M_{G_{A-H}}) = |V_{H}(G)| = \omega(RM_{G_{A-H}}) = |V_{H}(G)|$ and the above theorem holds with $RM_{G_{A-H}}$ instead of $M_{G_{A-H}}$.

This reduction could be computationally worthwhile, since it removes from the very beginning some individual projections of the relation vertices of $G$, which do not belong to a projection of the whole graph $G$.

Reduced matching graph algorithm

Our algorithm devised to construct reduced matching graphs is as follows.

1. for each $r \in V_{G}(G)$ do 
   $V_r = \emptyset$
   for each $s \in V_{H}(H)$ do
   if $\lambda_{G}(r) \geq \lambda_{H}(s)$
   and $\lambda_{G}(N_{G}(r)) \geq \lambda_{H}(N_{H}(s))_{i=1}d_{G}(r)$
   then $V_r \leftarrow V_r \cup \{(r, s)\}$
2. for each $r \in V_{G}(G)$ do
   for each $(r, s) \in V_r$ do

Let \( m_G \) be the number of relation vertices of graph \( G(H) \) and \( e_G \) the number of edges of graph \( G(H) \). If the comparison of each label is accomplished in constant time, then the time complexity of algorithm step 1 is \( O(m_G \cdot m_H \cdot e_G) \). Moreover, as the number of vertices of graph \( M_{G-H} \) is \( O(m_G \cdot m_H) \), the time complexity of algorithm step 2 is \( O(m_G^2 \cdot m_H^2) \). In algorithm step 3, the vertices \( (r, s) \) of graph \( M_{G-H} \) are collected in the stack \( S \) in \( O(m_G^2 \cdot m_H^2) \). These vertices will be deleted, since they have no neighbours in some nonempty stable set \( V_r(r' \neq r) \). The vertex deletion process is controlled in order to capture all the new occurrences of a vertex that must be deleted in the current graph. In this case, each vertex occurrence is added to the stack \( S \). The overall complexity of algorithm step 3 is dominated by step 2. Hence, graph \( RM_{G-H} \) can be constructed in \( O(m_G^2 \cdot m_H^2) \) time. The projection checking algorithm that follows from Theorem 1 can be now described as follows:

1. Construct the reduced matching graph \( RM_{G-H} \);
2. Find the clique number \( w(RM_{G-H}) \);
3. If \( w(RM_{G-H}) < |V(G)| \), then return \( \text{"G \geq H\"} \) and a clique \( Q \in C_{\omega(RM_{G-H})} \) else return \( \text{\"G \geq H\"} \) and a clique \( Q \in C_{\omega(RM_{G-H})} \)
Let \( \{r_j, s\} = Q \cap V_{r_j} \) \((r_{i+1}, s)\) be any vertex from \( Adj((r_j, s), r_{i+1}) \) and let \((r, s)\) be an arbitrary vertex of \( Q \). If \( N_{F}(r) \cap N_{F}(r_{i+1}) \neq \emptyset \), then \( N_{F}(r) \cap N_{F}(r_{i+1}) = N_{F}(r) \cap N_{F}(r_{i+1}) \). Since \((r, s)\) is adjacent to \((r_{i+1}, s)\), it follows that \((r, s)\) is adjacent to \((r_{i+1}, s)\). If \( N_{F}(r) \cap N_{F}(r_{i+1}) = \emptyset \) then \((r, s)\) is adjacent to \((r_{i+1}, s)\).

We have obtained that \((r_{i+1}, s)\) is adjacent in \( RM_{F-H} \) to all vertices in \( Q \), i.e. \( Q \cup \{(r_{i+1}, s)\}\) is an \((i+1)\)-clique, and the theorem is thus proved.

Furthermore, it follows from it that the recursive algorithm introduced below can be used to generate the set of all maximum cliques of \( C_{RM_{F-H}} \), i.e. \( \Pi_{F-H} \).

**Generate(leveled)**

**Input:** leveled the current size of the clique constructed.

First call is for leveled = 0.

**Global variables:** Array \( Q \), the current clique constructed.

Reduced matching graph \( RM_{F-H} \); sets \( V_{r_i} \neq \emptyset \) and adjacency lists \( Adj((r_i, s)) \) for each \((r_i, s) \in V_{r_i} \) and \( i \neq j \in \{1, \ldots, m\} \).

Array parent associated to an acyclic ordering: \( \{r_1, \ldots, r_m\} \) \( \in V_R(F) \).

**Output:** \( C_{RM_{F-H}} \), the set of all m-cliques of \( RM_{F-H} \), initially empty.

\[
\begin{align*}
\{ \text{level} + 1 \} & \\
\text{if level} > m \text{ then } C_{RM_{F-H}} & \leftarrow C_{RM_{F-H}} \cup Q & \\
\text{else if } \text{parent}[r_{\text{level}}] & \not= r_{\text{level}} & \\
& \text{then for each } (r_{\text{level}}, s) \in V_{r_{\text{level}}} \text{ do } \{ \text{level} \leftarrow (r_{\text{level}}, s); \}
\text{Generate(level) } \} & \\
\text{else } & \\
& \text{let } (r_j, s) \leftarrow Q[\text{parent}[r_{\text{level}}]] & \\
& \text{for each } (r_{\text{level}}, s) \in Adj((r_j, s)) \text{ do } \{ \text{level} \leftarrow (r_{\text{level}}, s); \}
\text{Generate(level) } \} & \\
\end{align*}
\]

In the above algorithm the time complexity per clique generated is \( O(m) \).

The algorithmic implications of Theorem 2 w.r.t. the general projection checking problem ‘\( SG \geq SH ? \)’ (where \( SG \) is an arbitrary SCG) are facilitated by the following two RMG properties, which describe an effective application of the Combination Lemma.

Let \( SG \geq SH \), \( A \subseteq V_R(G) \) and let \( RM_{A-H} \) be the subgraph of \( RM_{G-H} \) induced by the set of vertices \( \{(r, s) \in V_R(G) \mid r \in A\} \). In our representation of the reduced matching graph (as described in Section 2), graph \( RM_{A-H} \) can be obtained by considering only the adjacency lists of these vertices with respect to \( A \). Note that \( RM_{A-H} \) is a subgraph of the matching graph \( RM_{[\{A\}, A-H]} \). Statement (ii) in Theorem 1 implies the following lemma.

**Lemma 2** The set of all projections of conceptual subgraph \( [A]_{G} \) to \( SH \), which can be extended to a projection of the entire simple conceptual graph \( SG \) into \( SH \), is contained in \( C_{RM_{[\{A\}, A-H]}} \).

Moreover, let \( V_R(G) = A \cup B \) \((A, B \neq \emptyset, A \cap B = \emptyset)\), and \( Q \) be an \([A]-\)clique of \( RM_{A-H} \). We denote by \( RM_{G-H} \mid Q \) the subgraph of \( RM_{B-H} \) induced by the set of vertices \( \{(r, s) \in V_{RM_{G-H}B} \mid r \in B, Q \subseteq N_{RM_{G-H}}(r)\} \).

**Lemma 3** The set of projections of the conceptual subgraph \([B]_{G} \) to \( SH \), which extend the projection of \([A]_G \) represented by \( Q \) to a projection of the entire \( SG \) to \( SH \), is contained in \( C_{RM_{B-H}Q} \).

We repeatedly use these lemmas to decompose an arbitrary SCG \( SG = (G, \lambda_G) \) into conceptual forest subgraphs. This is due to the algorithmic reductions described in Theorem 2 and the algorithm previously designed.

**Definition 7** Let \( SG = (G, \lambda_G) \) be an arbitrary SCG. An acyclic decomposition of \( SG \) is a union \( F_1 \cup \ldots \cup F_k \) such that \( \{V_R(F_1), \ldots, V_R(F_k)\} \) is a partition into \( k \) classes of \( V_R(G) \), where \( 1 \leq k \leq |V_R(G)| \) and \( F_i \) are conceptual forest subgraphs.

Since the number \( k \) of conceptual subforests in which a conceptual graph can be decomposed is relatively small in practical applications, here we are not concerned with the minimisation of this parameter (which we suspect to be a \( NP \)-hard problem). Instead, we use the following greedy method to obtain an acyclic decomposition.

**Input:** SCG \( SG = (G, \lambda_G) \)

**Output:** Acyclic decomposition \( G = F_1 \cup \ldots \cup F_k \), where \( F_i = \{(r_1, \ldots, r_m)\} \) \((i = 1, k)\) is a conceptual subforest of \( G \).

\[
\begin{align*}
&\{ r_j \leftarrow r \text{. an arbitrary relation vertex of } V_R(G) \}
&\quad m_j \leftarrow 1; k \leftarrow 1; V_R(G) \leftarrow V_R(G) \setminus \{r\}
&\quad \text{for each } r \in V_R(G) \text{ do } \{ \text{placed } \leftarrow \text{false; } j \leftarrow 1 \}
&\quad \text{while } (j \leq k & \text{ not placed }) \text{ do } \{
&\quad \text{if } N_G(r) \cap N_G((r_1, \ldots, r_m)) = \emptyset \text{ then } \}
&\quad \quad \{ m_j + 1; r_m \leftarrow r; \text{parent}[r_m] \leftarrow r_m; \}
&\quad \quad \text{placed } \leftarrow \text{true; } \}
&\quad \text{else } \{ \}
&\quad \quad \text{if } |N_G(r) \cap N_G((r_1, \ldots, r_m))| = 1 \text{ then } \}
&\quad \quad \quad \{ m_j + 1; r_m \leftarrow r; \text{placed } \leftarrow \text{true; } \}
&\quad \quad \quad \text{find } r_j \text{ such that } N_G(r) \cap N_G(r_j) \neq \emptyset; \}
&\quad \quad \quad \text{parent}[r_m] \leftarrow r_j; \}
&\quad \quad \text{else } j \leftarrow \}
&\quad \text{if not placed then } \{
&\quad \quad \{ k + 1; m_j \leftarrow 1; r_k \leftarrow r; \}
&\quad \quad \text{parent}[r_k] \leftarrow r \} \}
&\quad \text{The above procedure can be used as a preprocessing ‘step 0’ in the projection checking algorithm outlined at the end of the section describing the reduced matching graph construction algorithm. Step 1 remains unchanged, with the reduced matching graph \( RM_{G-H} \) encoding (by way of Lemma 2) the entire set of reduced matching graphs \( RM_{B-H} \) \((i = 1, k)\). Steps 2 and 3 are implemented using a backtracking scheme working with ‘big’ steps corresponding to each conceptual forest constructed in the preprocessing step. More precisely, we maintain a current partial solution \( Q_1, \ldots, Q_h \) describing the current projection of \( F_1 \cup \ldots \cup F_h \) to \( H \). From
Lemma 3 the compatible projections of the remaining graph \( F_{h+1} \cup \ldots \cup F_h \) must be searched in graph \( R M^h \), which only contains the \( R M^{h-1} \) vertices that are adjacent to all vertices in \( Q_1, \ldots, Q_h \). Initially, that holds for \( h = 0 \), \( R M^0 \) simply being \( R M_G \).

In the following description of the algorithm, we use the term ‘for each clique in \( C_{RM_{F \rightarrow H}} \)’. It denotes the call of \textit{Generate} in order to determine the maximum cliques of \( R M_{F \rightarrow H} \) when \( F \) is a conceptual forest.

\[
\text{Input: SCG's } SG = (G, \lambda_G) \text{ and } SH = (H, \lambda_H) \\
\text{Output: Either } G \not\geq H \text{ or } "G \geq H" \text{ and clique } Q \text{ encoding the projection constructed.}
\]

1. **Preprocessing step:**
   - Find an acyclic decomposition \( G = F_1 \cup \ldots \cup F_k \);
   - Construct the reduced matching graph \( R M_{G \rightarrow H} \);
   - \( R M^0 \leftarrow R M_{G \rightarrow H} \); \text{found} \leftarrow \text{false};

2. **Find Projection(0);**
   - if \text{not found} then \( G \not\geq H \) else \( G \geq H \): \( Q = Q_1 \cup \ldots \cup Q_h \).

3. **Find Projection(h);**
   - if \text{not found} then
     - if \( h = k \) then \text{found} \leftarrow \text{true}
     - else if (\( \forall r \in \cup_{i=1}^{h+1} V_R(F_i), R M^h \) has \( V_r \neq \emptyset \))
       - then for each clique \( Q \in C_{RM_{F \rightarrow H}} \) do { \\
         - construct \( R M^{h+1} \), \\
         - \( Q_{h+1} \leftarrow Q \); \\
         - \text{Find Projection}(h+1) }

**Conclusions**

In this paper we introduced a further step along the [6] results, discussing an approach that uses graph-theoretical tools to speed-up reasoning with CGs. Our approach uses effective look-ahead graph theoretical procedures based on a conceptual forest decomposition of the graph to be projected. Until now, the various approaches to conceptual graph projection proposed in the literature did not fully use the rich combinatorial structure of CGs. By exploiting this structure, our algorithm is able to eliminate unneeded comparisons, providing improved complexity results over existing techniques. Further efficiencies are obtained by studying a graph’s acyclic tree decomposition in the context of its corresponding reduced matching graph. These techniques, in our opinion, will be useful to any CG based system making use of projection.

As previously mentioned, conceptual graph reasoning is polynomially equivalent to constraint satisfaction in combinatorial optimization. Further investigation of tree decomposition mechanisms from a constraint satisfaction point of view, and applying them to our work, promises to be a fruitful future area of research.

**References**


