ACYCLIC HYPERGRAPH PROJECTIONS
AND RELATIONSHIPS TO CIRCULAR-ARC GRAPHS
AND CIRCULAR REPRESENTABLE HYPERGRAPHS

by

Aviv Lustig and Oded Shmueli

Technical Report #385
November 1985
ABSTRACT

Hypergraphs can be partitioned into two classes: tree (acyclic) hypergraphs and cyclic hypergraphs. This paper analyzes a new class of cyclic hypergraphs called Xrings. Hypergraph \( H \) is an Xring if the edges of \( H \) can be circularly ordered so that for every vertex, all edges containing the vertex are consecutive; in addition no edge may be a subset of another edge and no vertex may appear in exactly one edge.

Let \( H_1 \) and \( H_2 \) be two hypergraphs. A tree projection is a tree hypergraph \( H_3 \) such that each edge in \( H_1 \) is contained in some edge of \( H_3 \) and each edge in \( H_3 \) is contained in some edge of \( H_2 \). A polynomial time algorithm is presented for deciding, given Xring \( H_1 \) and arbitrary hypergraph \( H_2 \), whether there exists a tree projection. It is shown that hypergraph \( H \) is an Xring iff a modified adjacency graph of \( H \) is a circular-arc graph. Finally, relationships between Xrings and circular representable hypergraphs are investigated.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems - computation, and discrete structures; G.2.2 [Discrete Mathematics]: Graph Theory - graph algorithms; trees; H.2.4 [Database Systems]: Query Processing

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Acyclic hypergraph, tree projection, circular-arc graph, computational complexity, relational databases
1. INTRODUCTION

Hypergraphs can be partitioned into two classes: tree (acyclic) hypergraphs and cyclic hypergraphs. H is a tree hypergraph if its edges can be placed as nodes of a tree T so that for every vertex x of H, the subtree of T defined by nodes containing x is connected. (Precise definitions are given in Section 2, see [1] regarding basic hypergraph definitions.) This work introduces a new type of cyclic hypergraphs called Xrings, generalizing the Arings and Acliques of [8]. Hypergraph H is an Xring if the edges of H can be ordered around a circle so that for every vertex, all edges containing the vertex are consecutive; in addition no edge may be a subset of another edge and no vertex may appear in exactly one edge.

The adjacency graph (2-section) of hypergraph H is an undirected graph G whose vertices are the vertices of H, such that there exists an edge connecting two vertices iff they appear together in some edge of H. Let H_u be the hypergraph obtained from H by adding to each edge a new unique vertex. It is shown that Hypergraph H is an Xring iff the adjacency graph of H_u is a circular-arc graph [5, 12, 13]. A polynomial time algorithm for recognizing Xrings can be constructed using the algorithm for recognizing circular-arc graphs presented in [13].

A tree projection of hypergraph H_2 with respect to hypergraph H_1 is a tree hypergraph H_3, such that each edge of H_1 is a subset of some edge of H_3 and each edge of H_3 is a subset of some edge of H_2. The problem of deciding whether a tree projection exists is still open in the general case, it is clearly in NP (see [4] for definitions). We present a polynomial time algorithm for solving this problem when H_1 above is an Xring.

Hypergraphs play an important role in the theory of relational database (see [14] for an overview of relational databases). A relational database schema can be seen as a hypergraph, with attributes corresponding to vertices and relation schemas corresponding to hyperedges. The motivation for studying tree projections originated with [7]. It was shown in [7] that if a finite program P over the operations project, join and semijoin solves a relational query over a set of relations (i.e.
hypergraph) $H_1$, then it must transform $H_1$ into a new set of relations (hypergraph) $H_2$, such that there exists a tree projection of $H_2$ with respect to $H_1$. Therefore, the algorithm presented in this paper is not merely of theoretical interest. It may be used in query processors within database systems. Identifying the existence of tree projections provides new options for processing queries [9].

Finally, the relationship between Xrings and the interval representable hypergraphs and circular representable hypergraphs (of [3] and [10]) are explored. It is shown that Xrings and circular representable hypergraphs are distinct but not disjoint.

Section 2 presents basic concepts and definitions. In Section 3, a connection between Xrings and circular-arc graphs is established. In Section 4, some properties of tree projections with respect to Xrings are proved. A polynomial time algorithm for identifying a tree projection with respect to Xrings is given in Section 5. Section 6 investigates relationships between Xrings, interval representable hypergraphs and circular representable hypergraphs. Section 7 presents conclusions and open problems.
2. TERMINOLOGY

2.1. Graphs and Hypergraphs

A **graph** is a pair \((V,E)\) where \(V\) is a set of **vertices** (or **nodes**) and \(E\) is a set of unordered pairs of vertices, called **edges**. A graph \(G=(V,E)\) is a **circular-arc graph** if there exists a 1-1 correspondence between the vertices of \(G\) and a family of arcs of a circle such that two distinct vertices are adjacent in \(G\) iff the corresponding arcs intersect. The family of arcs is called a **(circular-arc) model** for \(G\). A **proper** circular-arc graph has a model in which no arc is contained in another arc. Without loss of generality (w.l.o.g.) we may assume that arcs are closed, i.e. include their endpoints. (See [5] regarding circular-arc graphs.) We adopt the convention that a circle is traversed clockwise. The first endpoint of an arc is called a **startpoint** and the second endpoint is called an **endpoint**. If an arc spans the whole circle then its startpoints and endpoints are equal.

A **hypergraph** is a pair \((X,H)\), where \(X\) is a set of **vertices** (or **nodes**) and \(H\) a family (i.e. a multiset) of subsets of \(X\) called **edges**. We usually denote a hypergraph by its set of edges \(H\), assuming that \(X=\bigcup e\). The **length** of hypergraph \(H\) is \(|H|=\sum_{e\in H}|e|\). The **adjacency graph** (or **2-section**) of a hypergraph \(H\) is the graph \(AG(H)=(X,\{(x_i,x_j)\}:\{x_i,x_j\}\subseteq e, \text{ for some } e\in H\})\).

Let \(G=(V,E)\) be an undirected graph whose nodes are in one-to-one correspondence with the hyperedges of hypergraph \(H\). We say \(G\) is **\(x\)-connected** if the subgraph of \(G\) induced by hyperedges (nodes) containing vertex \(x\) is connected. \(G\) is **\(Y\)-connected** if for all \(x\in Y\), \(G\) is **\(x\)-connected**. \(G\) is a **qual graph** for \(H\) if it is **\(X\)-connected** [2]. This property of a qual graph is called **vertex connectivity**. A qual graph for hypergraph \(H\) is **minimal** if there exists no other qual graph for \(H\) with a smaller number of edges.

---

1 We use traditional graph theory notation. All structures in this paper are finite.

2 When no confusion may arise we let \(V=H\) and talk about "node \(e_i\)" rather than "node \(v_i\in V\) corresponding to \(e_i\)."
H is a tree hypergraph (equivalently, acyclic hypergraph) if some qual graph for it is a tree; otherwise H is a cyclic hypergraph. A vertex \( x \in X \) is isolated if it appears in exactly one hyperedge. The following simple procedure, called Graham reduction and discovered independently by [8] and [15], recognizes tree hypergraphs.

**Procedure GR**

Apply the following two steps until neither is applicable:

1. Delete any isolated vertex.
2. Find two hyperedges \( e_r \) and \( e_s \) in \( H \) such that \( e_r \subseteq e_s \); delete \( e_r \) from \( H \).

We denote by \( GR(H) \) the output of procedure GR when given hypergraph \( H \) as input. It can be shown that \( H \) is a tree hypergraph iff upon termination of GR the resulting hypergraph consists of a single (empty) hyperedge, i.e. \( GR(H) = \emptyset \). A hypergraph \( H \) is Graham reduced if \( GR(H) = H \). (A linear time algorithm for recognizing tree hypergraphs appears in [11].)

**Example 2.1.** The following is a tree hypergraph: \( H_1 = \{1,2\}, \{1,4\}, \{1,5\}, \{2,3\}, \{1,2,3\}, \{1,4,5\} \). A qual graph for \( H_1 \) is:

\[
\begin{align*}
\{1,2\} & \rightarrow \{1,2,3\} \rightarrow \{1,4,5\} \rightarrow \{1,4\} \\
\{2,3\} & \qquad \{1,5\}
\end{align*}
\]

The hypergraph \( H_2 = \{1,2\}, \{2,3\}, \{3,4\}, \{4,1\} \) is cyclic. Any qual graph for \( H_2 \) is isomorphic to:

\[
\begin{align*}
\{1,2\} & \rightarrow \{2,3\} \\
\{1,4\} & \rightarrow \{3,4\}
\end{align*}
\]

**2.2. Tree Projections**

\( H \) is a projection of \( H' \), denoted \( H \preceq H' \), if for every hyperedge \( e \in H \) there exists a hyperedge \( e' \in H' \) such that \( e \subseteq e' \). Let \( H \) and \( H' \) be hypergraphs. We say that \( H' \) is a tree projection (TP) of \( H' \) with respect to \( H \) if \( H' \) is a tree hypergraph
and $H':=H' \leq H' [7]$. When $H'$ is a TP of itself w.r.t. $H$, we usually just say that $H'$ is a TP w.r.t. $H$. Because $H \leq H'$, for the problems addressed in this paper, we may assume that $H \leq H'$. In the remainder of this paper we use $H=(e_1,\ldots,e_n)$ and $H':=(e_1,\ldots,e_n,e_{n+1},\ldots,e_{n+m})$. We call $e_i$ ($1 \leq i \leq n$) base hyperedges; $e_i$ ($n+1 \leq i \leq n+m$) are called non-base hyperedges.

**Example 2.2.** Consider the following hypergraphs.

- $H=((1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,8),(8,1))$
- $H':=H\cup (\{1,2,3,8\},\{3,4,7,8\},\{4,5,8,7\})$
- $H':=(\{1,2,5,6\},\{1,2,3,5,8\},\{1,3,4,7,8\},\{4,5,8,7\},\{5\})$

Clearly both $H$ and $H'$ are cyclic, and $H'$ is a tree hypergraph. $H \leq H' \leq H'$, and therefore $H'$ is a TP of $H'$ w.r.t. $H$.

2.3. Xrings

A hypergraph $H$ over $m$ distinct vertices $x_1,\ldots,x_m$ is an Xring (of size $n \geq 3$) if $H=GR(H)$ and its minimal qual graph is a cycle of size $n$. Clearly any Xring is a cyclic hypergraph. Since an Xring $H$ is Graham reduced, in any qual graph for $H$ every hyperedge must have vertex degree $\geq 2$. It follows that every minimal qual graph for $H$ is a cycle. Note that there may be several non-isomorphic cycle qual graphs for an Xring. In the sequel we assume that the hyperedges of an Xring $H$ are numbered $e_1,e_2,\ldots,e_n$ according to the order of their appearance in some cycle qual graph $Q$ for $H$, starting with some chosen hyperedge and going clockwise. Hyperedges $e_i$ and $e_j$ are said to be overlapping if $|j-i|=1$. Observe that overlapping is a property with respect to a particular enumeration.

**Example 2.3.** Hypergraph $H=((1,4,5),(1,4,5,8),(1,4,6,8),(1,2,4,6,7),(1,2,3,7),(1,2,3,4))$ is an Xring of size 6 over vertices $1,2,\ldots,8$. Figure 1 shows the minimal qual graph for $H$.

An Aring (of size $n$) is a hypergraph isomorphic to the hypergraph $(\{x_1,x_2\},\{x_2,x_3\},\ldots,\{x_n,x_1\})$ for distinct vertices $x_1,\ldots,x_n$ and $n \geq 3$. Clearly, Arings
Lemma 2.1. If \( H \) is an Xring of size \( n \), then there exists an Aring \( H_A \) of size \( n \) such that \( H_A \leq H \).

Proof. Let \( Q \) be a cycle qual graph for \( H \). Starting with some chosen hyperedge, number the hyperedges in \( H \) by their order on the cycle. For \( 1 \leq i \leq n \), consider hyperedges \( e_i \) and \( e_{i+1} \). Since \( H \) is Graham reduced, there exists some vertex \( x_i \in e_i - e_{i+1} \). If \( x_i \in e_{i+1} \), then \( e_i \leq e_{i+1} \); otherwise, either \( H \) is not Graham reduced or \( Q \) is not \( x_i \)-connected. The vertices thus obtained satisfy \( x_i \neq x_j \) for \( i \neq j \), since otherwise \( x_i \in e_i, x_i \in e_j \), but \( x_i \notin e_{i+1} \) and \( x_i \notin e_{j+1} \), contradicting vertex connectivity in \( Q \). The hypergraph \( H_A = \{x_1, x_2, \ldots, x_n\} \) is an Aring of size \( n \). By construction, \( \{x_i, x_{i+1}\} \subseteq e_i \), therefore \( H_A \leq H \).

It follows as an immediate corollary that \( m \geq n \).

\[3\] Arings, together with Acliques, are basic "building blocks" for all cyclic hypergraphs [6].
3. XRINGS AND CIRCULAR-ARC GRAPHS

In this section we show a connection between Xrings and circular-arc graphs and obtain an algorithm for recognizing Xrings in polynomial time. Given hypergraph $H$, let $H_u$ be a new hypergraph obtained from $H$ by adding to each hyperedge $e_i \in H$ a new unique vertex $y_i$.

**Lemma 3.1.** If $H$ is an Xring then $AG(H)$ and $AG(H_u)$ are circular-arc graphs.

**Proof.** Let $H$ be an Xring of size $n$, and $Q$ a cycle qual graph for $H$. For each vertex $x_i \in X$, let $e_{\text{first}(i)}$ (resp. $e_{\text{last}(i)}$) be the first (resp. last) hyperedge of $H$ in which vertex $x_i$ appears (clockwise around $Q$). If vertex $x_i$ appears in all hyperedges then $\text{last}(i) = \text{first}(i) - 1$. Since $H$ is an Xring, and by the enumeration of hyperedges, for $\text{first}(i) \leq j \leq \text{last}(i)$, $x_i \in e_j$, and for $\text{last}(i) < j < \text{first}(i)$, $x_i \notin e_j$.

Consider a circle divided into $n$ equal segments. The endpoints of a segment are called *markers*. (Markers may be visualized as the hour markers of a clock with $n$ hours.) Let $e_k$ be the $k$'th marker. For every vertex $x_i \in X$ draw a closed arc $a_i$ from marker $e_{\text{first}(i)}$ to marker $e_{\text{last}(i)}$. Intuitively, marker $e_i$ is hyperedge $e_i$, i.e. the hyperedge includes all vertices corresponding to arcs passing through marker $e_i$. By construction arc $a_i$ includes point $e_k$ iff $x_i \in e_k$. Consider the intersection graph $G=(V,E)$ of the arcs just drawn which, by definition, is a circular-arc graph. $G$ is isomorphic to $AG(H)$, since

\[
\{a_i, a_j\} \in E \iff \text{arcs } a_i \text{ and } a_j \text{ intersect} \\
\iff \text{arcs } a_i \text{ and } a_j \text{ intersect at some marker } e_k \\
\iff \text{for some } e_k \in H, x_i \in e_k \text{ and } x_j \in e_k \\
\iff \{x_i, x_j\} \text{ is an edge in } AG(H).
\]

For each $e_i \in H$, add now an open arc $b_i$ starting midway between markers $e_{i-1}$ and $e_i$ and ending midway between markers $e_i$ and $e_{i+1}$. The intersection graph obtained from the $a$-arcs, together with the $b$-arcs, is now $AG(H_u)$; it is a circular-arc graph by definition. $\blacksquare$

---

$4 \cdot 1 - 1 \equiv n$. 

Technion - Computer Science Department - Technical Report CS0385 - 1985
**Lemma 3.2.** Let $H$ be a cyclic hypergraph satisfying $H = GR(H)$. If $AG(H_u)$ is a circular-arc graph then $H$ is an $X$-ring.

**Proof.** In any circular-arc model for $AG(H_u)$, let arcs $a_i$ correspond to vertices $x_i$ in $AG(H_u)$, and arcs $b_i$ correspond to vertices $y_i$ in $AG(H_u)$. (Vertices $y_i$ are the unique vertices added to $H$.) Renumber arcs $b_i$ and hyperedges $e_i$ according to the clockwise order of the $b$-arcs around the circle (starting with some $b$-arc). This order is well defined since no two $b$-arcs intersect. We show that the cycle $Q$, whose nodes are the hyperedges of $H$ numbered according to the order of their corresponding $b$-arcs in the circular-arc model, is a qual graph for $H$.

**Claim:** $Q$ is $X$-connected.

**Proof of claim.** For $1 \leq i \leq m$, consider vertex $x_i$ and the corresponding arc $a_i$ in the model. For all hyperedges $e_j$ that include vertex $x_i$, arc $b_j$ intersects arc $a_i$ in the model. Also, there is no hyperedge $e_k$ which does not include $x_i$ such that $b_k$ intersects $a_i$. Furthermore, since $a_i$ is an arc, all arcs $b_k$ intersecting with $a_i$ appear consecutively along the circle. Since the hyperedges in $Q$ are ordered according to their corresponding $b$-arcs, all hyperedges including vertex $x_i$ are consecutive in $Q$, which is therefore $x_i$-connected.

Since $H = GR(H)$, $Q$ is the minimal qual graph for $H$, therefore $H$ is an $X$-ring. 

**Remark.** Consider hypergraph $H = \{(1,2), (3,4), (2,3,5), (1,4,5)\}$. $H$ is Graham reduced, and $AG(H)$ is a circular-arc graph. However, $H$ is not an $X$-ring. Therefore, the fact that $AG(H)$ is a circular arc graph does not imply that $H$ is an $X$-ring.

**Theorem 3.1.** Given hypergraph $H$ with $n$ hyperedges over $m$ vertices, deciding whether $H$ is an $X$-ring can be done in time $O((n+m)^3)$.

**Proof.** We assume that $H$ is given as a list of hyperedges; each hyperedge is a list of vertices. Checking that $H = GR(H) \neq (\phi)$ can be done in time $O(nm \log m + n^2 m)$. By Lemmas 3.1 and 3.2, given a hypergraph $H$, we can decide whether $H$ is an $X$-ring by checking if $AG(H_u)$ is a circular-arc graph. $AG(H_u)$ can be constructed in time $O(n m^2)$ (using an adjacency matrix representation). The resulting graph has $n+m$ vertices, and no more than $m^2 + nm$ edges. An algorithm for recognizing
circular-arc graphs appears in [13]. This algorithm uses as input a matrix $M'$, which can be obtained from the adjacency matrix representing $AG(H_u)$ in $O((n + m)^5)$ time. The complete algorithm also takes $O((n + m)^3)$ time. Therefore, deciding whether hypergraph $H$ is an Xring can be done in $O((n + m)^3)$ time.

**Remark.** $AG(H_u)$ is not always a proper circular arc graph. Thus, a simpler algorithm recognizing proper circular arc graphs [12] cannot be used.
4. PROPERTIES OF TREE PROJECTIONS OVER XRINGS

This section contains several useful properties of tree projections w.r.t. Xrings. Let H be an Xring; fix a cycle qual graph Q for H and index its hyperedges accordingly. Let H' be a TP w.r.t. H, T a rooted qual tree for H' and T1 a subtree of T rooted at hyperedge e. A sequence G of overlapping base hyperedges e_1,e_2,...,e_j appearing in T1 is a maximal sequence of T1 if neither e_{j-1} nor e_j appears in T1. Hyperedge e is the ancestor of the maximal sequence G; e is simple (in T) if it is the ancestor of exactly one maximal sequence, otherwise it is composed. Note that a hyperedge e composed in some qual tree T cannot be a leaf in T; neither can it be the root of T since in that case it is the ancestor of exactly one maximal sequence containing all base hyperedges.

Example 4.1. Consider Xring H=\{1,2\}U\{3,4\}U\{5,6\}U\{7,8\}U\{9,10\} and hypergraph H''=H\cup\{1,2,3,5,6,7\}U\{3,4,5,7,8\}. H'' is a TP w.r.t. H, and in any qual tree for H'', there is a composed hyperedge. Figure 2 shows the only (up to isomorphism) qual tree for H''. Hyperedge \{1,3,4,5,7,8\} is composed since it is the ancestor of the two maximal sequences G1=\{3,4\},\{4,5\} and G2=\{7,8\},\{8,1\}.

Let H'' be a TP w.r.t. Xring H and T a qual tree for H''. We now describe a transformation applied to H'' and T resulting in a desirable structure.

Procedure Transform
1. H''_1 := H''; T_1 := T;
2. For each base hyperedge e internal in T_1 add a copy of e to H''_1 and attach it (via a new edge) to internal node e in T_1.
3. Repeatedly do the following until there are no more changes.
   3.1. If there exists a leaf e in T_1 which is a non-base hyperedge, then remove e from both H''_1 and T_1.
   3.2. If there exists a base hyperedge e which appears more than once as a leaf in T_1, then remove e from H''_1 and T_1.
4. Root T_1 at hyperedge e_root, the only hyperedge adjacent to base hyperedge e in T_1.
Let \( \mathbf{H}'_i \) and \( \mathbf{T}_i \) be the resulting hypergraph and qual tree. Following step 2 in procedure \textit{Transform}, all base hyperedges appear as leaves in the qual tree. Following step 3, \( \mathbf{T}_i \) satisfies (*) if each base hyperedge appears exactly once as a leaf in \( \mathbf{T}_i \); and all leaves are base hyperedges. Note that \( \mathbf{H}'_i \) is a projection of \( \mathbf{H}' \).

Clearly every tree projection may be modified by \textit{Transform} to satisfy property (*). In the remainder of this section, all qual trees satisfy property (*). In addition, we only consider leaves of qual trees as genuine base hyperedges, and regard any internal node, even one having the same vertices as those in some base hyperedge, as non-base.

**Lemma 4.1.** Let \( \mathbf{H}' \) be a TP w.r.t. Xring \( \mathbf{H} \), and \( \mathbf{T} \) a rooted qual tree for \( \mathbf{H}' \) satisfying property (*). Let the number of hyperedges composed in \( \mathbf{T} \) be \( w \). Then \( \mathbf{H}' \) and \( \mathbf{T} \) can be modified to obtain a TP \( \mathbf{H}'_s \leq \mathbf{H}' \), together with qual tree \( \mathbf{T}_s \) satisfying property (*), such that the number of hyperedges composed in \( \mathbf{T}_s \) is \( w-1 \).

**Proof.** We show how to obtain \( \mathbf{H}'_s \) and \( \mathbf{T}_s \) from \( \mathbf{H}' \) and \( \mathbf{T} \). Choose a hyperedge \( \mathbf{e}_s \) with parent \( \mathbf{e}_p \), such that \( \mathbf{e}_s \) is composed in \( \mathbf{T} \) and all of \( \mathbf{e}_s \)'s descendants are simple. Let \( \mathbf{T}_l \) be the subtree of \( \mathbf{T} \) rooted at \( \mathbf{e}_s \). \( \mathbf{H}'_s \) and \( \mathbf{T}_s \) are constructed using the following procedure.

**Procedure Simplify**

\[
\begin{align*}
\mathbf{H}'_s & := \mathbf{H} \\
\mathbf{T}_s & := \mathbf{T}
\end{align*}
\]

repeat

\[
\text{Decrease by one the number of maximal sequences contained in } \mathbf{T}_l
\]

by modifying \( \mathbf{H}'_s \) and \( \mathbf{T}_s \);

until \( \mathbf{T}_l \) contains exactly one maximal sequence.

The modification needed to decrease the number of maximal sequences in \( \mathbf{T}_l \) by one is as follows. Let \( G_1 = \mathbf{e}_i, \ldots, \mathbf{e}_{j-1} \) be one of the maximal sequences appearing in \( \mathbf{T}_l \). Let \( \mathbf{Y} = \bigcup_{\mathbf{e}_q \in \mathbf{e}_i, \ldots, \mathbf{e}_{j-1}} \mathbf{e}_q \), and \( \mathbf{Z} = \bigcup_{\mathbf{e}_q \in \mathbf{H} \setminus \mathbf{e}_i, \ldots, \mathbf{e}_{j-1}} \mathbf{e}_q \). The base hyperedges comprising \( G_1 \) are descendants (not necessarily proper) of some set \( \mathbf{C}_1 \) of children of \( \mathbf{e}_s \).

By choice of \( \mathbf{e}_s \) all hyperedges in \( \mathbf{C}_1 \) are simple; it follows that the base hyperedges...
which are descendants of hyperedges in $C_1$ are exactly the base hyperedges comprising $G_1$. The base hyperedges of $G_1$ are now removed from $T_1$ by creating a new subtree of $e_p$, which includes these hyperedges. This modification is done as follows:

(i) A new hyperedge $e_{\text{new}} = e_s \cap (\cup e) \cap Y$ is created and added to $H_s'$. (Observe that $H_s' \leq H'$. ) It is made a child of $e_p$ in $T_s$.

(ii) All hyperedges in $C_1$ together with their subtrees are detached from $e_s$ and attached to $e_{\text{new}}$, via new edges. Then, all hyperedges in the subtree rooted at $e_{\text{new}}$ are intersected with $Y$.

(iii) All remaining hyperedges in the subtree rooted at $e_s$ are intersected with $Z$.

Step (i) preserves vertex connectivity because $e_p \supset (e_{i-1} \cap e_i) \cup (e_{j-1} \cap e_j)$. Steps (ii) and (iii) ensure that there remain no superfluous connections between the newly created subtree (rooted at $e_{\text{new}}$), and the reduced subtree (rooted at $e_s$). Therefore, the above modification preserves vertex connectivity in $T_s$. The only base hyperedges contained in the new subtree whose root is $e_{\text{new}}$ are the base hyperedges in $G_1$, $e_{\text{new}}$ is therefore simple in $T_s$. Note that we do not regard $e_{\text{new}}$ as a base hyperedge, even if it contains the same vertices as some base hyperedge. Also, property (*) is preserved, since no leaves are added or deleted. All hyperedges that correspond to children of $e_s$ in $T$ clearly remain simple in $T_s$ - none of their descendants are affected by the above modification. For all other hyperedges, except $e_s$, the set of base hyperedge descendants remains unchanged, thus their status - simple or composed - remains as before the modification.

By the preceding discussion, after applying $\text{Simplify}$ the number of hyperedges composed in $T_s$ is one less than the number of hyperedges composed in $T$. Since $H_s' \leq H'$, $H_s'$ and $T_s$ obtained by procedure $\text{Simplify}$ have the properties claimed by the lemma. 

**Lemma 4.2.** Let $H$ be an $X$ring. If there exists a TP $H''$ w.r.t. $H$ together with a rooted qual tree $T$ satisfying property (*), then there exists a TP $H''_s$ w.r.t. $H$.
together with a rooted qual tree $T_z$ satisfying property (*), such that every hyperedge of $H'_z$ is simple in $T_z$.

**Proof.** By induction on $n$, the number of hyperedges composed in $T$.

**Basis.** ($n = 0$). Then $H''_z = H'$, and the lemma holds trivially.

**Induction.** ($n > 0$). Assume that $H'$ and $T$ can be modified to obtain $H''_z$ and $T_z$ as claimed, if the number of hyperedges composed in $T$ is less than $n$. Consider hypergraph $H''$ with rooted qual tree $T$ satisfying property (*), such that there are $n$ hyperedges composed in $T$. By Lemma 4.1, $H''$ and $T$ can be modified to obtain a hypergraph $H'_z$, together with rooted qual tree $T_z$ satisfying property (*), with $n-1$ composed hyperedges. By the induction hypothesis, $H'_z$ and $T_z$ may be further modified to obtain $H''_z$ and $T_z$ as claimed. ■
5. AN ALGORITHM FOR DECIDING TREE PROJECTION EXISTENCE

Tree Projection Existence over Xrings (TPX): Let \( H \) be an Xring of size \( n \), over \( m \) different vertices. Given a hypergraph \( H' \), is there a TP \( H'' \) of \( H' \) w.r.t. \( H \).

An algorithm is presented to decide TPX in time polynomial in \( n, m \) and the length of \( H \) and \( H' \). A detailed description of the algorithm is given in the Appendix. Consider Xring \( H \) and hypergraph \( H' \), and suppose a tree projection of \( H' \) w.r.t. \( H \) need be constructed. The algorithm attempts to build tree projections w.r.t. groups of overlapping base hyperedges, using subsets of hyperedges in \( H' \).

Stretch \([i,j]\) can be spanned by \( H' \) if it is possible to construct a TP w.r.t. overlapping base hyperedges \( e_i, e_{i+1}, ..., e_{j-1} \) which has a qual tree with hyperedge \( e = (e_{i-1}\cap e_i)\cup(e_{j-1}\cap e_j) \) as its root, using subsets of hyperedges in \( H' \). The length of stretch \([i,j]\) is \( j-i \); base hyperedge \( e_i \) is defined as stretch \([i,i+1]\) of length 1.

To prove that hypergraph \( H' \) has a tree projection w.r.t. Xring \( H \), it suffices to show that stretch \([1,n]\) can be spanned by \( H' \). In this case there exists a TP w.r.t. base hyperedges \( e_1, ..., e_{n-1} \) whose qual tree has hyperedge \((e_{n-1}\cap e_n)\cup(e_n\cap e_1) \equiv e_n \) at its root. This is a TP of \( H' \) w.r.t. \( H \).

Algorithm Check is presented below. When given Xring \( H \) and hypergraph \( H' \), the algorithm repeatedly produces stretches, i.e. finds stretches that can be spanned by \( H' \) and adds them to the following data structures. Set \( currentS \) accumulates all produced stretches. Newly produced stretches are taken as building blocks for additional stretches. Queue \( Sq \) holds all stretches produced but not yet considered as candidates for further extension. Sets \( end_at \) and \( start_at \) facilitate efficient extension of stretches. Suppose stretches \([i,j]\) and \([j,k]\) can be spanned by hypergraph \( H' \). Stretch \([i,k]\) can be spanned if there exists some hyperedge \( e \in H' \) such that \( e \supset (e_{i-1}\cap e_i)\cup(e_{j-1}\cap e_j)\cup(e_{k-1}\cap e_k) \). When attempting to combine stretches \([i,j]\) and \([j,k]\), the algorithm checks whether there exists such a hyperedge. If so, stretch \([i,k]\) is produced. The algorithm terminates when no new stretches can be produced. If stretch \([1,n]\) is produced, the algorithm returns \text{true}, otherwise \text{false} is returned.
function Check(H, H': hypergraph; n, m: integer): boolean;
  /* n is size of Xring, m number of vertices */

  var currentS: set_of_stretches; /* all stretches produced so far */
  Sq: queue_of_stretches; /* holds all stretches not yet checked for extension */
  end_at, start_at: array[1..n] of set of 1..n;
  /* end_at[i][j] contains those j for which \exists\ stretch [j,i] \in currentS */
  /* start_at[i][j] contains those j for which \exists\ stretch [i,j] \in currentS */

procedure ExtendS(str:stretch;
  var Sq: queue_of_stretches; var currentS: set_of_stretches);
  /* try to produce additional stretches using a new stretch */

begin
  let str = [i,j];
  for all k \in end_at[i] do /* try to extend stretch counterclockwise */
    if [k,j] \notin currentS then /* this is potentially a new stretch */
      if \exists e \in H' such that ( (e_{k-1} \cap e_k) \cup (e_{i-1} \cap e_i) \cup (e_{j-1} \cap e_j) ) \subseteq e
        then /* extension possible */
          begin /* update data structures */
            update end_at[i] and start_at[k];
            Sq:=addq(Sq,[k,j]); /* record a new stretch */
            currentS := currentS \cup \{[k,j]\}; /* remember this stretch */
          end;
    for all k \in start_at[i] do /* try to extend stretch clockwise */
      similar to extension counterclockwise;
  end;

begin /* main */
  /* initialize */
  currentS := \emptyset;
  Sq:=an_empty_queue;
  produce all stretches [i,i+1] of length 1, except [n,1], such that \exists e \in H': e \subseteq e;
  Appropriately initialize arrays end_at and start_at;
  repeat
    str:=frontq(Sq); /* consider a new stretch and remove it from Sq */
    ExtendS(str,Sq,currentS) /* try to extend */
    until emptyq(Sq);
  Check:=(1..n) \subseteq currentS
end.

Remark. By recording for each new stretch the two stretches which produced it and the subsets of edges in H' used, a TP may easily be obtained (if one exists). •

Lemma 5.1. Let H be an Xring (of size n over m vertices) and H' a hypergraph. Assume that on input H, H', n and m algorithm Check produces, in some order, b \geq 1 disjoint stretches [k_0,k_1], [k_1,k_2], ..., [k_{i-1},k_i], and that there exists a hyperedge e \in H' such that e \supseteq \bigcup_{m=0}^{i} (e_{k_{m-1}} \cap e_{k_m}). Then stretch [k_0,k_1] is produced by Check on input H, H', n and m.
Proof. We prove a stronger result, namely that all stretches \([k_p, k_r]\), for \(0 \leq p < r \leq l\), are produced by Check on input \(H, H', n\) and \(m\). The proof is by induction on \(r-p\).

Basis. \(r-p = 1\). Then the stretch is \([k_p, k_r] = [k_p, k_{p+1}]\), which, by the statement of the lemma, is produced.

Induction. Assume that the lemma holds for \(r-p < h\), i.e. assume that for \(r-p < h\) all stretches \([k_p, k_r]\) are produced. Consider stretch \([k_p, k_r]\) with \(r-p = h\). By the induction hypothesis, stretches \([k_p, k_{r-1}]\) and \([k_{r-1}, k_r]\) are produced. One of them, w.l.o.g. \([k_p, k_{r-1}]\), appears later at the front of \(S^q\). Since each stretch is checked for all possible extensions, and since e\(\in\)\((e_{p-1} \cap e_p) \cup (e_{r-1} \cap e_{r-2}) \cup (e_{r-2} \cap e_{r-1})\), when stretch \([k_p, k_{r-1}]\) appears at the front of \(S^q\) it is combined with the already produced stretch \([k_{r-1}, k_r]\) to produce stretch \([k_p, k_r]\). This concludes the induction step and proves the lemma.

Lemma 5.2. Let \(H\) be an Xring of size \(n\) over \(m\) vertices, and \(H'\) a hypergraph. If there exists a TP of \(H'\) w.r.t. \(H\), then algorithm Check returns \textbf{true} on input \(H, H', n\) and \(m\).

Proof. Let \(H''\) be a TP of \(H'\) w.r.t. \(H\), and let \(T\) be a qual tree for \(H''\). Using procedure \textit{Transform}, obtain a TP \(H'_z \leq H''\), together with qual tree \(T_z\) rooted at \(e_{\text{root}}\) and with base hyperedge \(e_n\) a child of \(e_{\text{root}}\), satisfying property (*) . By the process outlined in Lemma 4.2, obtain \(H'_z \leq H'' \leq H''\) together with qual tree \(T_z\) such that every hyperedge of \(H'_z\) is simple in \(T_z\). Observe that \(e_{\text{root}}\) remains the root of \(T_z\), and that \(e_n\) remains a child of \(e_{\text{root}}\). Reroot now \(T_z\) at base hyperedge \(e_n\). Hyperedge \(e_{\text{root}}\) becomes the only child of \(e_n\). The rest of the tree is unaffected. Clearly \(e_{\text{root}}\) is simple in the new tree \(T\), since all base hyperedges except \(e_n\) are descendants of it. All other hyperedges remain simple.

We show that all hyperedges of \(H'_z\) (except \(e_n\)) correspond to stretches produced during the execution of algorithm Check. We consider the hyperedges of \(H'_z\) ordered by the postorder (i.e. reverse breadth-first order) traversal of \(T\).
The leaves of $T$ are the base hyperegeses except $e_n$. Since a TP exists by the statement of the lemma, $H \subseteq H'$ and for $1 \leq i \leq n-1$, all stretches $[i, i+1]$ of length 1 are produced by Check. For all other hyperegeses, we prove the following.

Claim: If hyperege $e_s$ is the ancestor of a maximal sequence $G = e_1, \ldots, e_{j-1}$, then stretch $[i, j]$ is produced during the execution of algorithm Check.

Proof of claim. By induction on the order in which non-leaf hyperegeses are visited during the postorder traversal of $T$.

Basis. The first non-leaf hyperege, $e_s$, is visited; all children of $e_s$ are leaves. These leaves are overlapping base hyperegeses $e_i, \ldots, e_{j-1}$. By vertex connectivity, $e_s \cup \bigcup_{k=1}^{j-1} e_k$; therefore, for $i \leq k \leq j$, $e_s \supseteq e_{k-1} \cap e_k$. Since a tree projection exists, $H \subseteq H'$ and thus stretches $[i, i+1], [i+1, i+2], \ldots, [j-1, j]$ are produced. It follows from Lemma 5.1 that stretch $[i, j]$ is produced.

Induction. Assume the claim holds for all nodes visited so far during the traversal. Next, visit node $e_s$ and consider its children $e_1, e_2, \ldots, e_l$. These children of $e_s$ are ancestors of $l$ maximal sequences $G_1, G_2, \ldots, G_l$, and by the induction hypothesis the corresponding stretches are produced. The $l$ maximal sequences are disjoint, since by property (*) no base hyperege appears as leaf more than once. Since $e_s$ is the ancestor of exactly one maximal sequence $G$, $G_1, G_2, \ldots, G_l$ taken together must form $G$. The corresponding stretches are $[i = k_0, k_1]$, $[k_1, k_2]$, $\ldots$, $[k_{l-1}, k_l = j]$, which by the induction hypothesis are all produced, in some order, during the execution of Check. By vertex connectivity of $T$, $e_s \supseteq e_{m-1} \cap e_m$ for $0 \leq m \leq l$; it follows from Lemma 5.1 that stretch $[k_0, k_l] = [i, j]$ is produced. This proves the induction step and the claim.

At the end of the postorder traversal, $e_{\text{root}}$, the sole child of $e_n$ is visited; it is the ancestor of maximal sequence $G = e_1, \ldots, e_{n-1}$. Thus, by the claim, stretch $[1, n]$ is produced and when the algorithm terminates it returns True.

Lemma 5.3. If on input $H, H', n$ and $m$ algorithm Check produces stretch $[i, j]$, then stretch $[i, j]$ can be spanned by $H'$.
Proof. By induction on the length of a stretch added to current $S$.

Basis. A stretch $[i,j] = [i, i+1]$ of length 1 is produced only if it can be spanned, i.e. if there exists a hyperedge $e \in H'$ such that $e \subset e$.

Induction. Assume the lemma holds for all stretches of length $< h$. Now suppose stretch $[i,j]$ of length $h$ is produced. This is possible only if there is some $k$ ($i < k < j$) such that $[i,k] \in currentS$, $[k,j] \in currentS$, and for some $e_k \in H'$, $e = (e_{k-1} \cap e_k) \cup (e_k \cap e_{j-1}) \cup (e_{j-1} \cap e_j) \subset e$. Clearly, stretches $[i,k]$ and $[k,j]$ are both of length $< h$. So, by the induction hypothesis, they can be spanned by $H'$ and there exists a TP w.r.t. base hyperedges $e_i, ..., e_{k-1}$ with a hyperedge $e_k = (e_{k-1} \cap e_k) \cup (e_k \cap e_{j-1})$ as its root, and a TP w.r.t. base hyperedges $e_k, ..., e_{j-1}$ with a hyperedge $e_{kj} = (e_{k-1} \cap e_k) \cup (e_{j-1} \cap e_j)$ as its root. Observe that $e_k \cup e_{kj} = e$. Let $e' = (e_{k-1} \cap e_k) \cup (e_{j-1} \cap e_j)$. The tree illustrated in Figure 3 is a TP w.r.t. base hyperedges $e_i, e_{i+1}, ..., e_{j-1}$. Note that there is no need for vertices $e \rightarrow e'$ to appear in the root hyperedge. ($x \in e$ and $x \not\in e'$ implies that vertex $x$ does not appear in any hyperedge $e_k$, with $k < i$ or $k > j$.)

Theorem 5.1. Algorithm Check returns true on input $H, H', n$ and $m$ iff there exists a TP from $H'$ w.r.t. $H$.

Proof. ($\rightarrow$) This is Lemma 5.2.

($\leftarrow$) If Check returns true, then stretch $[1,n]$ is produced. By Lemma 5.3 stretch $[1,n]$ can be spanned, so there exists a TP w.r.t. base hyperedges $e_i, ..., e_{n-1}$, built with hyperedges that are subsets of hyperedges in $H'$. The qual tree for that tree projection has as its root a hyperedge $(e_{n-1} \cap e_i) \cup (e_{n-1} \cap e_{n})$. Since $H$ is Graham reduced, this is exactly base hyperedge $e_n$, so we have a TP w.r.t. all the hyperedges in $H$.

Let $w$ be the number of hyperedges in $H'$. Assume that $H'$ is given as a list of hyperedges with the vertices in each hyperedge listed in sorted order. By choosing appropriate data structures we can show the following.

Theorem 5.2 (see Appendix). Algorithm Check terminates in time $O(|H| + |H'| + nmw + n^2w)$. ■
6. XRINGS AND GRAPH REPRESENTABLE HYPERGRAPHS

In this section we present results and counterexamples that clarify the relationships between different types of hypergraphs. A graphic representation is given in Figure 4.

A family of hypergraphs called circular representable hypergraphs has recently been introduced [10]. A hypergraph $H$ is circular representable if its vertices can be placed around a circle, so that the vertices appearing in any hyperedge $e \in H$ are consecutive on the circle [10]. Such an ordering is called a circular representation of $H$. A hypergraph $H$ is interval representable if its vertices can be placed along a straight line so that the vertices appearing in any hyperedge $e \in H$ are consecutive on the line. (See [3], where these hypergraphs are called interval hypergraphs.) Obviously, the interval representable hypergraphs are a (proper) subset of the circular representable hypergraphs.

Lemma 6.1. If $H$ is an interval representable hypergraph, then $H$ is a tree hypergraph.

Proof. We show that $GR(H) = (\emptyset)$. Consider an interval representation of $H$, and renumber the vertices in ascending order according to their position on the line. Repeatedly consider vertices $x_i, x_{i+1}, \ldots$, starting with $x_1$. There are two possibilities.

- $x_i$ appears exactly in one hyperedge $e_r$. Then $x_i$ is an isolated vertex. Eliminate $x_i$ from $H$.
- $x_i$ appears in several hyperedges, w.l.o.g. $e_1, e_2, \ldots, e_j$. Inductively, no vertex having index $< i$ is included in any of these hyperedges. Since all hyperedges are sets of vertices consecutive on the line, one hyperedge, w.l.o.g. say $e_1$, is a superset of all the others. Eliminate hyperedges $e_2, \ldots, e_j$ from $H$. Now vertex $x_i$ appears only in hyperedge $e_1$, and is thus eliminated.

For each vertex considered, $H$ is reduced. Eventually a hypergraph having only one vertex is obtained, which is then reduced to the hypergraph $(\emptyset)$. $H$ is therefore a tree hypergraph. \qed

Technion - Computer Science Department - Technical Report CS0385 - 1985
The converse of Lemma 6.1 does not hold, viz. hypergraph $H_1=\{(1,2),\{2,3\},\{1,3\},\{1,2,3\}\}$ which is acyclic, but not interval representable. However, this hypergraph is circular representable. As another example consider hypergraph $H_2=\{(1,2),\{3,4\},\{5,6\},\{1,3,5\}\}$, which is acyclic but not circular representable.

The following examples clarify the relationships between circular-representable hypergraphs and Xrings. Consider Xring $H_3=\{(1,2,3),\{1,2,8\},\{1,3,4\},\{1,5,6\},\{4,5\}\}$ (see Figure 5). Since vertices 1 and 2 both appear in hyperedges $\{1,2,3\}$ and $\{1,2,6\}$, in any circular representation of $H_3$ vertices 1 and 2 must be adjacent on the circle. Since vertex 1 also appears in hyperedge $\{1,3,4\}$, the second vertex adjacent to 1 must be 3. But now we cannot account for hyperedge $\{1,5,6\}$, as neither 5 nor 6 can be adjacent to vertex 1 on the circle. Therefore Xring $H_3$ is not a circular representable hypergraph. Consider hypergraph $H_4=\{(1,2),\{1,4\},\{2,3\},\{1,3,4\}\}$. $H_4$ is clearly circular representable. (Place the vertices around a circle in ascending order. Note that vertex 4 is adjacent to vertex 1.) However, although cyclic, $H_4$ is not an Xring. (It is not Graham reduced.) Finally, consider Xring $H_5=\{(1,2,3),\{2,3,4\},\{3,4,5\},\{4,5,1\},\{5,1,2\}\}$, which is circular representable. Therefore Xrings and circular representable hypergraphs constitute distinct, yet non-disjoint, hypergraph classes.

**Lemma 6.2.** If $H$ is a circular representable hypergraph satisfying $H=\text{GR}(H)\neq (\phi)$, then $H$ is an Xring.

**Proof.** Consider a circular representation for $H$. Since all vertices included in a hyperedge are consecutive in the circular representation of $H$, each hyperedge can be visualized as an arc going around the circle including exactly the vertices of the hyperedge\(^6\). Since $H$ is Graham reduced, no arc is contained in another. Therefore, when going around the circle, for every vertex $z_i$ encountered, exactly one or zero arcs start at vertex $z_i$. In proof observe that if several arcs start at vertex $z_i$, then a hyperedge represented by one of these arcs must be a superset

---

\(^6\) Note that the arcs here correspond to hyperedges, whereas the arcs in Lemmas 3.1 and 3.2 correspond to vertices. Recall that traversal of the circle is by convention clockwise.
of the hyperedges represented by the other arcs, a contradiction to $H = GR(H)$.

The hyperedges of $H$ can therefore be uniquely indexed in the order determined by the startpoints of their corresponding arcs. Consider a cycle $Q$ whose vertices are the hyperedges of $H$ arranged clockwise on the cycle and renumbered according to the indexing produced above.

Claim: $Q$ is X-connected.

Proof of claim. Consider any vertex $x_q$. Let $e_i$ and $e_k$, $i < k$, be any two hyperedges which include vertex $x_q$. By the way the hyperedges were renumbered, hyperedge $e_i$ appears before hyperedge $e_k$ in $Q$ (going clockwise). Since $x_q \in e_i$, this hyperedge includes all vertices on the circle between its first (i.e. most counterclockwise) vertex and $x_q$. Also, $x_q$ appears somewhere after (i.e. more clockwise relative to) the startpoint of $e_k$. Suppose there exists a hyperedge $e_j$, $i < j < k$, such $x_q \notin e_j$. By the above indexing the first vertex in hyperedge $e_i$ appears before the first vertex of hyperedge $e_j$ in the circular representation, and the first vertex of $e_j$ appears before the first vertex of $e_k$. However, since $x_q \notin e_j$, the last vertex of $e_j$ must appear before $x_q$ on the circle. It follows that $e_j \subset e_i$, a contradiction to the fact that $H$ is Graham reduced.

We have thus shown that $H$ is a Graham reduced hypergraph with a cycle qual graph, i.e. it is an Xring.

Example 6.1. Consider hypergraph $H = \{(1,2,3),(2,3,4),(4,5),(5,6),(6,1,2)\}$, which is both circular representable and Graham reduced. A circular representation is obtained by placing the vertices in their natural order around a circle. The arcs corresponding to the five hyperedges of $H$, (as in the proof of Lemma 6.2,) start with vertices 1, 2, 4, 5 and 6. Figure 6 shows a qual graph for $H$, where the hyperedges appear in the same order as their corresponding arcs.

Consider an Xring $H$ of size $n$ over $m$ vertices. $H$ is a uniform Xring if $m = n$.

Lemma 6.3. If $H$ is a uniform Xring then all hyperedges in $H$ have the same cardinality.
Proof. Let $Q$ be a cycle qual graph for $H$. By Lemma 2.1, there exist $n$ distinct vertices $z_{i+1} \in e_{i+1}-e_i$ $(1 \leq i \leq n)$. Suppose that not all hyperedges have the same cardinality, i.e. for some $i$, $|e_i| \neq |e_{i+1}|$. W.l.o.g. $|e_i| < |e_{i+1}|$. Thus $e_{i+1}$ includes at least two "new" vertices $z_i \in e_{i+1}-e_i$ and $y \in e_{i+1}-e_i$. We claim that $y$ does not belong to $\{z_1, \ldots, z_n\}$.

In proof, suppose there is an index $j$ such that $y \in e_{j+1}-e_j$. However, $y \in e_{i+1}-e_i$, which implies that $Q$ is not $y$-connected. Therefore $y$ is not one of $\{z_1, \ldots, z_n\}$. It follows that $H$ must have at least $n+1$ vertices, a contradiction. 

Example 6.2. Hypergraph $H_6=\{1,2,3,4\}, \{2,3,4,5\}, \{3,4,5,6\}, \{4,5,6,7\}, \{4,6,7,8\}, \{1,6,7,8\}, \{1,2,7,8\}, \{1,2,3,8\}$ is a uniform Xring. 

The above proof implies that in going around $Q$, for each hyperedge encountered there is exactly one dropped vertex ($|e_i-e_{i+1}|=1$) and one added vertex ($|e_{i+1}-e_i|=1$). If the vertex dropped is the "oldest", i.e. that which appears in the most hyperedges going counterclockwise, then $H$ is super-uniform ($s$-uniform). Clearly, if $H$ is $s$-uniform, then all vertices in $H$ appear in an equal number of hyperedges. We can therefore characterize a $s$-uniform Xring by its size $n$, and by the number of hyperedges $s$ in which each vertex appears (which is also the size of each hyperedge). Arings are $s$-uniform Xrings with $s=2$. Acliques [6] are $s$-uniform Xrings with $s=n-1$.

Example 6.3. Hypergraph $H_7=\{1,2,3,4\}, \{2,3,4,5\}, \{3,4,5,6\}, \{1,4,5,6\}, \{1,2,5,6\}, \{1,2,3,6\}$ is a $s$-uniform Xring with $n=6$ and $s=4$. Note that hypergraph $H_6$ of Example 6.2 is not $s$-uniform. 

Lemma 6.4. If $H$ is a $s$-uniform Xring then $H$ is circular representable.

Proof. Let $z_i$ be $e_i-e_{i-1}$. Place the vertices according to this order around a circle. By definition of $s$-uniformity, this is a circular representation for $H$. 

A hypergraph is semi-Graham reduced ($s$GR) if it has no hyperedges which are subsets (or duplicates) of other hyperedges. Denote by $s$GR($H$) the hypergraph obtained from $H$ by removing all hyperedges that are subsets or duplicates of other hyperedges (i.e. applying procedure GR without step 1).
Lemma 6.5. Let \( H \) be a tree hypergraph which is circular representable but not interval representable. Then \( H_{GR} = sGR(H) \) is interval representable.

**Proof.** The case of interest is \( H_{GR} \neq (\emptyset) \). Since \( H \) is acyclic, \( H_{GR} \) is also acyclic. \( H_{GR} \) is sGR, so the next step in the Graham reduction must be the deletion of isolated vertices, and then (if non-empty hyperedges remain) the deletion of some hyperedge \( e_r \), which becomes a subset of another hyperedge \( e_s \). If no non-empty hyperedges remain following isolated vertex elimination, \( H_{GR} \) is clearly interval representable since every vertex appears in exactly one hyperedge. Otherwise, consider an hyperedge \( e_r \) that can be eliminated next. Since \( H \) is sGR, \( e_r \) originally includes at least one isolated vertex \( x \in e_r \in H_{GR} \). Consider a circular representation of \( H_{GR} \) (one exists since \( H \) is circular representable).

**Claim:** If \( e_r \) has several isolated vertices, they must all be adjacent in the circular representation.

**Proof of claim.** Suppose there exists a non-isolated vertex \( y \) on the circle between two isolated vertices \( x_1 \in e_r \) and \( x_2 \in e_r \). (We assume, of course, that \( x_1 \) and \( x_2 \) are not adjacent on the circle.) Then \( y \in e_r \), and there exists a hyperedge \( e_s \neq e_r \) such that \( y \in e_s \), \( x_1 \notin e_s \), and \( x_2 \notin e_s \). Clearly \( e_s \subseteq e_r \), and \( H_{GR} \) is not sGR.

By the discussion above, \( e_r \) can be partitioned into two disjoint subsets \( e_I \) and \( e_C \), such that \( e_I \) includes all isolated vertices, and \( e_C \) is a subset of another hyperedge \( e_s \in H_{GR} \), which may induce the elimination of \( e_r \) during \( GR \). By the claim, \( e_s \) must be entirely on one side of \( e_r \) in the circular representation (see Figure 7). W.l.o.g. we assume that \( e_s \) appears more clockwise than \( e_r \). An interval representation of \( H_{GR} \) may therefore be obtained by "disconnecting" the circle at the point separating \( e_r \) from its counterclockwise neighbor in the representation.
7. CONCLUSIONS

A new family of cyclic hypergraphs, called Xrings, is introduced. A characterization of Xrings using circular-arc graphs is presented, yielding a polynomial time algorithm for recognizing Xrings. Some additional properties of Xrings are investigated, in particular the relationship between Xrings, interval representable hypergraphs and circular representable hypergraphs.

A polynomial time algorithm is presented for discovering a tree projection, embedded in an arbitrary hypergraph, with respect to an Xring. Such an algorithm may be useful in query processors for relational database systems. Determining the existence of a tree projection with respect to a hypergraph in the general case is in \(NP\), and is conjectured to be \(NP\)-complete.
REFERENCES


APPENDIX

Algorithm Check is presented below. It is assumed that the edges in $H$ are numbered according to their appearance in some cycle qual graph for $H$. Set $currentS$ is implemented as an array of size $n^2$. (Only entries above the main diagonal are used.) Testing set membership can therefore be done in constant time. All other sets are implemented as ordered linked lists, to facilitate computing set intersection in time linear in the size of the participating sets. Three data structures not mentioned in the body of the paper are arrays $have$, $good$ and $intersect$. Arrays $intersect$ and $good$ are used when combining two stretches. For $1 \leq i \leq n$, $intersect[i]$ holds $e_{i-1} \cap e_i; 0 \equiv n$. For $1 \leq i \leq n$, $good[i]$ holds the indices of all hyperedges which are a superset of $e_{i-1} \cap e_i$. Array $have$ is used during initialization to compute the content of array $good$; $i \in have[i]$ iff hyperedge $e_i \in H'$ includes vertex $i$. 
function Check(H,H' :hypergraph; n,m: integer):boolean;
/* n is size of XriI1g, m number of vertices, w number of hyperedges in H' */
type base = 1..n;
   vertex = 1..m;
   stretch = array[1..2] of base;
   set_of_stretches = array[base,base] of boolean;
   queue_of_stretches = queue of stretch;
   set_of_vertices = set of vertex;
   hyp_indices = set of 1..w; /* indices of hyperedges in H' */

var currentS : set_of_stretches; /* all stretches produced so far */
   Sq : queue_of_stretches; /* holds all stretches not yet checked for extension */
   end_at, start_at : array[1..n] of set of base;
   /* end_at[i] contains those j for which \exists \text{ stretch }[j,i] \in \text{ currentS} */
   /* start_at[i] contains those j for which \exists \text{ stretch }[i,j] \in \text{ currentS} */
   have : array[vertex] of hyp_indices;
   /* \( i \in \text{have}[l] \iff x_l \in X_{H'} \) */
   good : array[base] of hyp_indices;
   /* \( j \in \text{good}[i] \iff x_{i-1} \in X_i \cap e'_j \) where \( e_{i-1}, e_i \in X_H \) and \( e_j \in X_{H'} \) */
   intersect : array[base] of set_of_vertices;
   /* \( i \in \text{intersect}[i] \iff x_i \in X_{e_{i-1}} \cap e_i \) */
   str : stretch;
   i,j : base;
   l : vertex;

procedure ExtendS(str:stretch;
   var Sq:queue_of_stretches; var currentS:set_of_stretches);
/* try to create additional stretches from a new stretch */

var str1 : stretch;
   i,j,k : base;

begin
   i:=str[1];
   j:=str[2];
   for all k \in \text{end_at}[i] do /* try to extend counterclockwise */
      if not currentS[k,j] then /* has not been produced yet */
         if (good[k] \cap good[i] \cap good[j] \neq \emptyset) then /* extend */
            begin /* update data structures */
               end_at[j]:=end_at[j] \cup \{k\};
               start_at[k]:=start_at[k] \cup \{i\};
               str1[1]:=k;
               str1[2]:=j;
               Sq:=addq(Sq,str1); /* remember this is a new stretch */
               currentS[k,j]:=true
            end;
   for all k \in \text{start_at}[j] do /* try to extend clockwise */
      if not currentS[i,k] then /* has not been produced yet */
         if (good[i] \cap good[j] \cap good[k] \neq \emptyset) then /* extend */
            then /* extend */
            begin /* update data structures */
               end_at[k]:=end_at[k] \cup \{i\};
               start_at[i]:=start_at[i] \cup \{k\};
               str1[1]:=i;
               str1[2]:=k;
              Sq:=addq(Sq,str1); /* remember this is a new stretch */
               currentS[i,k]:=true
            end
end;
begin /* main */
  /* initialize have */
  for l:=1 to m do have[l]:=∅;
  for all hyperedges e_i ∈ H' do
    for all vertices x_i ∈ e_i do
      have[l]:=have[l] ∪ {i}; /* vertex x_i appears in hyperedge e_i */
  for i:=1 to n-1 do
    for j:=i+1 to n do currentS[i,j]:=false;
    Sq:=an_empty_queue;
    /* for each Xtring hyperedge except e_n */
    /* form a length one stretch and add it to data structures */
    for i:= 2 to n do
      begin
        str[1]:=i-1;
        str[2]:=i;
        Sq:=addq(Sq,str);
        end_at[i]:=i-1;
        start_at[i-1]:=i;
        intersect[i]:={e_{i-1} ∩ e_i}
      end;
      intersect[1]:=e_n ∩ e_1;
      for i:=1 to n do
        begin /* initialize good */
          good[i]:={1,2,...,w}; /* initialize for intersection */
          for all x_i ∈ intersect[i] do good[i]:=good[i] ∩ have[l]
        end;
        repeat
          str:=frontq(Sq); /* consider a new stretch */
          Sq:=removeq(Sq);
          ExtendS(str,Sq,currentS) /* try to extend */
        until emptyq(Sq) or currentS[1,n];
      Check:=currentS[1,n]
    end.
Remark. Stretches of length 1 are added to the data structures during initialization. Checking if $H \leq H'$ is redundant, since it is implicit in the production of stretches of length > 1. Clearly, if some stretch $[i,j]$, $j-i>1$, is produced by the algorithm, then for all $i \leq k \leq j-1$ there exists some hyperedge $e \in H'$ such that $e_{i-1} \cap e_k \subseteq e$ and $e_k \cap e_{k+1} \subseteq e$. Since $H$ is Graham reduced this implies that $e_k \subseteq e$. If the algorithm returns true, then stretch $[1,n]$ is produced. It follows that $H \leq H'$. 

Proof of Theorem 5.2. The number of stretches that can be produced is bounded by $n^2$. Procedure `ExtendS` considers at most $n$ points for (clockwise or counterclockwise) extension. For each such point, if there is a possibility of producing a new stretch, the intersection of three elements of array `good`, each a set of size at most $w$, is computed. Since these sets are implemented as ordered linked lists, these intersections may be computed in time $O(w)$. Therefore, all possible stretches are produced in time $O(n^3w)$. Initialization of `good[i]` is done by intersecting at most $m$ sets, each of size at most $w$. The whole array may therefore be initialized in time $O(nmw)$. Initialization of `have` takes $O(|H'|)$ time, and initialization of `intersect` is done in $O(|H|)$ time. Algorithm Check therefore terminates in time $O(|H|+|H'|+nmw+n^3w)$. 

Technion - Computer Science Department - Technical Report CS0385 - 1985
Figure 1

\{1,4,5\} -- \{4,5,6,8\}
\{1,2,3,4\} -- \{1,4,6,8\}
\{1,2,3,7\} -- \{1,2,4,6,7\}

Figure 2

\{1,2,3,5,6,7\}
\{1,2\} -- \{2,3\} -- \{5,6\} -- \{6,7\}
\{1,3,4,5,7,8\}
\{3,4\} -- \{4,5\} -- \{7,8\} -- \{8,1\}

Figure 3
cyclic hypergraphs

Figure 4

{1,2,3}

{1,2,6} {1,3,4}

{1,5,6} {4,5}

interval representable hypergraphs

Figure 5

{2,3,4}

{1,2,3} {4,5}

{8,1,2} {5,6}

Figure 6

acyclic hypergraphs

Xrings

circular representable hypergraphs

s-uniform Xrings

Figure 7

Technion - Computer Science Department - Technical Report CS0385 - 1985