EXPRESSIBILITY OF BOUNDED-ARITY FIXED-POINT QUERY HIERARCHIES

Pratul Dublish
Dept. of Computer Science & Engg.,
I.I.T. Kanpur, Kanpur, India - 208016

S.N. Maheshwari
Dept. of Computer Science & Engg.,
I.I.T. Delhi, New Delhi, India - 110016

ABSTRACT

The expressibility of bounded-arity query hierarchies resulting from the extension of first-order logic by the least fixed-point, inductive fixed-point and generalized fixed-point operators is studied. In each case, it is shown that increasing the arity of the predicate variable from k to k+1 always allows some more k-ary predicates to be expressed. Further, k-ary inductive fixed-points are shown to be more expressive than k-ary least fixed-points and k-ary generalized fixed-points are shown to be more expressive than k-ary inductive fixed-points.

1. Introduction

The failure of query languages based on first-order logic (FO) to express several queries of interest, like transitive closure, is well known [AU]. This has led to the development of query languages based on the extension of FO by several fixed-point operators like least fixed-point (LFP) [AU,CH,Im2], inductive fixed-point (IFP) [GS] and generalized fixed-point (GFP) [Im2]. Each fixed-point operator enables the computation of some fixed-point of FO formulae $g(P,x_1,\ldots,x_k)$ where P is a k-ary predicate variable and x_is are the free variables of **g**.

We study the expressibility of bounded-arity fixed-point query hierarchies obtained by restricting the arity of the predicate variable P in formulae **g**. The k^{th} level in each hierarchy, $k \geq 1$, is the query language resulting from restricting the arity of the predicate variable P to exactly k. Chandra and Harel [CH] were the first to express an interest in the study of such a hierarchy for FO+LFP. They had proved that certain k-ary predicates were not expressible in the k^{th} level of the FO+LFP hierarchy. It had been shown in [Im2] that these predicates were expressible in the $2k+3^{th}$ level of the FO+LFP hierarchy. Thus it appeared that the arity of the predicate variable had to be more than doubled before some more k-ary predicates could be expressed. They left it as an open problem to decide whether increasing the arity by one enables some more k-ary predicates to be expressed. We resolve this problem by showing the existence of k-ary predicates which are not expressible in the k^{th} level of the FO+LFP hierarchy but are expressible in its $k+1^{th}$ level. Similar results are also shown for the FO+IFP and FO+GFP hierarchies. These results are then extended for the case when a valid successor predicate on the database domain is available to the query language.

The k-ary hierarchies can also be used to obtain a better idea of the relative expressive powers of FO+LFP, FO+IFP and FO+GFP. Although it is known that FO+LFP and FO+IFP have the same expressive power [GS], we show the existence of predicates which are expressible in the k^{th} level of the FO+IFP hierarchy but are not expressible in the k^{th} level of the FO+LFP hierarchy. A query in the k^{th} level of the FO+IFP hierarchy takes atmost n^k iterations, n = size of the database domain, for its evaluation. However, a query in the k^{th} level of the FO+GFP hierarchy may take 2^p, $p=n^k$, iterations for its evaluation. At present it is not known whether FO+GFP, restricted to queries that take polynomial number of iterations for their evaluation, is equivalent to FO+IFP. However, we show that there exist k-ary predicates which are not expressible in the k^{th} level of the FO+IFP hierarchy but are expressible in the k^{th} level of the FO+GFP hierarchy. Further the FO+GFP queries used to express these predicates take atmost n iterations for their evaluation. These results are shown to hold even in the presence of a successor predicate on the database domain.

2. Definitions

A finite **structure** (or a **relational database**) S with a **vocabulary** $\mathbf{v} = \langle \bar{R}_1,\ldots,\bar{R}_k,\bar{c}_1,\ldots,\bar{c}_m\rangle$ can be regarded as a tuple $S = \langle D,R_1,\ldots,R_k,c_1,\ldots,c_m\rangle$ consisting of a **finite** domain D, predicates (or relations) $R_1,\ldots,R_k$ on D corresponding to the predicate-symbols $\bar{R}_1,\ldots,\bar{R}_k$ of $\mathbf{v}$ and constants $c_1,\ldots,c_m$, which are elements of D, corresponding to the constant-symbols $\bar{c}_1,\ldots,\bar{c}_m$ from $\mathbf{v}$. In future we shall often use the same symbol to denote a predicate (constant) and its predicate-symbol (constant-symbol) relying on the context to resolve the ambiguity.

Two structures $S = \langle D,R_1,\ldots,R_k,c_1,\ldots,c_m\rangle$ and $S' = \langle D',R_1',\ldots,R_k',c_1',\ldots,c_m'\rangle$, with the same vocabulary, are **isomorphic** if there exists a one-one onto mapping $h : D \rightarrow D'$ such that $R_i' = h(R_i)$, $1\leq i\leq k$, and $c_i' = h(c_i)$, $1\leq i\leq m$.

Given a vocabulary $\mathbf{v}$, a **query** Q is a function which maps each structure S, with vocabulary $\mathbf{v}$, to a k-ary predicate R^k defined over the domain of S, i.e., $Q : S \rightarrow R^k$. In addition, a query Q must result in isomorphic predicates on isomorphic structures [CH], i.e, if h is an isomorphism from S to S′ then $Q(S') = h(Q(S))$.

The **first-order language** of a vocabulary $\mathbf{v}$, $\mathbf{FO}(\mathbf{v})$, is the set of all formulae of first-order logic with equality [En] built using the symbols of $\mathbf{v}$. The query language **FO** is the union of the languages $FO(\mathbf{v})$ for all possible vocabularies, i.e, $FO = \bigcup_{\mathbf{v}} FO(\mathbf{v})$.

Let $\mathbf{g}_\mathbf{v}(P,\bar{x})$ be a first-order formula built using the symbols from some vocabulary $\mathbf{v}$ (in future we shall omit the subscript $\mathbf{v}$) and an additional k-ary predicate symbol P which is **not** in $\mathbf{v}$. Let $\bar{x} = (x_1,x_2,\ldots,x_k)$ be the sequence of free variables occuring in $\mathbf{g}$. Given a structure S, with vocabulary $\mathbf{v}$, on domain D, we can use $\mathbf{g}$ to compute P in an iterative fashion until a fixed-point is reached, i.e.,

$$P(\bar{x}) = \mathbf{g}^i(\emptyset,\bar{x}) = \mathbf{g}^{i+1}(\emptyset,\bar{x}) \quad \text{where}$$
$$\mathbf{g}^i(\emptyset,\bar{x}) = \mathbf{g}(\mathbf{g}^{i-1}(\emptyset,\bar{x}),\bar{x}) \quad \text{and} \quad \mathbf{g}^1(\emptyset,\bar{x}) = \mathbf{g}(\emptyset,\bar{x}).$$

We emphasize that the length of $\mathbf{g}$ is fixed, i.e., its length is independent of the size of D, but the number of iterations needed to reach a fixed-point may depend on the size of D. There exist formulae for which the above iterative computation never terminates, e.g.,

$$\mathbf{f}(P,x) = [\forall y \neg P(y) \wedge x = x] \vee [\exists y\, P(y) \wedge x \neq x].$$

In the following we only consider those formulae for which the iterative computation terminates on all valid input structures. In general, it is undecidable to check this property.

A k-ary predicate P_{fp} is a **fixed-point (fp)** of $\mathbf{g}(P,\bar{x})$ if $P_{fp}(\bar{x}) = \mathbf{g}(P_{fp},\bar{x})$ and P_{lfp} is the **least fixed-point (lfp)** of $\mathbf{g}$ if it is contained in every fixed-point of $\mathbf{g}$. The iteratively computed **fp** of $\mathbf{g}$ is called the **generalized fixed-point (gfp)** of $\mathbf{g}$. If $\mathbf{g}$ is positive in P, i.e., each occurence of P in $\mathbf{g}$ is under an even number of negations, then the **lfp** of $\mathbf{g}$ is guaranteed to exist and it is the same as its **gfp** [CH]. The **inductive fixed-point (ifp)** of $\mathbf{g}$ is the **gfp** of $\mathbf{h}(P,\bar{x}) = P(\bar{x}) \vee \mathbf{g}(P,\bar{x})$. If $\mathbf{g}$ is positive in P then the **lfp** of $\mathbf{g}$ is the same as its **ifp** [GS].

It has been shown in [GS] that $\mathbf{h}^i(\emptyset,\bar{x}) \subseteq \mathbf{h}^{i+1}(\emptyset,\bar{x})$, i.e., the iterative computation of **ifp (lfp)** proceeds monotonically. Hence, if $|D| = n$ then the **ifp (lfp)** of $\mathbf{g}$ can be computed in atmost n^k iterations where k is the arity of P. However, in general, the iterative computation of the **gfp** need not proceed monotonically. Thus the **gfp** computation may take upto 2^p, $p = n^k$, iterations.

The LFP operator accepts an FO formula $\mathbf{g}(P,\bar{x})$, positive in P, and computes its **lfp**. Similarly, the IFP (GFP) operator computes the **ifp (gfp)** of a given formula. The query languages based on the extension of FO by the above fixed-point operators are defined as follows :

FO+LFP = { LFP $\mathbf{g}(P,\bar{x})$: $\mathbf{g}$ is positive in P } ,
FO+IFP = { IFP $\mathbf{g}(P,\bar{x})$ } and
FO+GFP = { GFP $\mathbf{g}(P,\bar{x})$ } .

The query languages $\mathbf{FO+LFP^k}$, $\mathbf{FO+IFP^k}$ and $\mathbf{FO+GFP^k}$, $k\geq 1$, are obtained from FO+LFP, FO+IFP and FO+GFP respectively by restricting the arity of the iteratively defined predicate P to exactly k. Since LFP $g(P,\bar{x})$ = IFP $g(P,\bar{x})$, if g is positive in P, and IFP $g(P,\bar{x})$ = GFP $h(P,\bar{x})$, $FO+LFP^k \subseteq FO+IFP^k$ and $FO+IFP^k \subseteq FO+GFP^k$. A query Q which defines a j-ary predicate R^j over structures with vocabulary **v** is said to be expressible in $FO+LFP^k$, $k \geq j$, iff there exists a formula $g(P,\bar{x})$ in $FO+LFP^k$ such that on all structures with vocabulary **v**

$$(d_1,\ldots,d_j) \in R^j \iff (d_1,\ldots,d_j,d_1',d_2',\ldots,d_m') \in P$$

where $m=k-j$ and each d_i', $1 \leq i \leq m$, is either an element of the set $\{d_1,\ldots,d_j\}$ or is a constant. Similar definitions of expressibility can be given for the FO+IFP and FO+GFP hierarchies.

It was pointed out in [CH] that even FO+GFP failed to express some simple queries, e.g., checking whether a given predicate has an even number of tuples. Immerman [Im2] pointed out that this query could be expressed by adding a successor predicate Suc(x,y), which enforces a total ordering on D, to FO+LFP. In fact the availability of Suc enables FO+LFP (FO+IFP) to express all queries computable in polynomial time [Im2,GS] and enables FO+GFP to express all queries computable in polynomial space (in the size of the structure) [Im2].

If Suc is available then we can construct formulae $g(P,Suc,\bar{x})$ by using the symbols from some vocabulary **v** and the predicate-symbol Suc. Given a structure S with vocabulary **v**, $g(P,Suc,\bar{x})$ is evaluated by augmenting S with some predicate $Suc_{p,D}$, where if $D = \{d_1,\ldots,d_n\}$ and p is some permutation on $\{1,2,\ldots,n\}$ then $Suc_{p,D} = \{(d_{p(i)},d_{p(i+1)}) : 1 \leq i \leq n-1\}$. The tuples of $Suc_{p,D}$ give us a total ordering on D as follows. Each tuple $(d_{p(i)},d_{p(i+1)}) \in Suc_{p,D}$ is interpreted to mean that $d_{p(i)}$ is the immediate successor of $d_{p(i+1)}$. The minimum element is $d_{p(n)}$ and the maximum element is $d_{p(1)}$.

Unfortunately, the availability of Suc allows one to write formulae where the value of the defined predicate depends on the particular $Suc_{p,D}$ substituted for Suc, e.g.,

$$\max(x) \iff \forall z\ [\ z \neq x \implies \neg Suc(z,x)\ ].$$

Such formulae are not queries since they do not assign a unique predicate to a given structure. Therefore we shall only consider formulae $g(P,Suc,\bar{x})$ which are **Suc-invariant**, i.e., on all input structures the value of P computed is independent of the particular $Suc_{p,D}$ substituted for the predicate-symbol Suc in **g**. It has been shown in [Du] that Suc-invariant formulae are queries.

Thus, if we restrict ourselves to Suc-invariant formulae, the addition of Suc to FO+LFP, FO+IFP and FO+GFP leads to the following query languages :

FO+LFP+Suc = { LFP $g(P,Suc,\bar{x})$ } ,
FO+IFP+Suc = { IFP $g(P,Suc,\bar{x})$ } and
FO+GFP+Suc = { GFP $g(P,Suc,\bar{x})$ }.

The query languages **$FO+LFP^k+Suc$**, **$FO+IFP^k+Suc$** and **$FO+GFP^k+Suc$** are obtained from FO+LFP+Suc, FO+IFP+Suc and FO+GFP+Suc respectively by restricting the arity of predicate variable P to exactly k, $k \geq 1$.

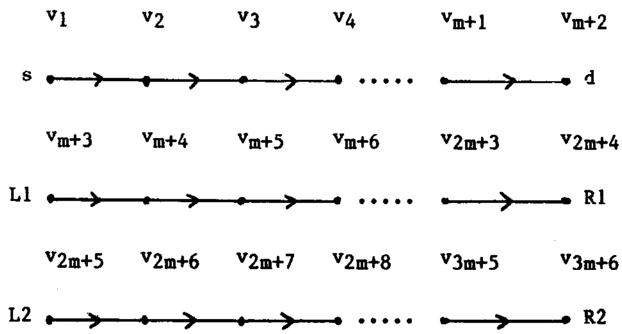
3. Higher Arity Leads to More Expressive Power

In this section we show that for each fixed-point operator increasing the arity of the predicate variable P from k to k+1, $k \geq 1$, always allows some more k-ary predicates to be expressed. We first show that there exist k-ary predicates which are not expressible in $FO+IFP^k+Suc$ but are expressible in $FO+LFP^{k+1}$. Since $FO+LFP^k \subseteq FO+LFP^k+Suc \subseteq FO+IFP^k+Suc$, $FO+IFP^k \subseteq FO+IFP^k+Suc$ and $FO+LFP^k \subseteq FO+IFP^k$, we immediately obtain the desired results for the FO+LFP, FO+IFP, FO+LFP+Suc and FO+IFP+Suc hierarchies. We then show that there exist k-ary predicates which are not expressible in $FO+GFP^k+Suc$ but are expressible in $FO+IFP^{k+1}$. Since $FO+GFP^k \subseteq FO+GFP^k+Suc$ and $FO+IFP^k \subseteq FO+GFP^k$, we immediately obtain the desired results for the FO+GFP and FO+GFP+Suc hierarchies.

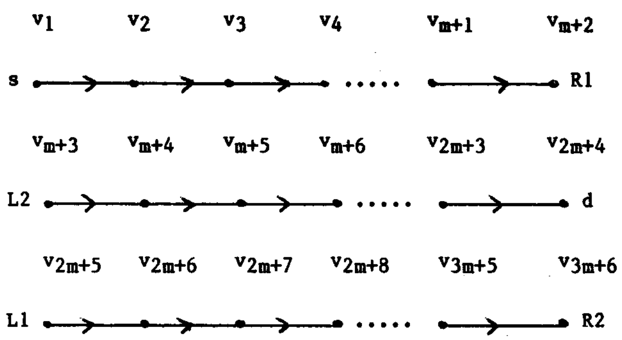
Consider a query on digraphs which defines the following k-ary predicate, $k \geq 2$:

$R^k(x_1,x_2,\ldots,x_k) \iff$
indegree$(x_1)=0$, outdegree$(x_k)=0$,
$E(x_i,x_{i+1})$, $1 \leq i \leq k-2$, $x_i \neq x_j$, $1 \leq i \neq j \leq k$, and
there exists a path from x_1 to x_k.

To show that R^k is not expressible in $FO+IFP^k+Suc$ we make use of digraphs G_n and H_n shown in Figure 1. G_n and H_n are structures with vocabulary **v** = <E,s,d,L1,R1,L2,R2> where E is the edge predicate and s,d,L1,R1,L2 and R2 are constants. We augment G_n and H_n with the successor predicate $Suc_V = \{ (v_i,v_{i+1}) : 1 \leq i \leq 3m+5 \}$.



The Graph G_n



The Graph H_n

Figure 1

We now introduce a tool which will be used to show that R^k is not expressible in FO+IFP^k+Suc. First we introduce a technical definition. The **quantifier rank (QR)** of an FO sentence is the maximum depth of nesting of quantifiers in it. An **Ehrenfeucht-Fraisse (EF)** game [Eh,Fr] is played by two players, P1 and P2, on a pair of structures G and H with the same vocabulary. P1 tries to show that the two structures are different whereas P2 tries to keep them looking alike. Formally the M move game is defined as follows :

At the i^{th} move, $1\le i\le M$, P1 chooses an element g_i (h_i) from G (H) and P2 responds with an element h_i (g_i) from H (G). P2 is said to win the M move game if the map which sends constants from G to constants from H and maps g_i to h_i, $1\le i\le M$, is an isomorphism of the induced substructures, i.e., the substructures obtained from G and H by restricting their respective domains to the constants and the elements chosen by both the players. The usefulness of EF games results from the following theorem :

Theorem [Eh,Fr] : P2 has a winning strategy for the M move game on G and H iff G and H satisfy the same set of FO sentences of QR M. ∎

We show that P2 wins the log m move EF game played on the augmented structures G_n+Suc$_V$ and H_n+Suc$_V$. Hence G_n+Suc$_V$ and H_n+Suc$_V$ agree on all FO sentences of QR log m, $m = n/3 - 2$. Therefore, no fixed-length FO sentence can distinguish G_n+Suc$_V$ from H_n+Suc$_V$. We then show that this is contradicted if R^k is expressible in FO+IFP^k+Suc.

Lemma 1 : P2 wins the log m move EF game played on structures G_n and H_n augmented with Suc$_V$.

Proof (Sketch) : We define a few terms before proceeding with the proof. Let $d_E(v_i,v_j)$ denote the **E-distance**, i.e., the length of the path, from v_i to v_j in G_n (H_n). Let $d_{Suc}(v_i,v_j)$ denote the **Suc-distance** from v_i to v_j, i.e., the length of the path from v_i to v_j in the graph constructed on V by using the tuples of Suc$_V$ as directed edges. Note that if $j=i+r$, $r\ge 1$, then $d_{Suc}(v_i,v_j)=r$ and $d_{Suc}(v_j,v_i)=\infty$. However $d_E(v_i,v_j)=r$ if $j=i+r$ and v_i and v_j lie in the same row of G_n (H_n). If v_i and v_j lie in different rows then $d_E(v_i,v_j)=d_E(v_j,v_i)=\infty$. An **r-chain**, $r\le m$, is a sequence of vertices $g_1,g_2,\ldots,g_p$ such that $d_E(g_i,g_{i+1})=d_{Suc}(g_i,g_{i+1})\le r$, $1\le i\le p-1$. Note that this definition ensures that all the vertices in a chain must be from the same row of G_n (H_n). Two chains $g_1,g_2,\ldots,g_p$ and $h_1,h_2,\ldots,h_p$ are **isomorphic** iff $d_E(g_i,g_{i+1})=d_E(h_i,h_{i+1})$ and $d_{Suc}(g_i,g_{i+1}) = d_{Suc}(h_i,h_{i+1})$, $1\le i\le p-1$, and some g_i is a constant iff h_i is also the same constant. Two chains $g_1,g_2,\ldots,g_p$ and $h_1,h_2,\ldots,h_q$ are **r_1,r_2-disjoint** if

(i) $d_E(g_i,h_j) > r_1$ and $d_E(h_j,g_i) > r_1$, $1\le i\le p$ and $1\le j\le q$, and
(ii) $d_{Suc}(g_i,h_j)>r_2$ and $d_{Suc}(h_j,g_i)>r_2$, $1\le i\le p$ and $1\le j\le q$.

A vertex v_j is said to be **r-free** from a chain if $d_E(g_i,v_j)>r$ and $d_E(v_j,g_i)>r$ for each g_i belonging to the chain.

P2 wins the log m move game by the following strategy. At the first move any vertex chosen by P1 from G_n (H_n) is at an E-distance of r, $r\le m/2$, of exactly one constant from G_n (H_n). P2 responds with a vertex from H_n (G_n) which is also at an E-distance of r of the same constant from H_n (G_n). Inductively, with j moves remaining the vertices from G_n (H_n) chosen by both the players and also the constants s,d,L1,R1,L2 and R2 (even if these are not chosen) can be partitioned into 2^j-chains $\bar{g}_1,\ldots,\bar{g}_p$ ($\bar{h}_1,\ldots,\bar{h}_p$) such that

(i) $\bar{g}_i$ is isomorphic to $\bar{h}_i$, $1\le i\le p$,
(ii) $\bar{g}_i$ and $\bar{g}_x$ ($\bar{h}_i$ and $\bar{h}_x$), $i\ne x$, are mutually either $2^j,2^j$-disjoint or $2^j,1$-disjoint, and
(iii) $\bar{g}_i$ and $\bar{g}_x$, $i\ne x$, are mutually $2^j,2^j$-disjoint ($2^j,1$-disjoint) iff $\bar{h}_i$ and $\bar{h}_x$ are mutually $2^j,2^j$-disjoint ($2^j,1$-disjoint).

Now, suppose P1 picks up a vertex v_g from G_n at the next, i.e., $\log m -j+1^{th}$, move. Then the following cases are possible :

Case 1 : v_g is at an E-distance r, $r\le 2^{j-1}$, of exactly one chain $\bar{g}_i$.

Case 2 : v_g is 2^{j-1}-free from each $\bar{g}_i$, $1\le i\le p$.

It can be shown that in case 1, P2 can respond with a vertex v_h from H_n which is at an E-distance of r from the chain $\bar{h}_i$. Similarly, in case 2 P2 can respond with a vertex v_h which is also 2^{j-1}-free from each chain $\bar{h}_i$. The details are given in [Du].

The case when P1 chooses a vertex from H_n can be similarly handled. At the end of the above move, the chains $\bar{g}_i$ and $\bar{h}_i$, $1\le i\le p$, and the chosen points v_g and v_h can be split into isomorphic 2^{j-1}-chains which are mutually either $2^{j-1},2^{j-1}$-disjoint or $2^{j-1},1$-disjoint. Thus the induction hypothesis is true with j-1 moves remaining. At the end, with no moves remaining, i.e., j=0, P2 wins the EF game. ∎

Lemma 2 : R^k is not expressible in FO+IFP^k+Suc, $k\ge 2$.

Proof : Suppose that $R^k(x_1,x_2,\ldots,x_k)$ is expressible in FO+IFP^k+Suc. Let $g(P,Suc,\bar{x})$ be a formula such that **ifp** of **g** is R^k. If G_n is the input structure then the **ifp** of **g** must contain the tuple $(s,v_2,\ldots,v_{k-1},d)$ but if H_n is the input structure then the **ifp** of **g** contains the tuple $(s,v_2,\ldots,v_{k-1},R1)$. Note that on both the structures the **ifp** of **g** contains exactly three tuples, one for each row of vertices. Therefore the **ifp** will be computed in atmost three iterations. Hence the sentence

$$\exists x_2,\ldots,x_{k-1}\;\; g^3(\emptyset,Suc,s,x_2,\ldots,x_{k-1},d)$$

is true on the augmented structure G_n+Suc_G and false on the augmented structure H_n+Suc_H where Suc_G and Suc_H are any two (maybe even same) valid succesor predicates on V (recall that **g** is Suc-invariant). This contradicts Lemma 1 and hence our initial assumption was incorrect. ∎

Lemma 3 : R^k is expressible in FO+LFP^{k+1}, $k\ge 2$.

Proof : We use a k+1-ary predicate variable to iteratively traverse the paths originating at zero-indegree vertices. When the path originating at x_1, indegree(x_1)=0, traverses a vertex z, $z\ne x_1$, the tuple $(x_1,\ldots,x_1,z)$ is added to P. If at any stage a tuple $(x_1,\ldots,x_1,x_k)$, outdegree(x_k)=0, is found in P then $(x_1,x_2,\ldots,x_k,x_k)$ is added to P iff $(x_1,x_2,\ldots,x_k)\in R^k$. The following formula accomplishes this task :

$$\begin{aligned} g(P,x_1,x_2,\ldots,x_{k+1}) = \; & [\, E(x_k,x_{k+1}) \wedge \forall z \neg E(z,x_k) \wedge x_k\ne x_{k+1} \wedge \bigwedge_{i=1}^{k-1} x_i=x_k \,] \\ \vee \; & [\, \exists z\, P(x_k,\ldots,x_k,z) \wedge E(z,x_{k+1}) \wedge x_k\ne x_{k+1} \wedge \bigwedge_{i=1}^{k-1} x_i=x_k \,] \\ \vee \; & [\, P(x_1,\ldots,x_1,x_k) \wedge \forall z \neg E(x_k,z) \wedge \bigwedge_{i=1}^{k-2} E(x_i,x_{i+1}) \wedge \bigwedge_{1\le i\ne j\le k} x_i\ne x_j \wedge x_k=x_{k+1} \,] \,. \end{aligned}$$

Since **g** is positive in P, its **lfp** is well defined. In the first and each subsequent iteration, the first disjunct adds a tuple $(x_1,\ldots,x_1,y)$ to P for each edge $E(x_1,y)$, $x_1\ne y$, in the input graph such that indegree(x_1)=0. Since P is empty during the first iteration, the other two disjuncts do not add any tuples to P.

The second disjunct traverses paths starting at vertices whose indegree is zero. For each such path, its first edge has already been added to P by the first disjunct. In the i^{th} iteration, $i\ge 1$, for each tuple $(x_1,\ldots,x_1,y)$ in P, the second disjunct checks if there is an edge $E(y,z)$, $z\ne x_1$, and adds the tuple $(x_1,\ldots,x_1,z)$ to P. Thus at the end of the i^{th} iteration P contains a tuple $(x_1,\ldots,x_1,y)$, $x_1\ne y$, iff indegree(x_1)=0 and there is a path of length $\le i$ from x_1 to y. If there is a path from x_1, indegree(x_1)=0, to x_k, outdegree(x_k)=0, then the tuple $(x_1,\ldots,x_1,x_k)$ will be added to P in atmost n-1 iterations. The third disjunct checks if the tuple $(x_1,\ldots,x_k)$ satisfies R^k and adds the tuple $(x_1,\ldots,x_k,x_k)$ to P. Since the tuples added by the first two disjuncts always have distinct k^{th} and $k+1^{th}$ components, it follows that

$$(x_1,\ldots,x_k) \in R^k \iff (x_1,\ldots,x_k,x_k) \in \text{LFP } g \,. \quad \blacksquare$$

Lemma 4 : There exists a unary predicate which is not expressible in FO+IFP^1+Suc but is expressible in FO+LFP^2.

Proof : Consider digraphs which are structures with vocabulary <E,s,d,L1,R1,L2,R2>. Consider a query on digraphs which defines the following unary predicate :

$R^1(x) \iff$ outdegree(x)=0 and there is a path from s to x.

By arguments similar to those used in Lemmas 1 and 2 we can show that R^1 is not expressible in FO+IFP^1+Suc. However R^1 can be expressed in FO+LFP^2 as shown below :

$$\begin{aligned} g(P,x_1,x_2) = &\ [\ x_1=s \wedge E(x_1,x_2) \wedge x_1 \neq x_2] \\ \vee &\ [\ \exists z\ P(x_1,z) \wedge P(z,x_2) \wedge x_1 \neq x_2\ ] \\ \vee &\ [\ \exists z\ P(z,x_1) \wedge \forall y\ (\ E(x_1,y) \implies y=x_1\ ) \wedge \\ &\ x_1=x_2\ ] \,. \quad \blacksquare \end{aligned}$$

Since FO+LFP^k $\subseteq$ FO+IFP^k and FO+LFP^k+Suc $\subseteq$ FO+IFP^k+Suc, Lemmas 1 - 4 give us

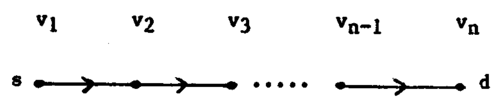
Theorem 1 : For $k \geq 1$, there exist k-ary predicates
(i) which are not expressible in FO+LFP^k (FO+LFP^k+Suc) but are expressible in FO+LFP^{k+1} (FO+LFP^{k+1}+Suc) and
(ii) which are not expressible in FO+IFP^k (FO+IFP^k+Suc) but are expressible in FO+IFP^{k+1} (FO+IFP^{k+1}+Suc). $\blacksquare$

We now show that there exist k-ary predicates which are not expressible in FO+GFP^k+Suc but are expressible in FO+IFP^{k+1}, $k \geq 1$. Consider the set of digraphs which are structures with the vocabulary <E,s,d>. We require that the digraphs be free from self-loops, indegree(s) = outdegree(d) = 0, outdegree(s) = indegree(d) = 1, and all other vertices must have an indegree and outdegree of one. Let F_1 be an FO sentence which checks if a given digraph satisfies these conditions. The following query defines a k-ary predicate, $k \geq 1$, on structures which satisfy F_1 :

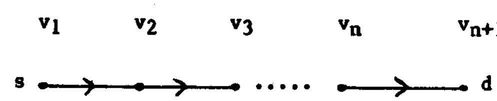
$R^k(x_1,x_2,\ldots,x_k) \iff$ path length from s to x_1 = path length from x_k to d and $E(x_i,x_{i+1})$, $1 \leq i \leq k-1$.

Lemma 5 : R^k is not expressible in FO+GFP^k+Suc, $k \geq 1$.

Proof : Consider structures $S_n = \langle V_n, Suc_n, v_1, v_n \rangle$ and $S_{n+1} = \langle V_{n+1}, Suc_{n+1}, v_1, v_{n+1} \rangle$ with the vocabulary <Suc,s,d> where $V_t = \{ v_i : 1 \leq i \leq t \}$ and $Suc_t = \{ (v_i,v_{i+1}) : 1 \leq i \leq t-1 \}$. Using the structure S_n (S_{n+1}) we can define, by an FO formula, a digraph G_n (G_{n+1}) whose edges are the tuples of Suc_n (Suc_{n+1}) (see Figure 2). Note that both G_n and G_{n+1} satisfy the sentence F_1. **Wlg** we assume that n is even.



The Graph G_n



The Graph G_{n+1}

Figure 2

If k is even (odd) then R^k contains exactly one tuple when evaluated on G_{n+1} (G_n) and is empty when evaluated on G_n (G_{n+1}). Suppose that R^k is expressible in FO+GFP^k+Suc. Let R^k be the **gfp** of $g(P,Suc,x_1,\ldots,x_k)$. If k is even then the **gfp** of **g** is empty when G_n is the input structure and contains a single tuple when G_{n+1} is the input structure. Thus the sentence

$$\exists\, x_1,x_2,\ldots,x_k\ g(\emptyset,Suc,x_1,x_2,\ldots,x_k)$$

is true on G_{n+1}. We claim that this sentence is false on G_n. This is so for the following reason. Suppose that $g(\emptyset,x_1,\ldots,x_k) \neq \emptyset$ on G_n. Since the **gfp** of **g** is empty on G_n, $g^i(\emptyset,\bar{x}) = \emptyset$, $i>1$. Thus $g(\emptyset,\bar{x}) \neq g^i(\emptyset,\bar{x})$ and $g(\emptyset,\bar{x}) = g^{i+1}(\emptyset,\bar{x})$. Hence the iterative computation of the **gfp** will never terminate on G_n, a contradiction. Thus we have a fixed-length FO sentence which distinguishes S_n from S_{n+1}. The case when k is odd can be handled similarly.

However, it is shown in [Gu] that P2 wins the log n - 1 move EF game played on the structures S_n and S_{n+1}. Thus no fixed-length sentence can distinguish these structures, a contradiction. ∎

Lemma 6 : R^k is expressible in $FO+IFP^{k+1}$, $k \geq 1$.

Proof : Since the input structure satisfies F_1, it has a unique path which originates at s and terminates at d. Our strategy is to simultaneously traverse this path, one edge at a time, in the forward and reverse direction starting from s and d respectively. Each step of this traversal yields vertices x_1 and x_k such that the distance of x_1 from s equals the distance of d from x_k. If the distance of x_k from x_1 is k-1 then we are done.

The following formula uses a k+1-ary predicate variable to keep track of the paths traversed :

$$\mathbf{g}(P,x_1,\ldots,x_k) = F_1 \wedge g1(P,x_1,\ldots,x_k) \quad \text{where}$$

$$g1(P,x_1,\ldots,x_{k+1}) =$$

$$[\ E(s,x_{k+1}) \wedge \bigwedge_{i=1}^{k} x_i = s\ ]$$

$$\vee\ [\ E(x_k,d) \wedge x_{k+1}=d \wedge \bigwedge_{i=1}^{k-1} x_i = x_k]$$

$$\vee\ [\ \exists z\ P(s,\ldots,s,z) \wedge E(z,x_{k+1}) \wedge \bigwedge_{i=1}^{k} x_i = s\ ]$$

$$\vee\ [\ \exists z\ P(z,\ldots,z,d) \wedge E(x_k,z) \wedge \bigwedge_{i=1}^{k-1} x_i = x_k \wedge x_{k+1} = d\ ]$$

$$\vee\ [\ P(s,\ldots,s,x_1) \wedge P(x_k,\ldots,x_k,d) \wedge \bigwedge_{i=1}^{k-1} E(x_i,x_{i+1}) \wedge x_k = x_{k+1} \wedge$$

$$\bigwedge_{i=2}^{k-2} \neg (\ P(s,\ldots,s,x_i) \vee P(x_i,\ldots,x_i,d)\ )\ ]\ .$$

The conjunct F_1 ensures that if the input structure does not satisfy F_1 then the **gfp** computation terminates at the first iteration with an empty predicate. In the first iteration, when P is empty, only the first two disjuncts of g1 add tuples to P. The first (second) disjunct adds a tuple (s,...,s,x) ((x,...,x,d)) for each edge E(s,x) (E(x,d)) in the input graph.

The third disjunct traverses the path starting from s, in the forward direction, while the fourth disjunct traverses the path terminating at d in the reverse direction. In the i^{th} iteration, i>1, the third (fourth) disjunct adds a tuple (s,...,s,x) ((x,...,x,d)) to P iff the path length from s to x (x to d) is exactly i. Thus the tuples $(s,\ldots,s,x_1)$ and $(x_k,\ldots,x_k,d)$ are added to P in the same iteration iff the length of the path from s to x_1 is equal to the length of the path from x_k to d.

If $(x_1,x_2,\ldots,x_k) \in R^k$ then there are exactly k-2 vertices x_i, $2 \leq i \leq k-2$, lying between x_1 and x_k. Further for each x_i lying between x_1 and x_k neither $(s,\ldots,s,x_i)$ nor $(x_i,\ldots,x_i,d)$ would have been added to P. The last disjunct uses this fact to check whether the tuple $(x_1,x_2,\ldots,x_k)$ satisfies R^k. If the check is successful it adds the tuple $(x_1,x_2,\ldots,x_k,x_k)$ to P. Note that the computation of the **gfp** proceeds monotonically (even though P occurs under an odd number of negations in g1) and therefore termination is guaranteed on all valid input structures. Further the **gfp** computation remains unchanged even if we replace $\mathbf{g}(P,\bar{x})$ by $P(\bar{x}) \vee \mathbf{g}(P,\bar{x})$. Since the tuples added to P by the first four disjuncts of g1 have distinct k^{th} and $k+1^{th}$ components, we have

$$(x_1,\ldots,x_k) \in R^k \iff (x_1,\ldots,x_k,x_k) \in \text{IFP } \mathbf{g}\ . \quad ∎$$

Since $FO+GFP^k \subseteq FO+GFP^k+Suc$ and $FO+IFP^k \subseteq FO+GFP^k$, Lemmas 5 and 6 give us the following theorem :

Theorem 2 : There exist k-ary predicates, $k \geq 1$, which are not expressible in $FO+GFP^k$ ($FO+GFP^k+Suc$) but are expressible in $FO+GFP^{k+1}$ ($FO+GFP^{k+1}+Suc$).

4. k-ary IFPs are More Powerful than k-ary LFPs

Consider structures with vocabulary $\langle E^k,s,d \rangle$ where E^k is a k-ary predicate-symbol. For k=2 such structures can be considered as digraphs and for k>2 they can be interpreted as directed hypergraphs [Be] in which each hyperedge has exactly k vertices. A **hyperedge** is an ordered tuple $\bar{e} = (x_1,x_2,\ldots,x_k)$ where x_1 is the **head** of $\bar{e}$, x_k is the **tail** of $\bar{e}$ and x_is, $2 \leq i \leq k-1$, are the **internal** vertices. The indegree (outdegree) of a vertex is r iff it is the tail (head) of exactly r hyperedges. If a vertex is not the tail (head) of any hyperedge then its indegree (outdegree) is zero. We only consider structures with the following properties :

(i) Outdegree(s)=2 and indegree(s)=0.
(ii) Outdegree(d)=0 and indegree(d)=2.
(iii) Each non-internal vertex, besides s and d, has an outdegree and indegree of one.
(iv) All vertices in a hyperedge are distinct.
(v) If a vertex is an internal vertex in some hyperedge then it appears in no other hyperegde.

Let F_k be an FO sentence such that a structure with vocabulary $\langle E^k, s, d\rangle$ satisfies F_k iff properties (i)-(v) are true for that structure. Note that the concept of a path in digraphs can be easily generalized for hypergraphs which satisfy F_k. Further, there are exactly two vertex disjoint paths from s to d in each structure which satisfies F_k. Consider a query which defines the following k-ary predicate :

$R^k(x_1,\dots,x_k) \Longleftrightarrow$
if the input structure satisfies F_k
then
  if the two paths from s to d contain the same number of hyperedges
  then
    R^k contains all the hyperedges from the two s-d paths
  else
    R^k contains all the hyperedges on the shorter s-d path and all but the last hyperedge on the longer s-d path
else R^k is empty ; ∎

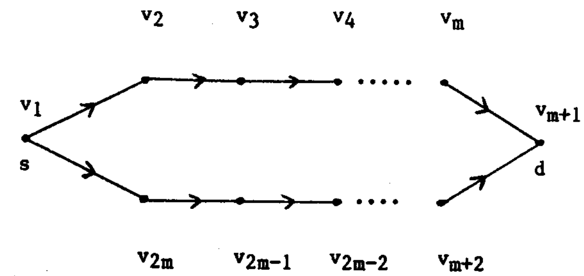
To show that R^k is not expressible in FO+LFP^k+Suc, we use structures $G_{k,n}$ and $H_{k,n+k-1}$ with vocabulary $\langle E^k, s, d\rangle$. For k=2, the graphs $G_{2,n}$ and $H_{2,n+1}$ are shown in Figure 3. The structure $G_{k,n}$ ($H_{k,n+k-1}$) can be obtained from $G_{2,n}$ ($H_{2,n+1}$) by replacing each edge by a hyperedge having k vertices. Formally, $G_{k,n}$ is the structure $\langle V_n, E^k, v_1, v_{m(k-1)+1}\rangle$, $n=2m(k-1)$, where

$V_n = \{ v_1, v_{m(k-1)+1} \} \cup V_{n,t} \cup V_{n,b}$,
$V_{n,t} = \{ v_i : 2\le i\le m(k-1) \}$,
$V_{n,b} = \{ v_i : m(k-1)+2\le i\le n \}$ and

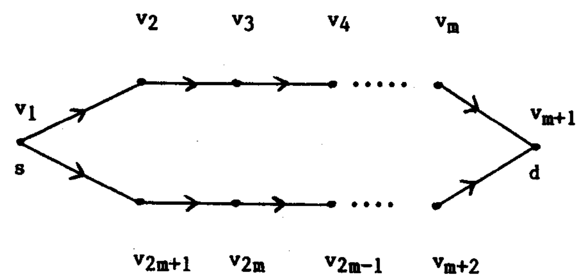
$E^k =$
$\{ (v_1, v_n, v_{n-1},\dots,v_{n-k+2}) \} \cup$
$\{ (v_i,\dots,v_{i+k-1}) : i=jk-(j-1) \text{ and } 0\le j\le m-1\} \cup$
$\{ (v_{i+k-1},\dots,v_i) : i=jk-(j-1) \text{ and } m\le j\le 2m-2\}$.

Similarly $H_{k,n+k-1}$ is the structure $\langle V_{n+k-1}, E^k, v_1, v_{m(k-1)+1}\rangle$ where

$V_{n+k-1} = \{ v_1, v_{m(k-1)+1} \} \cup V_{n+k-1,t} \cup V_{n+k-1,b}$,
$V_{n+k-1,t} = \{ v_i : 2\le i\le m(k-1) \}$,



The Graph $G_{2,n}$



The Graph $H_{2,n+1}$

Figure 3

$V_{n+k-1,b} = \{ v_i : m(k-1)+2\le i\le n+k-1 \}$ and

$E^k =$
$\{ (v_1, v_{n+k-1},\dots,v_{n+1}) \} \cup$
$\{ (v_i,\dots,v_{i+k-1}) : i= jk-(j-1) \text{ and } 0\le j\le m-1 \} \cup$
$\{ (v_{i+k-1},\dots,v_i) : i= jk-(j-1) \text{ and } m\le j\le 2m-1 \}$.

Note that the both the s-d paths in $G_{k,n}$ have m hyperedges. However, $H_{k,n+k-1}$ has m hyperedges on the top path from s to d but the bottom path from s to d contains m+1 hyperedges. We augment $G_{k,n}$ and $H_{k,n+k-1}$ with successor predicates Suc_n and Suc_{n+k-1} respectively where
$Suc_t = \{ (v_i, v_{i+1}) : 1\le i\le t-1 \}$.

Lemma 7 : P2 wins the log m - 2 move EF game played on $G_{k,n}$ and $H_{k,n+k-1}$ augmented with Suc_n and Suc_{n+k-1} respectively.

Proof (Sketch) : We can consider a hyperedge $(g_1,\dots,g_k)$ in $G_{k,n}$ ($H_{k,n+k-1}$) as an ordered sequence of k binary edges (g_1,g_2), $(g_2,g_3),\dots,(g_{k-1},g_k)$. If v_x is the i^{th} component and v_y the j^{th} component, j>i, of some hyperedge $\bar{e}$ then $d_E(v_x,v_y)=j-i$. If v_x is the i^{th} component of hyperedge $\bar{e}_1$ and v_y is the j^{th} component of $\bar{e}_2$

and there are p hyperedges on the path from the tail of $\bar{e}_1$ to the head of $\bar{e}_2$, then $d_E(v_x,v_y) = (k-i+1)+pk+(j-1)$. However, in both cases $d_E(v_y,v_x) = \infty$. Note that in $G_{k,n}$ ($H_{k,n+k-1}$) if v_x and v_y lie in different rows then $d_E(v_x,v_y) = \infty$. We define $d_{Suc}(v_x,v_y)$ as in Lemma 1.

A **top r-chain** is a sequence of vertices $g_1,\ldots,g_p$ such that $d_E(g_i,g_{i+1})=d_{Suc}(g_i,g_{i+1})\leq r$, $1\leq i\leq p-1$. A **bottom r-chain** is a sequence of vertices $g_1,\ldots,g_p$ such that $d_E(g_i,g_{i+1})=d_{Suc}(g_{i+1},g_i)\leq r$, $1\leq i\leq p-1$. The notion of isomorphism for top (bottom) chains, r_1,r_2-disjoint chains and r-free vertices carry over from Lemma 1.

Inductively, with j moves remaining, the vertices chosen by both the players from $G_{k,n}$ and the constants s and d can be partitioned into top 2^j-chains (bottom 2^j-chains) $\bar{g}_{ts}$, $\bar{g}_{td}$, $\bar{g}_1,\ldots,$ $\bar{g}_p$ ($\bar{g}_{bs}$, $\bar{g}_{bd}$, $\bar{g}_{p+1},\ldots,\bar{g}_q$). These top (bottom) chains are mutually $2^j,2^j$-disjoint. Further the chains $\bar{g}_x$, $1\leq x\leq p$, and $\bar{g}_y$, $p+1\leq y\leq q$, are mutually $2^j,2^j$-disjoint. Since the chains $\bar{g}_{ts}$ and $\bar{g}_{bs}$ ($\bar{g}_{td}$ and $\bar{g}_{bd}$) contain the constant s (d), they are mutually $2^j,0$-disjoint. Similarly the chosen vertices from $H_{k,n+k-1}$ and the constants s and d can be partitioned into top (bottom) 2^j-chains $\bar{h}_{ts},\bar{h}_{td},\bar{h}_1,\ldots,\bar{h}_p$ ($\bar{h}_{bs},\bar{h}_{bd},\bar{h}_{p+1},\ldots,\bar{h}_q$) with the same properties as the corresponding chains in $G_{k,n}$. The chains $\bar{g}_i$ and $\bar{h}_i$, $1\leq i\leq q$, $\bar{g}_{ts}$ and $\bar{h}_{ts}$, $\bar{g}_{bs}$ and $\bar{h}_{bs}$, $\bar{g}_{td}$ and $\bar{h}_{td}$, and $\bar{g}_{bd}$ and $\bar{h}_{bd}$ are isomorphic.

At the next, i.e., log m $-(j+1)^{th}$, move if P1 chooses a constant from $G_{k,n}$ then P2 responds with the same constant from $H_{k,n+k-1}$. Otherwise suppose that P1 chooses $v_g \in V_{n,t}$ such that v_g is the q^{th} component of some hyperedge. Then either v_g is not 2^{j-1}-free from some top chain $\bar{g}_i$ or v_g is 2^{j-1}-free from all top chains. In either case, we can show that P2 can respond with a vertex $v_h \in V_{n+k-1}$, v_h is also the q^{th} component of some hyperedge, such that the inductive assertion remains true at the end of this move. The case when $v_g \in V_{n,b}$ or when P1 chooses a vertex from $H_{k,n+k-1}$ can be handled similarly. ∎

Lemma 8 : R^k is not expressible in FO+LFP^k+Suc, $k\geq 2$.

Proof : Suppose that R^k is expressible in FO+LFP^k+Suc. Then there exists a formula $\mathbf{g}(P,Suc,x_1,\ldots,x_k)$, positive in P, such that its **lfp** is R^k. Thus the **lfp** of **g** evaluated on $G_{k,n}+Suc_n$ contains a tuple $(x_1,\ldots,x_k)$ for each hyperedge $(x_1,\ldots,x_k)$ in $G_{k,n}$. However, the **lfp** of **g** evaluated on $H_{k,n+k-1}+Suc_{n+k-1}$ contains all the hyperedges of $H_{k,n+k-1}$ except the hyperedge on the bottom row whose tail is d. Since the **lfp** of **g** evaluated on $G_{k,n}+Suc_n$ is E^k, no proper subset of E^k can be a fixed-point of **g** when $G_{k,n}+Suc_n$ is the input structure.

Let T be a 2k-ary predicate defined as follows :

$$T(u_1,\ldots,u_k,x_1,\ldots,x_k) = E^k(u_1,\ldots,u_k) \wedge E^k(x_1,\ldots,x_k) \wedge \bigwedge_{i=1}^{k} x_i \neq u_i .$$

The k-ary predicate $T(u_1,\ldots,u_k,\_,\ldots,\_)$ evaluated on a hypergraph contains all its hyperedges except the hyperedge $(u_1,\ldots,u_k)$. Hence, the following sentence

$$\exists u_1,\ldots,u_k\ [\forall x_1,\ldots,x_k\ \mathbf{g}(T(u_1,\ldots,u_k,\_,\ldots,\_),Suc,x_1,\ldots,x_k) \iff T(u_1,\ldots,u_k,x_1,\ldots,x_k)\ ]$$

is true for $H_{k,n+k-1}+Suc_{n+k-1}$ but false for $G_{k,n}+Suc_n$ for all valid succesor predicates Suc_n and Suc_{n+k-1} (recall that **g** is Suc-invariant). This contradicts Lemma 8 and hence our initial assumption was incorrect. ∎

Lemma 9 : R^k is expressible in FO+IFP^k, $k\geq 2$.

Proof : We use a k-ary predicate variable P to store the hyperedges occuring on the two s-d paths. Initially, the two hyperedges out of s are placed in P. In each subsequent iteration one more hyperedge from each path is added to P. If the two s-d paths have the same number of hyperedges then the two hyperedges, whose tails are d, will be added to P in the same iteration. If the number of hyperedges on the two s-d paths are not equal, we make use of the above fact to avoid adding the last hyperedge on the longer path to P. The following formula accomplishes this task :

$$\mathbf{g}(P,x_1,\ldots,x_k) = F_k \wedge gl(P,x_1,\ldots,x_k) \quad \text{where}$$

$$gl(P,x_1,\ldots,x_k) = [\ x_1=s \wedge E^k(x_1,\ldots,x_k)\ ]$$
$$\vee\ [\ \exists y_1,\ldots,y_{k-1}\ P(y_1,\ldots,y_{k-1},x_1) \wedge E^k(x_1,\ldots,x_k) \wedge (\ x_k=d \implies \forall\ z_1,\ldots,z_{k-1}\ \neg P(z_1,\ldots,z_{k-1},d)\ )].$$

The conjunct F_k ensures that the fixed-point computation terminates in the first iteration with an empty predicate if the input hypergraph does not satisfy F_k. The first disjunct initializes P to contain the two hyperedges out of s. The second disjunct traverses the two s-d paths adding an extra hyperedge at each iteration. Note that this disjunct checks that P does not already contain a hyperedge whose tail is d before it adds such a hyperedge to P. Thus it ensures that the last hyperedge on the longer s-d path will not be added to P. The computation of the fixed-point of **g** proceeds monotonically even though P occurs negated in **g**. Thus the **ifp** of **g** is the same as the **gfp** of **g**. Therefore it follows that

$(x_1,\ldots,x_k) \in R^k \Longleftrightarrow (x_1,\ldots,x_k) \in \text{IFP } \mathbf{g}$. ∎

Lemmas 7-9 give us the following theorem :

Theorem 3 : There exist k-ary predicates, $k \geq 2$, which are not expressible in $FO+LFP^k$ ($FO+LFP^k+Suc$) but are expressible in $FO+IFP^k$ ($FO+IFP^k+Suc$).

5. k-ary GFPs are More Powerful than k-ary IFPs

In this section we show that there exist k-ary predicates, $k \geq 1$, which are not expressible in $FO+IFP^k$ but are expressible in $FO+GFP^k[n^k]$ where $FO+GFP^k[n^k]$ contains only those formulae of $FO+GFP^k$ for which the **fp** computation takes atmost n^k iterations (n = size of the domain). A similar result is also shown for $FO+IFP^k+Suc$ and $FO+GFP^k+Suc[n^k]$. Our proof strategy is to show the existence of k-ary predicates which are not expressible in $FO+IFP^k+Suc$ but are expressible in $FO+GFP^k$. Since $FO+IFP^k \subseteq FO+IFP^k+Suc$ and $FO+GFP^k[n^k] \subseteq FO+GFP^k+Suc[n^k]$, we immediately obtain the desired results.

Consider digraphs with the following properties :

(i) there are no self-loops ,
(ii) the indegree of any vertex is atmost one and
(iii) the outdegree of any vertex is atmost one.

Let F_3 be an FO sentence such that a digraph satisfies F_3 iff it satisfies properties (i)-(iii). Consider a query on such digraphs which defines the following k-ary predicate, $k \geq 2$:

$R^k(x_1,\ldots,x_k) \Longleftrightarrow$
indegree$(x_1)=0$, outdegree$(x_k)=0$,
$E(x_i,x_{i+1})$, $1 \leq i \leq k-2$, and there exists a path from x_1 to x_k.

It has been shown in Section 3, using Lemmas 1 and 2, that R^k is not expressible in $FO+IFP^k+Suc$. Note that the digraphs G_n and H_n used in Lemma 1 satisfy the sentence F_3. Thus it follows that R^k is not expressible in $FO+IFP^k+Suc$ even if the input structures are restricted to those satisfying F_3.

Lemma 10 : R^k is expressible in $FO+GFP^k[n]$, $k \geq 2$, when the input structures are restricted to those satisfying F_3.

Proof : The following formula uses a k-ary predicate variable to compute R^k :

$\mathbf{g}(P,x_1,\ldots,x_k) = F_3 \wedge g1(P,x_1,\ldots,x_k)$ where

$$
\begin{aligned}
g1(P,x_1,\ldots,x_k) = \;& [\forall y_1,\ldots,y_k \;\neg P(y_1,\ldots,y_k) \wedge \\
& \forall z \;\neg E(z,x_1) \wedge \exists z_1,\ldots,z_k \;\; z_1=x_1 \wedge \\
& \bigwedge_{i=1}^{k-1} E(z_i,z_{i+1}) \wedge z_k=x_k \wedge \bigwedge_{i=2}^{k-1} x_i=x_1] \\
\vee\; & [\exists z\; P(x_1,\ldots,x_1,z) \wedge E(z,x_k) \wedge \\
& \bigwedge_{i=2}^{k-1} x_i=x_1] \\
\vee\; & [P(x_1,\ldots,x_1,x_k) \wedge \forall z \;\neg E(x_k,z) \wedge \\
& \bigwedge_{i=1}^{k-2} E(x_i,x_{i+1})] \\
\vee\; & [P(x_1,x_2,\ldots,x_k) \wedge x_1 \neq x_2] .
\end{aligned}
$$

The conjunct F_3 ensures that if the input structure does not satisfy F_3 then the **gfp** computation terminates at the first iteration with an empty predicate. The **gfp** computation is actually carried out by g1. The first disjunct of g1 checks if P is empty (which it will be in the first iteration) and initializes it to contain tuples $(x_1,\ldots,x_1,y)$, indegree$(x_1)=0$, such that the path length from x_1 to y is exactly k-1. Note that digraphs which satisfy F_3 are free from self-loops and vertices with indegree greater than one. Hence if a tuple is added to P in the first iteration then the input digraph is guaranteed to contain vertices $x_1,\ldots,x_k$ such that $(x_1,\ldots,x_k) \in R^k$. Since P is empty during the first iteration, the other three disjuncts do not contribute any tuples. If no vertex in the input graph has an indegree of zero the **gfp** computation terminates at the first iteration with an empty predicate.

If P is non-empty at the end of the first iteration, the first disjunct will not add any tuples to P in any subsequent iteration since (as shown below) P will never be empty again. The function of the second disjunct is to traverse the paths originating at zero-indegree vertices. Note that for each such path, its first k-1 edges have already been traversed by the first disjunct and the corresponding tuple added to P. In the i^{th} iteration, $i \geq 2$, for each tuple $(x_1,\ldots,x_1,y)$ in P, the second disjunct checks if there is an edge from y to z and adds $(x_1,\ldots,x_1,z)$ to P if the check succeeds. Note that the absence of self-loops and vertices with indegree greater than one ensures that the tuple $(x_1,\ldots,x_1,z)$ was **not** added to P in any previous iteration. Hence at the end of the i^{th} iteration, $i \geq 2$, P contains a tuple $(x_1,\ldots x_1,z)$ iff the path length from x_1 to z is exactly k+i-2. We emphasize that the tuples added to P, by the first two disjuncts, in an iteration are **dropped** from P in the next iteration. This ensures that the **gfp** does nót contain tuples which do not belong to R^k.

The third disjunct keeps on checking whether P contains a tuple $(x_1,\ldots,x_1,x_k)$ where outdegree$(x_k)=0$. Such a tuple is guaranteed to be added to P in atmost n-1 iterations. If P contains such a tuple then the input digraph also contains the vertices $x_2,\ldots,x_{k-1}$ such that t $= (x_1,x_2,\ldots,x_k) \in R^k$. The third disjunct finds the vertices x_i, $2 \leq i \leq k-1$, and adds the tuple t to P. Note that if $(x_1,\ldots,x_1,x_k)$, which caused t to be added to P, was added to P in the i^{th} iteration then it will be dropped from P in the next iteration. Thus we must ensure that t does not drop out of P in any future iteration. Note that the tuples added to P by the third disjunct have distinct values in all the k components (this follows from the fact that the input graph satisfies F_3) but the tuples added by the first two disjuncts have the same value in the first k-1 components. The fourth disjunct uses this property to maintain the R^k tuples in P. Finally when the **gfp** computation terminates, the tuples in P are exactly the tuples of R^k, i.e.,

$$(x_1,\ldots,x_k) \in R^k \iff (x_1,\ldots,x_k) \in \text{GFP } \mathbf{g}.$$

We now show that the above **gfp** computation will terminate in atmost n iterations. Consider any two vertices x_1, indegree$(x_1)=0$, and x_k, outdegree$(x_k)=0$, such that there is a path from x_1 to x_k of length i, $i \geq k-1$. Hence the tuple $(x_1,\ldots,x_1,x_k)$ will be added to P in the $i-k+2^{th}$ iteration by first disjunct, if i=k-1, or by the second disjunct, if i>k-1. Therefore the third disjunct will add the tuple $(x_1,\ldots,x_k)$ to P in the $i-k+3^{th}$ iteration. Since $i \leq n-1$, each such tuple will be added to P by the end of the $n-k+2^{th}$ iteration. Since $k \geq 2$, the **gfp** computation will always terminate in n iterations. ∎

For the case when k=1 consider digraphs with vocabulary <E,s,d,L1,R1,L2,R2> which satisfy $F_4 = F_3 \wedge \forall z \neg E(z,s)$, i.e., in addition to satisfying F_3 the indegree of s is zero. Lemma 4 shows that a unary predicate R^1 is not expressible in FO+IFP^1+Suc on digraphs satisfying F_4. However, for digraphs which satisfy F_4, arguments similar to those in Lemma 10 can be used to show that R^1 is expressible by the following FO+GFP^1[n] formula

$$\begin{aligned} g(P,x) &= F_4 \wedge g1(P,x) \quad \text{where} \\ g1(P,x) &= [\, x=s \ \wedge \forall z \neg P(z) \,] \\ &\vee [\, \exists z\, P(z) \wedge E(z,x) \,] \\ &\vee [\, P(x) \wedge \forall y \neg E(x,y) \,] . \end{aligned}$$

Lemma 10 and the above discussions yield :

Theorem 4 : For $k \geq 1$, there exist k-ary predicates which are not expressible in FO+IFP^k (FO+IFP^k+Suc) but are expressible in FO+GFP$^k[n^k]$ (FO+GFP^k+Suc$[n^k]$). ∎

References

[AU] A.V.Aho and J.D.Ullman, Universality of Data Retrieval Languages, Proc. 6th ACM-POPL, 1979.

[Be] C.Berge, Graphs and Hypergraphs, North-Holland, 1974.

[CH] A.Chandra and D.Harel, Structure and Complexity of Relational Queries, JCSS, Vol 25, No 1, 1982.

[Du] P.Dublish, Optimization and Expressibility of Relational Queries, Ph.D. Thesis, Dept. of Computer Science & Engg., IIT Delhi, July 1988.

[Eh] A.Ehrenfeucht, An Application of Games to the Completeness Problem for Formalized Theories, Fundamenta Mathematicae, Vol 49, 1961.

[En] H.Enderton, A Mathematical Introduction to Logic, Academic Press, 1972.

[Fr] R.Fraisse, Sur les Classifications des Systems de Relations, Publ. Sci. Univ. Alger I, 1954.

[Gu] Y.Gurevich, Towards Logic Tailored for Computational Complexity, Computation and Proof Theory, Lec. Notes in Mathematics, Vol 1104, Springer-Verlag, 1984.

[GS] Y.Gurevich and S.Shelah, Fixed-Point Extensions of First-Order Logic, Proc. of 26th IEEE-FOCS, 1985.

[Im1] N.Immerman, Number of Quantifiers is Better than Number of Tape Cells, JCSS, Vol 22, No 3, 1981.

[Im2] N.Immerman, Relational Queries Computable in Polynomial Time, Information and Control, Vol 68, 1986.