An Extension of Datalog for Graph Queries

M. Mazuran[1] E. Serra[2] C. Zaniolo[3]

Politecnico di Milano DEI[1]– mazuran@elet.polimi.it
University of Calabria DEIS[2]– eserra@deis.unical.it
University of California, Los Angeles UCLA[3]– zaniolo@cs.ucla.edu

June 26, 2012

Motivation

Motivation

Motivation

Motivation

- A resurgence of interest in Datalog in many research areas—in particular “Big Data”.

Motivation

- A resurgence of interest in Datalog in many research areas—in particular "Big Data".
- Parallel fixpoint-based computation of recursive predicates dovetails with MapReduce framework [Afrati et al.]

Motivation

- A resurgence of interest in Datalog in many research areas—in particular “Big Data”.
- Parallel fixpoint-based computation of recursive predicates dovetails with MapReduce framework [Afrati et al.]
- There are several limitation due to the fact that aggregates cannot be allowed in recursive definitions since they are non-monotone.

Motivation

- A resurgence of interest in Datalog in many research areas—in particular "Big Data".
- Parallel fixpoint-based computation of recursive predicates dovetails with MapReduce framework [Afrati et al.]
- There are several limitation due to the fact that aggregates cannot be allowed in recursive definitions since they are non-monotone.
- Most systems only support aggregates outside recursion, i.e., in programs that are stratified w.r.t. aggregates and negation.

Motivation

- A resurgence of interest in Datalog in many research areas—in particular "Big Data".
- Parallel fixpoint-based computation of recursive predicates dovetails with MapReduce framework [Afrati et al.]
- There are several limitation due to the fact that aggregates cannot be allowed in recursive definitions since they are non-monotone.
- Most systems only support aggregates outside recursion, i.e., in programs that are stratified w.r.t. aggregates and negation.
- Much previous work has not produced a general solution.

Motivation

- A resurgence of interest in Datalog in many research areas—in particular "Big Data".
- Parallel fixpoint-based computation of recursive predicates dovetails with MapReduce framework [Afrati et al.]
- There are several limitation due to the fact that aggregates cannot be allowed in recursive definitions since they are non-monotone.
- Most systems only support aggregates outside recursion, i.e., in programs that are stratified w.r.t. aggregates and negation.
- Much previous work has not produced a general solution.

Datalog^{FS}

Datalog^{FS}

Datalog^{FS}

From Datalog to Datalog^{FS}

Datalog

Datalog^{FS}

From Datalog to Datalog^{FS}

Datalog

- A program P is a finite set of rules.

Datalog^{FS}

From Datalog to $Datalog^{FS}$

Datalog

- A program P is a finite set of rules.
- A rule is of the form $A \leftarrow A_1, \ldots, A_m$ where A is the head and A_i is a body literal (can be negated or not).

$Datalog^{FS}$

From Datalog to Datalog^{FS}

Datalog

- A program P is a finite set of rules.
- A rule is of the form $A \leftarrow A_1, \ldots, A_m$ where A is the head and A_i is a body literal (can be negated or not).
- Stratified Negation.

Datalog^{FS}

From Datalog to Datalog^{FS}

Datalog

- A program P is a finite set of rules.
- A rule is of the form $A \leftarrow A_1, \ldots, A_m$ where A is the head and A_i is a body literal (can be negated or not).
- Stratified Negation.
- Reachability Example

$$\begin{aligned} \mathtt{path(X,Y)} &\leftarrow \mathtt{arc(X,Y)} \\ \mathtt{path(X,Y)} &\leftarrow \mathtt{path(X,Z), path(Z,Y).} \end{aligned}$$

Datalog^{FS}

From Datalog to Datalog^{FS}

Datalog

- A program P is a finite set of rules.
- A rule is of the form $A \leftarrow A_1, \ldots, A_m$ where A is the head and A_i is a body literal (can be negated or not).
- Stratified Negation.
- Reachability Example

$$\begin{aligned} \mathtt{path(X, Y)} \leftarrow & \quad \mathtt{arc(X, Y)} \\ \mathtt{path(X, Y)} \leftarrow & \quad \mathtt{path(X, Z), path(Z, Y)}. \end{aligned}$$

Datalog^{FS}

- An extended Datalog that allows reasoning about the number of distinct occurrences satisfying a conjunction of goals

From Datalog to Datalog^{FS}

Datalog

- A program P is a finite set of rules.
- A rule is of the form $A \leftarrow A_1, \ldots, A_m$ where A is the head and A_i is a body literal (can be negated or not).
- Stratified Negation.
- Reachability Example

$$\begin{aligned} \mathtt{path(X,Y)} \leftarrow &\quad \mathtt{arc(X,Y)} \\ \mathtt{path(X,Y)} \leftarrow &\quad \mathtt{path(X,Z), path(Z,Y).} \end{aligned}$$

Datalog^{FS}

- An extended Datalog that allows reasoning about the number of distinct occurrences satisfying a conjunction of goals
- It adds 2 special constructs that appear as body literals:

From Datalog to Datalog^{FS}

Datalog

- A program P is a finite set of rules.
- A rule is of the form $A \leftarrow A_1, \ldots, A_m$ where A is the head and A_i is a body literal (can be negated or not).
- Stratified Negation.
- Reachability Example

$$\begin{aligned} \texttt{path}(\texttt{X}, \texttt{Y}) \leftarrow & \quad \texttt{arc}(\texttt{X}, \texttt{Y}) \\ \texttt{path}(\texttt{X}, \texttt{Y}) \leftarrow & \quad \texttt{path}(\texttt{X}, \texttt{Z}), \texttt{path}(\texttt{Z}, \texttt{Y}). \end{aligned}$$

Datalog^{FS}

- An extended Datalog that allows reasoning about the number of distinct occurrences satisfying a conjunction of goals
- It adds 2 special constructs that appear as body literals:
  1. Frequency Support goal (FS goal).

From Datalog to Datalog^{FS}

Datalog

- A program P is a finite set of rules.
- A rule is of the form $A \leftarrow A_1, \ldots, A_m$ where A is the head and A_i is a body literal (can be negated or not).
- Stratified Negation.
- Reachability Example

$$\begin{aligned} \mathtt{path(X,Y)} \leftarrow & \quad \mathtt{arc(X,Y)} \\ \mathtt{path(X,Y)} \leftarrow & \quad \mathtt{path(X,Z), path(Z,Y)}. \end{aligned}$$

Datalog^{FS}

- An extended Datalog that allows reasoning about the number of distinct occurrences satisfying a conjunction of goals
- It adds 2 special constructs that appear as body literals:
  1. Frequency Support goal (FS goal).
  2. Final-FS goal: derived from the first.

From Datalog to Datalog^{FS}

Datalog

- A program P is a finite set of rules.
- A rule is of the form $A \leftarrow A_1, \ldots, A_m$ where A is the head and A_i is a body literal (can be negated or not).
- Stratified Negation.
- Reachability Example

$$\begin{aligned} \mathtt{path(X,Y)} &\leftarrow \quad \mathtt{arc(X,Y)} \\ \mathtt{path(X,Y)} &\leftarrow \quad \mathtt{path(X,Z), path(Z,Y).} \end{aligned}$$

Datalog^{FS}

- An extended Datalog that allows reasoning about the number of distinct occurrences satisfying a conjunction of goals
- It adds 2 special constructs that appear as body literals:
  1. Frequency Support goal (FS goal).
  2. Final-FS goal: derived from the first.
- and define a new type of predicates called Multi-Occuring Predicates.

Datalog^{FS} constructs

Frequency Support goal

Final-FS goal

Datalog^{FS} constructs

Frequency Support goal

- $\mathtt{K} : [\mathtt{Bexpr}(\mathtt{X}, \mathtt{Y})]$ where:
  - $\mathtt{K}$ is a positive integer variable and
  - $\mathtt{Bexpr}(\mathtt{X}, \mathtt{Y})$ is a conjunction of positive literals with variables X and Y.

Final-FS goal

Datalog^{FS} constructs

Frequency Support goal

- $\mathtt{K} : [\mathtt{Bexpr}(\mathtt{X}, \mathtt{Y})]$ where:
  - $\mathtt{K}$ is a positive integer variable and
  - $\mathtt{Bexpr}(\mathtt{X}, \mathtt{Y})$ is a conjunction of positive literals with variables X and Y.
- Example about friends that will come to the party.

$$\mathtt{willcome}(\mathtt{X}) \leftarrow \mathtt{sure}(\mathtt{X}).$$
$$\mathtt{willcome}(\mathtt{X}) \leftarrow 3 : [\mathtt{friend}(\mathtt{X}, \mathtt{Y}), \mathtt{willcome}(\mathtt{Y})].$$

Final-FS goal

Datalog^{FS} constructs

Frequency Support goal

- $\mathtt{K} : [\mathtt{Bexpr}(\mathtt{X}, \mathtt{Y})]$ where:
    - $\mathtt{K}$ is a positive integer variable and
    - $\mathtt{Bexpr}(\mathtt{X}, \mathtt{Y})$ is a conjunction of positive literals with variables X and Y.
- Example about friends that will come to the party.

$$\mathtt{willcome}(\mathtt{X}) \leftarrow \mathtt{sure}(\mathtt{X}).$$
$$\mathtt{willcome}(\mathtt{X}) \leftarrow 3 : [\mathtt{friend}(\mathtt{X}, \mathtt{Y}), \mathtt{willcome}(\mathtt{Y})].$$

- Semantics: there exist at least $\mathtt{K}$ assignments of variables $\mathtt{Y}$ that satisfy the conjunction $\mathtt{Bexpr}(\mathtt{X}, \mathtt{Y})$ $(\mathtt{friend}(\mathtt{X}, \mathtt{Y}), \mathtt{willcome}(\mathtt{Y}))$.

Final-FS goal

$Datalog^{FS}$ constructs

Frequency Support goal

- $\texttt{K} : [\texttt{Bexpr}(\texttt{X}, \texttt{Y})]$ where:
  - $\texttt{K}$ is a positive integer variable and
  - $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ is a conjunction of positive literals with variables X and Y.
- Example about friends that will come to the party.

$$\begin{aligned} \texttt{willcome}(\texttt{X}) \leftarrow & \quad \texttt{sure}(\texttt{X}). \\ \texttt{willcome}(\texttt{X}) \leftarrow & \quad 3 : [\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y})]. \end{aligned}$$

- Semantics: there exist at least $\texttt{K}$ assignments of variables $\texttt{Y}$ that satisfy the conjunction $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ ($\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y})$).
- It is monotone and can be used in recursion.

Final-FS goal

Datalog^{FS} constructs

Frequency Support goal

- $\texttt{K} : [\texttt{Bexpr}(\texttt{X}, \texttt{Y})]$ where:
  - $\texttt{K}$ is a positive integer variable and
  - $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ is a conjunction of positive literals with variables X and Y.
- Example about friends that will come to the party.

$$\begin{aligned}\texttt{willcome}(\texttt{X}) \leftarrow & \quad \texttt{sure}(\texttt{X}).\\ \texttt{willcome}(\texttt{X}) \leftarrow & \quad 3 : [\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y})].\end{aligned}$$

- Semantics: there exist at least $\texttt{K}$ assignments of variables $\texttt{Y}$ that satisfy the conjunction $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ $(\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y}))$.
- It is monotone and can be used in recursion.

Final-FS goal

- $\texttt{K} = ! [\texttt{Bexpr}(\texttt{X}, \texttt{Y})]$

Datalog^{FS} constructs

Frequency Support goal

- $\texttt{K} : [\texttt{Bexpr}(\texttt{X}, \texttt{Y})]$ where:
  - $\texttt{K}$ is a positive integer variable and
  - $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ is a conjunction of positive literals with variables X and Y.
- Example about friends that will come to the party.

$$\texttt{willcome}(\texttt{X}) \leftarrow \quad \texttt{sure}(\texttt{X}).$$
$$\texttt{willcome}(\texttt{X}) \leftarrow \quad 3 : [\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y})].$$

- Semantics: there exist at least $\texttt{K}$ assignments of variables $\texttt{Y}$ that satisfy the conjunction $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ $(\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y}))$.
- It is monotone and can be used in recursion.

Final-FS goal

- $\texttt{K} =! [\texttt{Bexpr}(\texttt{X}, \texttt{Y})]$
- Example about person that has exactly 10 friends:

$$\texttt{p}(\texttt{X}) \leftarrow \quad 10 =! [\texttt{friend}(\texttt{X}, \texttt{Y})].$$

Datalog^{FS} constructs

Frequency Support goal

- $\texttt{K} : [\texttt{Bexpr}(\texttt{X}, \texttt{Y})]$ where:
  - $\texttt{K}$ is a positive integer variable and
  - $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ is a conjunction of positive literals with variables X and Y.
- Example about friends that will come to the party.

$$\texttt{willcome}(\texttt{X}) \leftarrow \texttt{sure}(\texttt{X}).$$
$$\texttt{willcome}(\texttt{X}) \leftarrow 3 : [\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y})].$$

- Semantics: there exist at least $\texttt{K}$ assignments of variables $\texttt{Y}$ that satisfy the conjunction $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ $(\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y}))$.
- It is monotone and can be used in recursion.

Final-FS goal

- $\texttt{K} =! [\texttt{Bexpr}(\texttt{X}, \texttt{Y})]$
- Example about person that has exactly 10 friends:

$$\texttt{p}(\texttt{X}) \leftarrow 10 =! [\texttt{friend}(\texttt{X}, \texttt{Y})].$$

- Semantics by using FS-goal and Negation:

$$\texttt{K} : [\texttt{Bexpr}(\texttt{X}, \texttt{Y})], \neg \texttt{K} + 1 : [\texttt{Bexpr}(\texttt{X}, \texttt{Y})].$$

Datalog^{FS} constructs

Frequency Support goal

- $\texttt{K} : [\texttt{Bexpr}(\texttt{X}, \texttt{Y})]$ where:
  - $\texttt{K}$ is a positive integer variable and
  - $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ is a conjunction of positive literals with variables X and Y.
- Example about friends that will come to the party.

$$\begin{aligned} \texttt{willcome}(\texttt{X}) &\leftarrow \quad \texttt{sure}(\texttt{X}). \\ \texttt{willcome}(\texttt{X}) &\leftarrow \quad 3 : [\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y})]. \end{aligned}$$

- Semantics: there exist at least $\texttt{K}$ assignments of variables $\texttt{Y}$ that satisfy the conjunction $\texttt{Bexpr}(\texttt{X}, \texttt{Y})$ $(\texttt{friend}(\texttt{X}, \texttt{Y}), \texttt{willcome}(\texttt{Y}))$.
- It is monotone and can be used in recursion.

Final-FS goal

- $\texttt{K} =! [\texttt{Bexpr}(\texttt{X}, \texttt{Y})]$
- Example about person that has exactly 10 friends:

$$\texttt{p}(\texttt{X}) \leftarrow \quad 10 =! [\texttt{friend}(\texttt{X}, \texttt{Y})].$$

- Semantics by using FS-goal and Negation:

$$\texttt{K} : [\texttt{Bexpr}(\texttt{X}, \texttt{Y})], \neg \texttt{K} + 1 : [\texttt{Bexpr}(\texttt{X}, \texttt{Y})].$$

- It is not monotone and requires stratified negation.

Multi-occurring predicates

Multi-occurring predicates

Multi-occurring predicates

Multi-occurring predicates

- $\texttt{m} - \texttt{predicate}(\texttt{x}) : \texttt{k}$

Multi-occurring predicates

Multi-occurring predicates

- $\mathtt{m - predicate(x) : k}$
- Semantics: the x value has k occurrences.

Multi-occurring predicates

Multi-occurring predicates

- $\mathtt{m-predicate(x)}:\mathtt{k}$
- Semantics: the x value has k occurrences.
- Facts Examples

$$\mathtt{ref}(''\mathtt{Bob2012}''):6. \qquad \mathtt{ref}(''\mathtt{Bob2012}''):4.$$
$$\mathtt{ref}(''\mathtt{Bob2012}'',\mathtt{journals}):6. \qquad \mathtt{ref}(''\mathtt{Bob2012}'',\mathtt{others}):4.$$

Multi-occurring predicates

Multi-occurring predicates

- $\mathtt{m - predicate(x) : k}$
- Semantics: the x value has k occurrences.
- Facts Examples

$$\mathtt{ref("Bob2012") : 6.} \qquad \mathtt{ref("Bob2012") : 4.}$$
$$\mathtt{ref("Bob2012", journals) : 6.} \qquad \mathtt{ref("Bob2012", others) : 4.}$$

- Rule example

$$\mathtt{ref("Bob2012") : 6.}$$
$$\mathtt{tref(Author) : N} \leftarrow \quad \mathtt{N : [author(Author, Pno), ref(Pno)].}$$

Multi-occurring predicates

Multi-occurring predicates

- $\texttt{m} - \texttt{predicate}(\texttt{x}) : \texttt{k}$
- Semantics: the x value has k occurrences.
- Facts Examples

$$\texttt{ref}(\text{"}\texttt{Bob2012}\text{"}) : 6. \qquad \texttt{ref}(\text{"}\texttt{Bob2012}\text{"}) : 4.$$

$$\texttt{ref}(\text{"}\texttt{Bob2012}\text{"}, \texttt{journals}) : 6. \qquad \texttt{ref}(\text{"}\texttt{Bob2012}\text{"}, \texttt{others}) : 4.$$

- Rule example

$$\texttt{ref}(\text{"}\texttt{Bob2012}\text{"}) : 6.$$

$$\texttt{tref}(\texttt{Author}) : \texttt{N} \leftarrow \quad \texttt{N} : [\texttt{author}(\texttt{Author}, \texttt{Pno}), \texttt{ref}(\texttt{Pno})].$$

$$total\_ref(A) = \sum_{Pno \in paper(A)} reference(Pno)$$

Multi-occurring predicates

Multi-occurring predicates

- $\mathtt{m-predicate(x) : k}$
- Semantics: the x value has k occurrences.
- Facts Examples

$$\mathtt{ref("Bob2012") : 6.} \qquad \mathtt{ref("Bob2012") : 4.}$$
$$\mathtt{ref("Bob2012", journals) : 6.} \qquad \mathtt{ref("Bob2012", others) : 4.}$$

- Rule example

$$\mathtt{ref("Bob2012") : 6.}$$
$$\mathtt{tref(Author) : N} \leftarrow \quad \mathtt{N : [author(Author, Pno), ref(Pno)].}$$

- Semantics by using Datalog.

$$\mathtt{ref("Bob2012", J)} \leftarrow \quad \mathtt{lessthan(J, 6).}$$
$$\mathtt{tref(Author, N)} \leftarrow \quad \mathtt{N : [author(Author, Pno), ref(Pno, J)].}$$

$$\mathtt{lessthan(1, K)} \leftarrow \quad \mathtt{K} \geq \mathtt{1.}$$
$$\mathtt{lessthan(J1, K)} \leftarrow \quad \mathtt{lessthan(J, K), K > J, J1 = J + 1.}$$

Applications

Applications

Some Examples

Some Examples

Example

Number of paths between two nodes in a Direct Acyclic Graph (DAG).

$$\begin{aligned} \mathtt{path}(\mathtt{X},\mathtt{Y}) : 1 &\leftarrow \quad \mathtt{edge}(\mathtt{X},\mathtt{Y}) \\ \mathtt{path}(\mathtt{X},\mathtt{Y}) : \mathtt{K} &\leftarrow \quad \mathtt{K}\!:\![\mathtt{path}(\mathtt{X},\mathtt{Z}),\mathtt{path}(\mathtt{Z},\mathtt{Y})]. \end{aligned}$$

Some Examples

Example

Number of paths between two nodes in a Direct Acyclic Graph (DAG)

$$n\_path(X, Y) = \sum_{Z \neq X, Y} n\_path(X, Z) \times n\_path(Z, Y)$$

Some Examples

Example

Number of paths between two nodes in a Direct Acyclic Graph (DAG).

$$\begin{aligned}\texttt{path}(\texttt{X},\texttt{Y}) : 1 &\leftarrow \quad \texttt{edge}(\texttt{X},\texttt{Y})\\ \texttt{path}(\texttt{X},\texttt{Y}) : \texttt{K} &\leftarrow \quad \texttt{K} : [\texttt{path}(\texttt{X},\texttt{Z}), \texttt{path}(\texttt{Z},\texttt{Y})].\end{aligned}$$

Example

Reachability in a directed hyper graph.
A hyperedge $(\{a, b\}, c)$ is represented by the following facts:

$$\texttt{hedgeS}(1,\texttt{a}).\ \ \texttt{hedgeS}(1,\texttt{b}).\ \ \texttt{hedgeT}(1,\texttt{c}).$$
$$\texttt{node}(\texttt{a}).\ \ \texttt{node}(\texttt{b}).\ \ \texttt{node}(\texttt{c}).$$

$$\begin{aligned}\texttt{reach}(\texttt{X},\texttt{X}) &\leftarrow \quad \texttt{node}(\texttt{X}).\\ \texttt{reach}(\texttt{X},\texttt{Y}) &\leftarrow \quad \texttt{K} : [\texttt{hedgeS}(\texttt{ID},\texttt{Z}), \texttt{reach}(\texttt{X},\texttt{Z})],\\ &\qquad \texttt{hedgeT}(\texttt{ID},\texttt{Y}), \texttt{K} = ![\texttt{hedgeS}(\texttt{ID}, \_)].\end{aligned}$$

Some Examples

Example

Number of paths between two nodes in a Direct Acyclic Graph (DAG).

$$\begin{aligned} \mathtt{path(X,Y)} : 1 &\leftarrow \mathtt{edge(X,Y)} \\ \mathtt{path(X,Y)} : \mathtt{K} &\leftarrow \mathtt{K}{:}[\mathtt{path(X,Z)}, \mathtt{path(Z,Y)}]. \end{aligned}$$

Example

Reachability in a directed hyper graph.
A hyperedge $(\{a, b\}, c)$ is represented by the following facts:

$$\mathtt{hedgeS(1,a).}\ \ \mathtt{hedgeS(1,b).}\ \ \mathtt{hedgeT(1,c).}$$
$$\mathtt{node(a).}\ \ \mathtt{node(b).}\ \ \mathtt{node(c).}$$

$$\begin{aligned} \mathtt{reach(X,X)} &\leftarrow \mathtt{node(X)}. \\ \mathtt{reach(X,Y)} &\leftarrow \mathtt{K} : [\mathtt{hedgeS(ID,Z)}, \mathtt{reach(X,Z)}], \\ &\quad\ \mathtt{hedgeT(ID,Y)}, \mathtt{K} = ![\mathtt{hedgeS(ID,\_)}]. \end{aligned}$$

- *Bill of Material*

Some Examples

Example

Number of paths between two nodes in a Direct Acyclic Graph (DAG).

$$\begin{aligned} \texttt{path}(\texttt{X},\texttt{Y}) : 1 &\leftarrow \quad \texttt{edge}(\texttt{X},\texttt{Y}) \\ \texttt{path}(\texttt{X},\texttt{Y}) : \texttt{K} &\leftarrow \quad \texttt{K} : [\texttt{path}(\texttt{X},\texttt{Z}), \texttt{path}(\texttt{Z},\texttt{Y})]. \end{aligned}$$

Example

Reachability in a directed hyper graph.
A hyperedge $(\{a, b\}, c)$ is represented by the following facts:

$$\texttt{hedgeS}(1,\texttt{a}).\quad \texttt{hedgeS}(1,\texttt{b}).\quad \texttt{hedgeT}(1,\texttt{c}).$$
$$\texttt{node}(\texttt{a}).\quad \texttt{node}(\texttt{b}).\quad \texttt{node}(\texttt{c}).$$

$$\begin{aligned} \texttt{reach}(\texttt{X},\texttt{X}) &\leftarrow \quad \texttt{node}(\texttt{X}). \\ \texttt{reach}(\texttt{X},\texttt{Y}) &\leftarrow \quad \texttt{K} : [\texttt{hedgeS}(\texttt{ID},\texttt{Z}), \texttt{reach}(\texttt{X},\texttt{Z})], \\ &\qquad \texttt{hedgeT}(\texttt{ID},\texttt{Y}), \texttt{K} =! [\texttt{hedgeS}(\texttt{ID}, \_)]. \end{aligned}$$

- *Bill of Material*
- *Path of minimum and maximum cost (more probable path)*

Some Examples

Example

Number of paths between two nodes in a Direct Acyclic Graph (DAG).

$$\begin{aligned} \mathtt{path}(\mathtt{X},\mathtt{Y}) : 1 &\leftarrow \mathtt{edge}(\mathtt{X},\mathtt{Y}) \\ \mathtt{path}(\mathtt{X},\mathtt{Y}) : \mathtt{K} &\leftarrow \mathtt{K}\!:\![\mathtt{path}(\mathtt{X},\mathtt{Z}),\mathtt{path}(\mathtt{Z},\mathtt{Y})]. \end{aligned}$$

Example

Reachability in a directed hyper graph.
A hyperedge $(\{a, b\}, c)$ is represented by the following facts:

$$\mathtt{hedgeS}(1,\mathtt{a}).\ \ \mathtt{hedgeS}(1,\mathtt{b}).\ \ \mathtt{hedgeT}(1,\mathtt{c}).$$
$$\mathtt{node}(\mathtt{a}).\ \ \mathtt{node}(\mathtt{b}).\ \ \mathtt{node}(\mathtt{c}).$$

$$\begin{aligned} \mathtt{reach}(\mathtt{X},\mathtt{X}) &\leftarrow \mathtt{node}(\mathtt{X}). \\ \mathtt{reach}(\mathtt{X},\mathtt{Y}) &\leftarrow \mathtt{K} : [\mathtt{hedgeS}(\mathtt{ID},\mathtt{Z}),\mathtt{reach}(\mathtt{X},\mathtt{Z})], \\ &\quad\ \mathtt{hedgeT}(\mathtt{ID},\mathtt{Y}),\mathtt{K} =![\mathtt{hedgeS}(\mathtt{ID},\_)]. \end{aligned}$$

- *Bill of Material*
- *Path of minimum and maximum cost (more probable path)*
- *Dynamic Programming (Knapsak, Viterbi, ...)*
- *........*

Jackson-Yariv Diffusion Model

Jackson-Yariv Diffusion Model

- Models how social structures influence the spread of behavior and trends in social networks (diffusion of innovations, behaviors, information and political movements).

Jackson-Yariv Diffusion Model

- Models how social structures influence the spread of behavior and trends in social networks (diffusion of innovations, behaviors, information and political movements).
- Given a graph $G = \langle V, E \rangle$ where V are agents and E are edges denoting their relationships, each agent has a default behavior A and decides on whether to adopt a new behavior B based on:

Jackson-Yariv Diffusion Model

- Models how social structures influence the spread of behavior and trends in social networks (diffusion of innovations, behaviors, information and political movements).
- Given a graph $G = \langle V, E \rangle$ where V are agents and E are edges denoting their relationships, each agent has a default behavior A and decides on whether to adopt a new behavior B based on:

$$B_i = bc_i * g(\Gamma_i) * \frac{1}{\Gamma_i} * \sum_{(j,i) \in E} B_j, \ \forall i \in V$$

Jackson-Yariv Diffusion Model

- Models how social structures influence the spread of behavior and trends in social networks (diffusion of innovations, behaviors, information and political movements).
- Given a graph $G = \langle V, E \rangle$ where V are agents and E are edges denoting their relationships, each agent has a default behavior A and decides on whether to adopt a new behavior B based on:

$$B_i = bc_i * g(\Gamma_i) * \frac{1}{\Gamma_i} * \sum_{(j,i) \in E} B_j, \forall i \in V$$

- a constant bc to denote how much an agent is susceptible to make a change.

Jackson-Yariv Diffusion Model

- Models how social structures influence the spread of behavior and trends in social networks (diffusion of innovations, behaviors, information and political movements).
- Given a graph $G = \langle V, E \rangle$ where V are agents and E are edges denoting their relationships, each agent has a default behavior A and decides on whether to adopt a new behavior B based on:

$$B_i = bc_i * g(\Gamma_i) * \frac{1}{\Gamma_i} * \sum_{(j,i) \in E} B_j, \ \forall i \in V$$

- a constant bc to denote how much an agent is susceptible to make a change.
- a function g denoting how much the number of neighbors influence the change.

Jackson-Yariv Diffusion Model

- Models how social structures influence the spread of behavior and trends in social networks (diffusion of innovations, behaviors, information and political movements).
- Given a graph $G = \langle V, E \rangle$ where V are agents and E are edges denoting their relationships, each agent has a default behavior A and decides on whether to adopt a new behavior B based on:

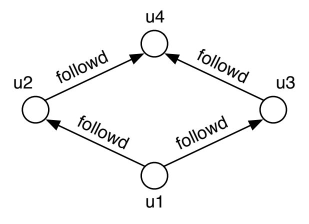
$$B_i = bc_i * g(\Gamma_i) * \frac{1}{\Gamma_i} * \sum_{(j,i) \in E} B_j, \ \forall i \in V$$

- a constant bc to denote how much an agent is susceptible to make a change.
- a function g denoting how much the number of neighbors influence the change.
- the percentage of neighbors that changed behavior.

Jackson-Yariv Diffusion Model

Jackson-Yariv Diffusion Model

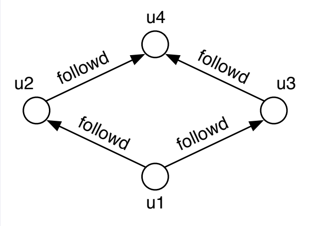
$$\begin{aligned}
\texttt{coeff}(\texttt{X},\texttt{C}) \leftarrow \quad & \texttt{K2} =![\texttt{followd}(\texttt{Y},\texttt{X})], \texttt{bc}(\texttt{X},\texttt{V1}),\\
& \texttt{g}(\texttt{K2},\texttt{V3}), \texttt{C} = \texttt{V1} * \texttt{V3}/\texttt{K2}.\\
\texttt{b}(\texttt{X}) \leftarrow \quad & \texttt{source}(\texttt{X}).\\
\texttt{b}(\texttt{X}) \leftarrow \quad & \texttt{coeff}(\texttt{X},\texttt{C}), \texttt{K} \geq 1/\texttt{C}, \texttt{K} : [\texttt{followd}(\texttt{Y},\texttt{X}), \texttt{b}(\texttt{Y})].
\end{aligned}$$



$\texttt{follwd}(u_1, u_2)$.	$\texttt{bc}(u_1, 1)$.	$\texttt{g}(1, 1.2)$.
$\texttt{follwd}(u_1, u_3)$.	$\texttt{bc}(u_2, 0.9)$.	$\texttt{g}(2, 2.3)$.
$\texttt{follwd}(u_2, u_4)$.	$\texttt{bc}(u_3, 0.5)$.	
$\texttt{follwd}(u_3, u_4)$.	$\texttt{bc}(u_4, 1)$.	
$\texttt{source}(u_1)$.		

Jackson-Yariv Diffusion Model

$$\begin{aligned}
\mathtt{coeff}(\mathtt{X},\mathtt{C}) \leftarrow \quad & \mathtt{K2} = ![\mathtt{followd}(\mathtt{Y},\mathtt{X})], \mathtt{bc}(\mathtt{X},\mathtt{V1}),\\
& \mathtt{g}(\mathtt{K2},\mathtt{V3}), \mathtt{C} = \mathtt{V1} * \mathtt{V3}/\mathtt{K2}.\\
\mathtt{b}(\mathtt{X}) \leftarrow \quad & \mathtt{source}(\mathtt{X}).\\
\mathtt{b}(\mathtt{X}) \leftarrow \quad & \mathtt{coeff}(\mathtt{X},\mathtt{C}), \mathtt{K} \geq 1/\mathtt{C}, \mathtt{K} : [\mathtt{followd}(\mathtt{Y},\mathtt{X}), \mathtt{b}(\mathtt{Y})].
\end{aligned}$$



$$\begin{array}{lll}
\mathtt{follwd}(\mathtt{u}_1,\mathtt{u}_2). & \mathtt{bc}(\mathtt{u}_1,1). & \mathtt{g}(1,1.2).\\
\mathtt{follwd}(\mathtt{u}_1,\mathtt{u}_3). & \mathtt{bc}(\mathtt{u}_2,0.9). & \mathtt{g}(2,2.3).\\
\mathtt{follwd}(\mathtt{u}_2,\mathtt{u}_4). & \mathtt{bc}(\mathtt{u}_3,0.5). & \\
\mathtt{follwd}(\mathtt{u}_3,\mathtt{u}_4). & \mathtt{bc}(\mathtt{u}_4,1). & \\
\mathtt{source}(\mathtt{u}_1). & &
\end{array}$$

the program derives the following atoms:

$$\begin{array}{lll}
\mathtt{coeff}(\mathtt{u}_2,1.08), & \mathtt{coeff}(\mathtt{u}_3,0.6), & \mathtt{coeff}(\mathtt{u}_4,1.15),\\
\mathtt{b}(\mathtt{u}_1), & \mathtt{b}(\mathtt{u}_2), & \mathtt{b}(\mathtt{u}_4).
\end{array}$$

Markov chains

Markov chains

- A process that consists of a finite number of states

Markov chains

- A process that consists of a finite number of states
- The process starts in one state and then moves from one state to another

Markov chains

- A process that consists of a finite number of states
- The process starts in one state and then moves from one state to another
- Represented by the transition matrix W of $s \times s$ components where w_{ij} is the probability to go from state i to state j in one step

Markov chains

- A process that consists of a finite number of states
- The process starts in one state and then moves from one state to another
- Represented by the transition matrix W of $s \times s$ components where w_{ij} is the probability to go from state i to state j in one step
- Given P a vector of stabilized probabilities of cardinality s then for each component we have:

$$p_i = \sum_{j=1}^{s} w_{ji} \cdot p_j$$

this is the equilibrium condition expressed by the fixpoint equation:

$$P = W \cdot P$$

Markov chains

- A process that consists of a finite number of states
- The process starts in one state and then moves from one state to another
- Represented by the transition matrix W of $s \times s$ components where w_{ij} is the probability to go from state i to state j in one step
- Given P a vector of stabilized probabilities of cardinality s then for each component we have:

$$p_i = \sum_{j=1}^{s} w_{ji} \cdot p_j$$

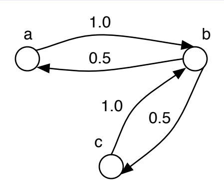
  this is the equilibrium condition expressed by the fixpoint equation:

$$P = W \cdot P$$

- Used for Page Rank.

Markov chains

Markov chains

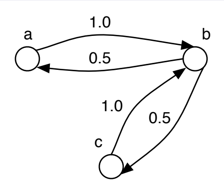


$$\mathtt{p\_st}(\mathtt{X}) : \mathtt{K} \leftarrow \quad \mathtt{K} : [\mathtt{p\_st}(\mathtt{Y}), \mathtt{w\_mat}(\mathtt{Y}, \mathtt{X})].$$

$\mathtt{w\_mat}(a, b) : 1.0.$
$\mathtt{w\_mat}(b, a) : 0.5.$
$\mathtt{w\_mat}(b, c) : 0.5.$
$\mathtt{w\_mat}(c, b) : 1.0.$
$\mathtt{p\_st}(a). \quad \mathtt{p\_st}(b). \quad \mathtt{p\_st}(c).$

where:

Markov chains



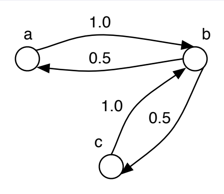
$$\mathtt{p\_st(X) : K} \leftarrow \quad \mathtt{K : [p\_st(Y), w\_mat(Y, X)]}.$$

$\mathtt{w\_mat(a, b) : 1.0.}$
$\mathtt{w\_mat(b, a) : 0.5.}$
$\mathtt{w\_mat(b, c) : 0.5.}$
$\mathtt{w\_mat(c, b) : 1.0.}$
$\mathtt{p\_st(a).} \quad \mathtt{p\_st(b).} \quad \mathtt{p\_st(c).}$

where:

- $\mathtt{p\_st(X) : K}$ *means the probability of staying in state X is K*

Markov chains



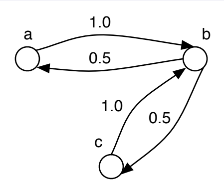
$$\texttt{p\_st}(\texttt{X}) : \texttt{K} \leftarrow \quad \texttt{K} : [\texttt{p\_st}(\texttt{Y}), \texttt{w\_mat}(\texttt{Y}, \texttt{X})].$$

$\texttt{w\_mat}(\texttt{a}, \texttt{b}) : 1.0.$
$\texttt{w\_mat}(\texttt{b}, \texttt{a}) : 0.5.$
$\texttt{w\_mat}(\texttt{b}, \texttt{c}) : 0.5.$
$\texttt{w\_mat}(\texttt{c}, \texttt{b}) : 1.0.$
$\texttt{p\_st}(\texttt{a}).\quad \texttt{p\_st}(\texttt{b}).\quad \texttt{p\_st}(\texttt{c}).$

where:

- $\texttt{p\_st}(\texttt{X}) : \texttt{K}$ *means the probability of staying in state X is K*
- $\texttt{w\_mat}(\texttt{Y}, \texttt{X}) : \texttt{W}$ *means the arc from Y to X has weight W*

Markov chains



$$\texttt{p\_st}(\texttt{X}) : \texttt{K} \leftarrow \quad \texttt{K} : [\texttt{p\_st}(\texttt{Y}), \texttt{w\_mat}(\texttt{Y}, \texttt{X})].$$

$\texttt{w\_mat}(\texttt{a}, \texttt{b}) : 1.0.$
$\texttt{w\_mat}(\texttt{b}, \texttt{a}) : 0.5.$
$\texttt{w\_mat}(\texttt{b}, \texttt{c}) : 0.5.$
$\texttt{w\_mat}(\texttt{c}, \texttt{b}) : 1.0.$
$\texttt{p\_st}(\texttt{a}). \quad \texttt{p\_st}(\texttt{b}). \quad \texttt{p\_st}(\texttt{c}).$

where:

- $\texttt{p\_st}(\texttt{X}) : \texttt{K}$ *means the probability of staying in state X is K*
- $\texttt{w\_mat}(\texttt{Y}, \texttt{X}) : \texttt{W}$ *means the arc from Y to X has weight W*

$$\texttt{p\_st}(\texttt{a}) : 0.25. \quad \texttt{p\_st}(\texttt{b}) : 0.5. \quad \texttt{p\_st}(\texttt{c}) : 0.25.$$

Computation and Opitmization

Computation and Opitmization

Computation and Opitmization

Computation and Opitmization

- Frequency Support Goal is monotone
  - Fixpoint Algorithm for positive programs.
  - Differential Fixpoint.
  - Magic Set.

Computation and Opitmization

- Frequency Support Goal is monotone
  - Fixpoint Algorithm for positive programs.
  - Differential Fixpoint.
  - Magic Set.
- Final-FS goal: its semantics is defined by using the negation. The stratified fixpoint for programs with stratified negation is used.

Computation and Opitmization

- Frequency Support Goal is monotone
  - Fixpoint Algorithm for positive programs.
  - Differential Fixpoint.
  - Magic Set.
- Final-FS goal: its semantics is defined by using the negation. The stratified fixpoint for programs with stratified negation is used.
- Multiplicity predicate: we only store the maximum value of multiplicity.

Conclusion and Future Work

Conclusion and Future Work

Conclusion

- We proposed a simple extension of Datalog, called Datalog^{FS}, with two new constructs (FS goal and Final-FS goal) and a new type of predicate (Multiplicity predicate).

Conclusion

- We proposed a simple extension of Datalog, called Datalog^{FS}, with two new constructs (FS goal and Final-FS goal) and a new type of predicate (Multiplicity predicate).
- Datalog^{FS} is more expressive then of stratified Datalog and stratified Aggregates.
  It allows the recursion with aggregate (FS goal).

Conclusion

- We proposed a simple extension of Datalog, called Datalog^{FS}, with two new constructs (FS goal and Final-FS goal) and a new type of predicate (Multiplicity predicate).
- Datalog^{FS} is more expressive then of stratified Datalog and stratified Aggregates.
  It allows the recursion with aggregate (FS goal).
- There are a lot of problems that can be solved by Datalog^{FS} with a simple and compact definition: Reachability Hyper Graph, Path of Minimum Cost, Diffusion Model, Page Rank ...

Conclusion

- We proposed a simple extension of Datalog, called Datalog^{FS}, with two new constructs (FS goal and Final-FS goal) and a new type of predicate (Multiplicity predicate).
- Datalog^{FS} is more expressive then of stratified Datalog and stratified Aggregates.
  It allows the recursion with aggregate (FS goal).
- There are a lot of problems that can be solved by Datalog^{FS} with a simple and compact definition: Reachability Hyper Graph, Path of Minimum Cost, Diffusion Model, Page Rank ...
- Datalog^{FS} uses the same efficient optimization used for Datalog.

Conclusion

- We proposed a simple extension of Datalog, called Datalog^{FS}, with two new constructs (FS goal and Final-FS goal) and a new type of predicate (Multiplicity predicate).
- Datalog^{FS} is more expressive then of stratified Datalog and stratified Aggregates.
  It allows the recursion with aggregate (FS goal).
- There are a lot of problems that can be solved by Datalog^{FS} with a simple and compact definition: Reachability Hyper Graph, Path of Minimum Cost, Diffusion Model, Page Rank ...
- Datalog^{FS} uses the same efficient optimization used for Datalog.
- Recent works show how it is possible to use parallel paradigms (Map Reduce) to improve Datalog's performances.

Conclusion

- We proposed a simple extension of Datalog, called Datalog^{FS}, with two new constructs (FS goal and Final-FS goal) and a new type of predicate (Multiplicity predicate).
- Datalog^{FS} is more expressive then of stratified Datalog and stratified Aggregates.
  It allows the recursion with aggregate (FS goal).
- There are a lot of problems that can be solved by Datalog^{FS} with a simple and compact definition: Reachability Hyper Graph, Path of Minimum Cost, Diffusion Model, Page Rank ...
- Datalog^{FS} uses the same efficient optimization used for Datalog.
- Recent works show how it is possible to use parallel paradigms (Map Reduce) to improve Datalog's performances.
  - Datalog^{FS} and in particular the multiplicity predicates can be also combined with such technologies.

Thank you!
Any question?

Compact way

Compact way of expressing counting goals:

- in Datalog^{FS}

$$\texttt{sixsubs}(\texttt{X}) \leftarrow \texttt{detective}(\texttt{X}), 6 : [\texttt{superior}(\texttt{X}, \texttt{Y})].$$

- in Datalog

$$\begin{aligned}\texttt{sixsubs}(\texttt{X}) \leftarrow\ & \texttt{detective}(\texttt{X}), \texttt{superior}(\texttt{X}, \texttt{Y1}), \\ & \texttt{superior}(\texttt{X}, \texttt{Y2}), \texttt{superior}(\texttt{X}, \texttt{Y3}), \\ & \texttt{superior}(\texttt{X}, \texttt{Y4}), \texttt{superior}(\texttt{X}, \texttt{Y5}), \\ & \texttt{superior}(\texttt{X}, \texttt{Y6}), \texttt{Y1} \neq \texttt{Y2}, \texttt{Y1} \neq \texttt{Y3}, \\ & \texttt{Y1} \neq \texttt{Y4}, \texttt{Y1} \neq \texttt{Y5}, \texttt{Y1} \neq \texttt{Y6}, \texttt{Y2} \neq \texttt{Y3}, \\ & \texttt{Y2} \neq \texttt{Y4}, \texttt{Y2} \neq \texttt{Y5}, \texttt{Y2} \neq \texttt{Y6}, \texttt{Y3} \neq \texttt{Y4}, \\ & \texttt{Y3} \neq \texttt{Y5}, \texttt{Y3} \neq \texttt{Y6}, \texttt{Y4} \neq \texttt{Y5}, \texttt{Y4} \neq \texttt{Y6}, \\ & \texttt{Y5} \neq \texttt{Y6}.\end{aligned}$$