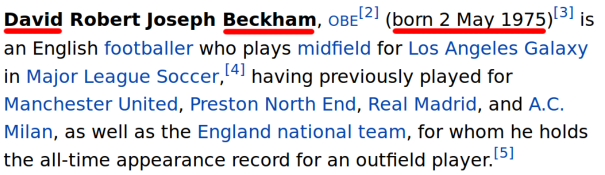
Resolving Temporal Conflicts in Inconsistent RDF Knowledge Bases

Maximilian Dylla Mauro Sozio Martin Theobald

Max Planck Institute for Informatics, Saarbrücken, Germany

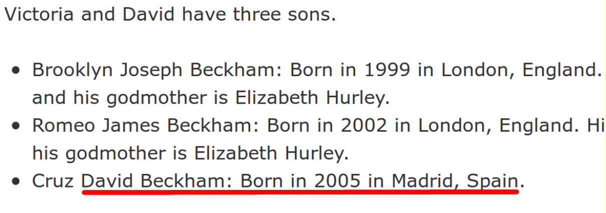
Motivation

http://en.wikipedia.org/wiki/David_beckham



David Robert Joseph Beckham, OBE[2] (born 2 May 1975)[3] is an English footballer who plays midfield for Los Angeles Galaxy in Major League Soccer,[4] having previously played for Manchester United, Preston North End, Real Madrid, and A.C. Milan, as well as the England national team, for whom he holds the all-time appearance record for an outfield player.[5]

http://marriage.about.com/od/sports/a/davidbeckham.htm



Victoria and David have three sons.

- Brooklyn Joseph Beckham: Born in 1999 in London, England. and his godmother is Elizabeth Hurley.
- Romeo James Beckham: Born in 2002 in London, England. Hi his godmother is Elizabeth Hurley.
- Cruz David Beckham: Born in 2005 in Madrid, Spain.

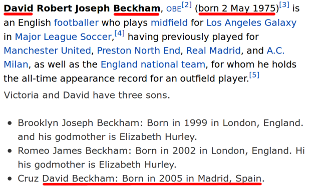
Extracting Facts

$$Facts \subset (Relation \times Entities \times Entities)$$

$$\text{Weight} \ : Facts \longrightarrow \mathbb{R}^{+}$$

$$\text{Time-Interval} \ : Facts \longrightarrow Intervals$$

Sources



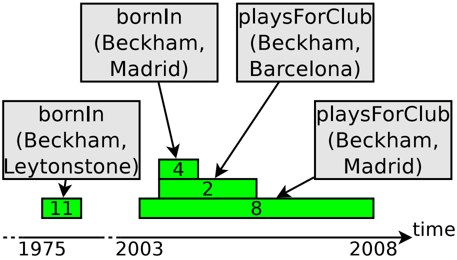
David Robert Joseph Beckham, OBE[2] (born 2 May 1975)[3] is an English footballer who plays midfield for Los Angeles Galaxy in Major League Soccer,[4] having previously played for Manchester United, Preston North End, Real Madrid, and A.C. Milan, as well as the England national team, for whom he holds the all-time appearance record for an outfield player.[5]

Victoria and David have three sons.

- Brooklyn Joseph Beckham: Born in 1999 in London, England. and his godmother is Elizabeth Hurley.
- Romeo James Beckham: Born in 2002 in London, England. Hi his godmother is Elizabeth Hurley.
- Cruz David Beckham: Born in 2005 in Madrid, Spain.

⇒

Noisy Facts



bornIn (Beckham, Madrid)
playsForClub (Beckham, Barcelona)
bornIn (Beckham, Leytonstone)
playsForClub (Beckham, Madrid)
4
2
11
8
time
1975
2003
2008

Constraints

Temporal Constraints:

- Precedence (*before*)
- Non-overlapping (*disjoint*)

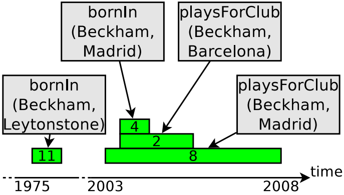
Non-temporal Constraint:

- Functional (*mutEx*)

Each Constraint:

- 2 relations from DB e.g. *bornIn*
- Both share a variable

Noisy Facts



bornIn (Beckham, Madrid)
playsForClub (Beckham, Barcelona)
bornIn (Beckham, Leytonstone)
playsForClub (Beckham, Madrid)
4
2
11
8
1975
2003
2008
time

Constraints

$$\begin{pmatrix} bornIn(p, l, t_1) \wedge \\ playsForClub(p, c, t_2) \end{pmatrix} \rightarrow before(t_1, t_2)$$

$$\begin{pmatrix} playsForClub(p, c_1, t_1) \wedge \\ playsForClub(p, c_2, t_2) \wedge \\ c_1 \neq c_2 \end{pmatrix} \rightarrow disjoint(t_1, t_2)$$

Answering Queries

Goal: Return only consistent Facts

1. Obtain consistent $F \subseteq Facts$

$$\max_{F \subseteq Facts} \sum_{f \in F} w(f)$$

   where F fullfills constraints

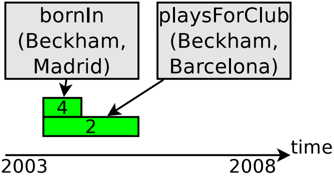
2. Answer query within F

- NP-hard

Query

$playsForClub(David\_Beckham, ?, ?)$

Facts



Constraint

$$\left(\begin{array}{c} bornIn(p, l, t_1) \wedge \\ playsForClub(p, c, t_2) \end{array} \right) \rightarrow before(t_1, t_2)$$

First Approach

Maximum Weight Independent Set

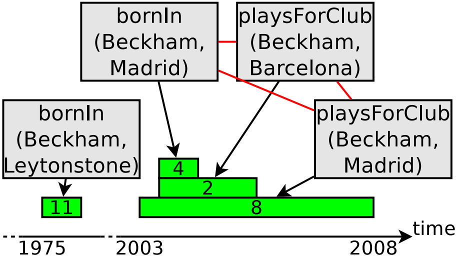
- Binary constraints
- NP-hard
- Heuristics in $\Omega(|\mathit{Facts}|^2)$

Constraints

$$\begin{pmatrix} \mathit{bornIn}(p, l, t_1) \wedge \\ \mathit{playsForClub}(p, c, t_2) \end{pmatrix} \\ \rightarrow \mathit{before}(t_1, t_2)$$

$$\begin{pmatrix} \mathit{playsForClub}(p, c_1, t_1) \wedge \\ \mathit{playsForClub}(p, c_2, t_2) \wedge \\ c_1 \neq c_2 \end{pmatrix} \\ \rightarrow \mathit{disjoint}(t_1, t_2)$$

Facts



bornIn (Beckham, Madrid)

playsForClub (Beckham, Barcelona)

bornIn (Beckham, Leytonstone)

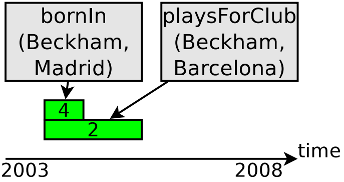
playsForClub (Beckham, Madrid)

Idea

Scheduling Problem

- Temporal constraints
- NP-hard
- Heuristics in $O(|Facts|\ log\ |Facts|)$

Facts

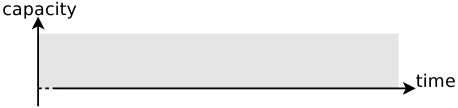


bornIn (Beckham, Madrid)

playsForClub (Beckham, Barcelona)

⇒

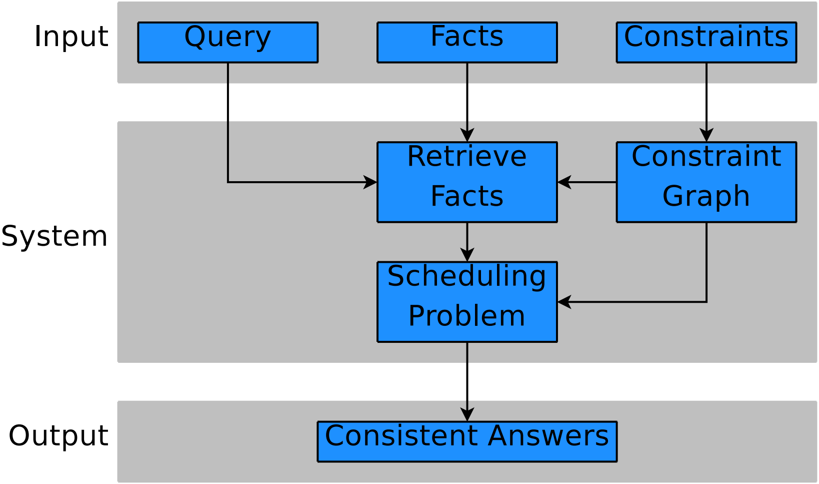
Scheduling Machine



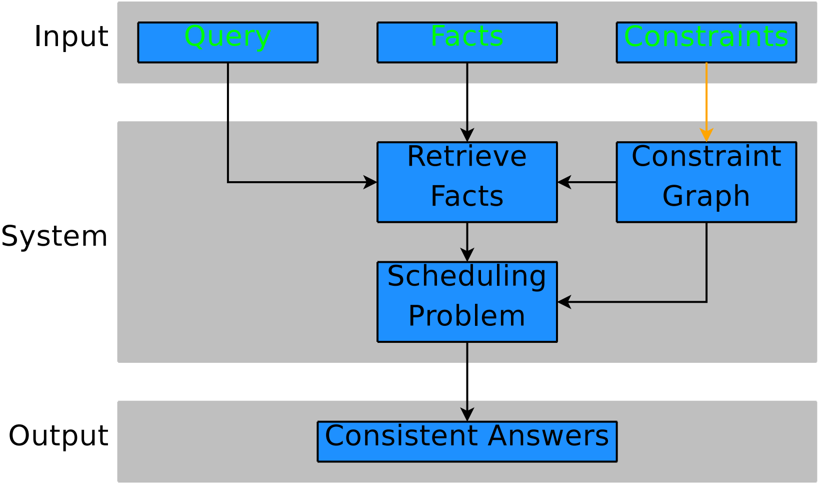
Constraint

$$\left(\begin{array}{c} bornIn(p, l, t_1)\wedge \\ playsForClub(p, c, t_2) \end{array}\right) \\ \rightarrow before(t_1, t_2)$$

Overview


Input
Query
Facts
Constraints
System
Retrieve Facts
Constraint Graph
Scheduling Problem
Output
Consistent Answers

Overview


Input
Query
Facts
Constraints
System
Retrieve Facts
Constraint Graph
Scheduling Problem
Output
Consistent Answers

Mapping

- *Relations → Vertices*
- *Constraints → Edges*

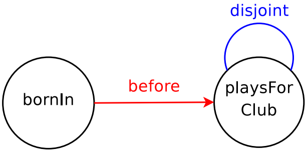
Constraints

$$\left(\begin{array}{l} bornIn(p, l, t_1)\wedge \\ playsForClub(p, c, t_2) \end{array}\right) \rightarrow before(t_1, t_2)$$

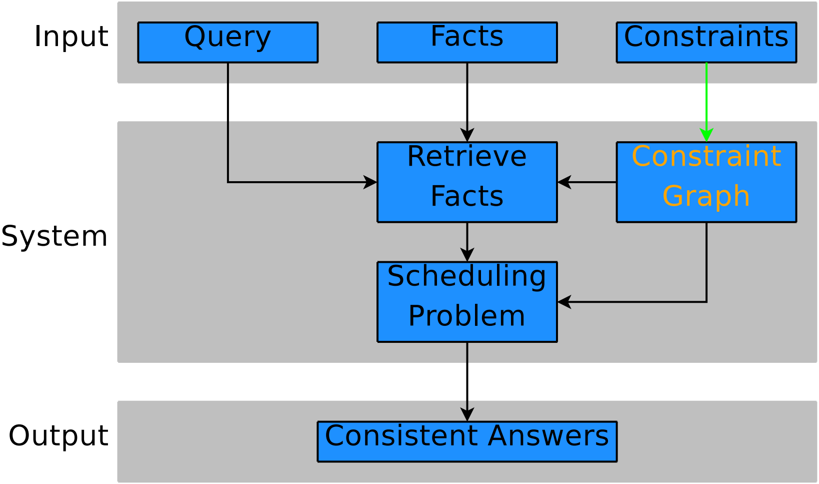
$$\left(\begin{array}{l} playsForClub(p, c_1, t_1)\wedge \\ playsForClub(p, c_2, t_2)\wedge \\ c_1 \neq c_2 \end{array}\right) \rightarrow disjoint(t_1, t_2)$$

$\Rightarrow$

Constraint Graph



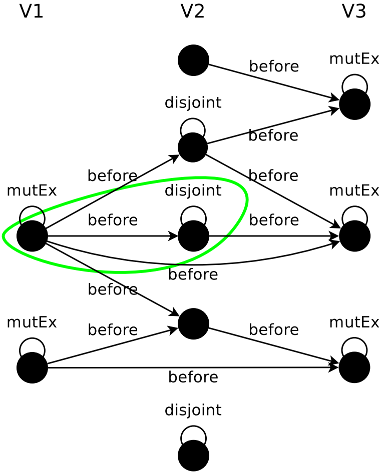
Overview


Input
Query
Facts
Constraints
System
Retrieve Facts
Constraint Graph
Scheduling Problem
Output
Consistent Answers

Constraint Graph

Constraints handled by Scheduling

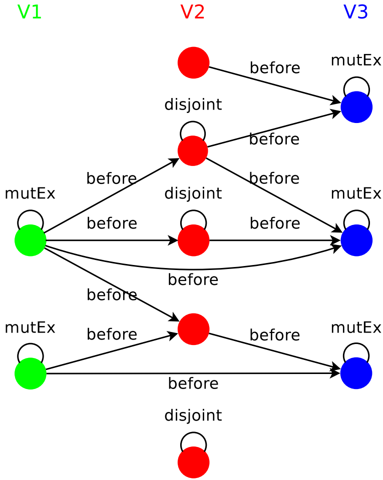
- Tripartide graph $(V_1 \cup V_2 \cup V_3, E)$
- $V_1 \cup V_3$ must have *mutEx* loops
- V_2 can have *disjoint* loops
- *before* can edges point:
  - from V_1 to $V_2 \cup V_3$
  - or from V_2 to V_3



Constraint Graph

Constraints handled by Scheduling

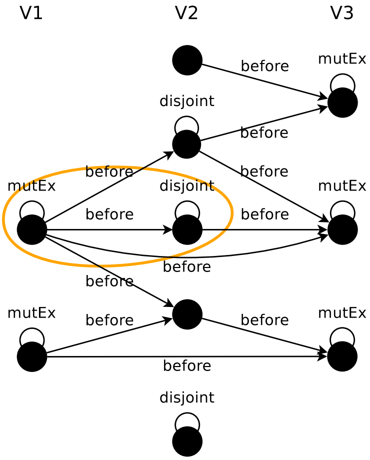
- Tripartide graph ($V_1 \cup V_2 \cup V_3, E$)
- $V_1 \cup V_3$ must have *mutEx* loops
- V_2 can have *disjoint* loops
- *before* can edges point:
  - from V_1 to $V_2 \cup V_3$
  - or from V_2 to V_3



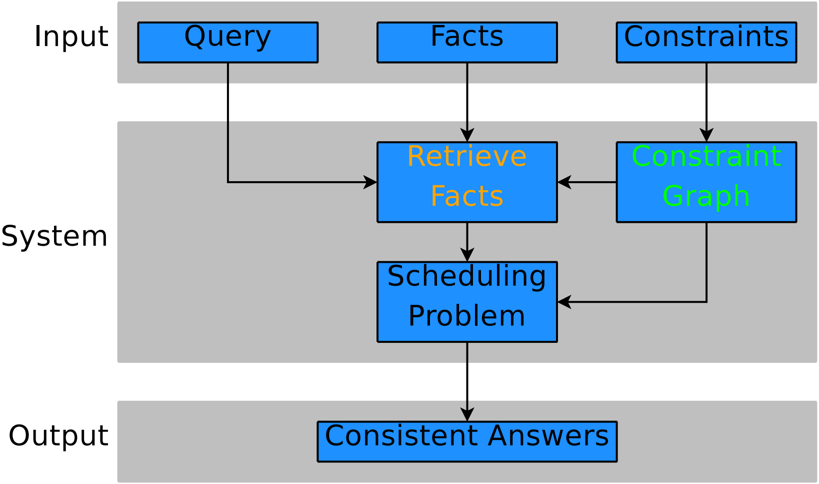
Constraint Graph

Constraints handled by Scheduling

- Tripartide graph $(V_1 \cup V_2 \cup V_3, E)$
- $V_1 \cup V_3$ must have *mutEx* loops
- V_2 can have *disjoint* loops
- *before* can edges point:
  - from V_1 to $V_2 \cup V_3$
  - or from V_2 to V_3



Overview


Input
Query
Facts
Constraints
System
Retrieve Facts
Constraint Graph
Scheduling Problem
Output
Consistent Answers

Retrieving Facts

Breadth-First Search

1. Start: Nodes matching Query
2. At each node:
    1. Retrieve Facts from DB
    2. If result $\neq \emptyset$: continue at not visited neighbours, pass argument

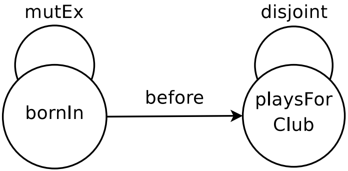
In General:

- For each shared Argument $\Rightarrow O(|Facts| \cdot |Constraints|)$

Query

$playsForClub(David\_Beckham, ?, ?)$

Constraint Graph



Retrieving Facts

Breadth-First Search

1. Start: Nodes matching Query
2. At each node:
    1. Retrieve Facts from DB
    2. If result $\neq \emptyset$:
       continue at not visited neighbours, pass argument

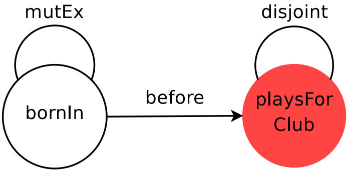
In General:

- For each shared Argument
  $\Rightarrow O(|Facts| \cdot |Constraints|)$

Query

$playsForClub(David\_Beckham, ?, ?)$

Constraint Graph



Retrieving Facts

Breadth-First Search

1. Start: Nodes matching Query
2. At each node:
   1. Retrieve Facts from DB
   2. If result $\neq \emptyset$: continue at not visited neighbours, pass argument

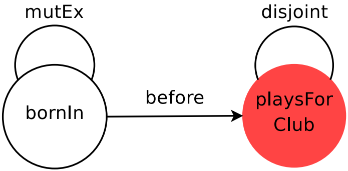
In General:

- For each shared Argument $\Rightarrow O(|Facts| \cdot |Constraints|)$

Query

$playsForClub(David\_Beckham, ?, ?)$

Constraint Graph



Select * from facts where rel=*playsForClub* and arg1=*David_Beckham*;

Retrieving Facts

Breadth-First Search

1. Start: Nodes matching Query
2. At each node:
    1. Retrieve Facts from DB
    2. If result $\neq \emptyset$:
       continue at not visited neighbours, pass argument

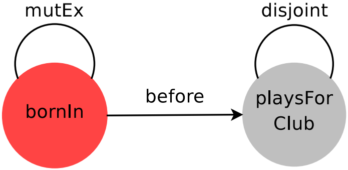
In General:

- For each shared Argument
  $\Rightarrow O(|Facts| \cdot |Constraints|)$

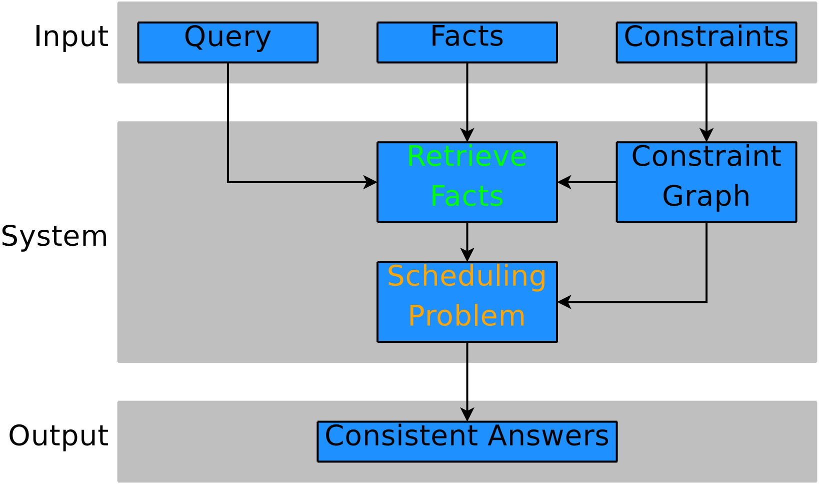
Query

$playsForClub(David\_Beckham, ?, ?)$

Constraint Graph



Overview


Input
Query
Facts
Constraints
System
Retrieve Facts
Constraint Graph
Scheduling Problem
Output
Consistent Answers

Scheduling

Facts $\longrightarrow$ *Scheduling Jobs*

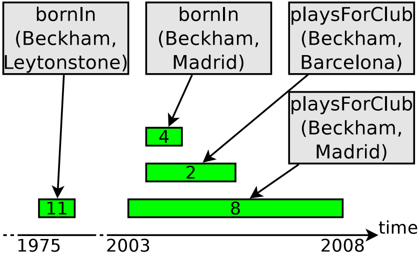
- Weight: Identical
- Capacity $\in [0, 1]$
- Begin: Identical or 0
- End: Identical or ∞

Depending on constraint graph

In General:

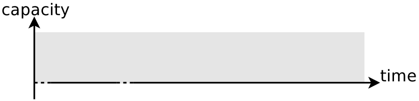
- Several Scheduling Machines
- $O(|F| log|F| + |F||machines|)$ [Bar-Noy et al., 2001]

Facts:

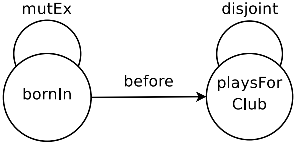


bornIn (Beckham, Leytonstone)
bornIn (Beckham, Madrid)
playsForClub (Beckham, Barcelona)
playsForClub (Beckham, Madrid)
4
2
11
8
time
1975
2003
2008

Scheduling Machine:



Constraint Graph:



Scheduling

Facts → *Scheduling Jobs*

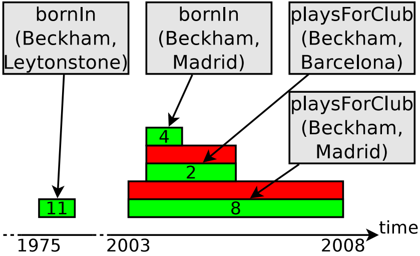
- Weight: Identical
- Capacity $\in [0, 1]$
- Begin: Identical or 0
- End: Identical or ∞

Depending on constraint graph

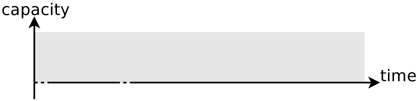
In General:

- Several Scheduling Machines
- $O(|F| log |F| + |F||machines|)$ [Bar-Noy et al., 2001]

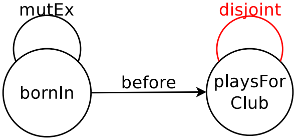
Facts:


bornIn (Beckham, Leytonstone)
bornIn (Beckham, Madrid)
playsForClub (Beckham, Barcelona)
playsForClub (Beckham, Madrid)
4
2
11
8
time
1975
2003
2008

Scheduling Machine:



Constraint Graph:



Scheduling

Facts $\longrightarrow$ *Scheduling Jobs*

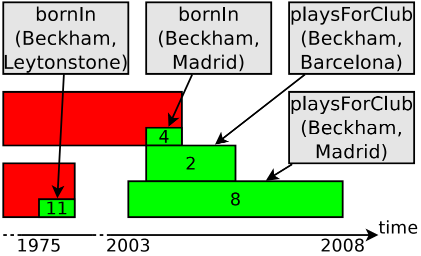
- Weight: Identical
- Capacity $\in [0, 1]$
- Begin: Identical or 0
- End: Identical or ∞

Depending on constraint graph

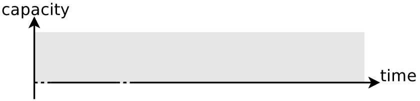
In General:

- Several Scheduling Machines
- $O(|F|log|F| + |F||machines|)$ [Bar-Noy et al., 2001]

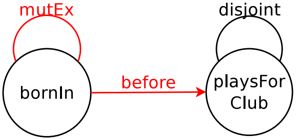
Facts:


bornIn (Beckham, Leytonstone)
bornIn (Beckham, Madrid)
playsForClub (Beckham, Barcelona)
playsForClub (Beckham, Madrid)
4
2
11
8
time
1975
2003
2008

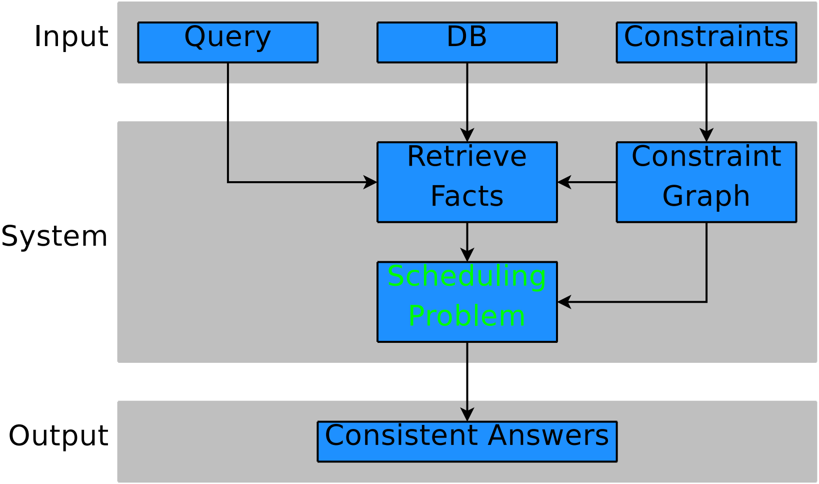
Scheduling Machine:



Constraint Graph:

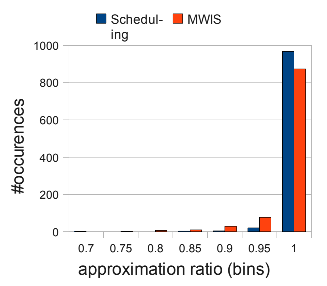


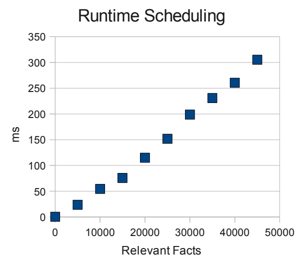
Overview

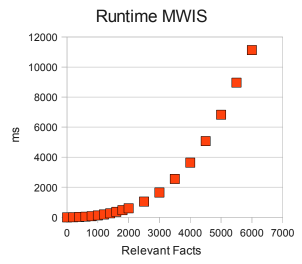

Input
Query
DB
Constraints
System
Retrieve Facts
Constraint Graph
Scheduling Problem
Output
Consistent Answers

Experiments

- 4 correct facts, 20 noisy facts
- Approximation ratio : $\frac{W_{heuristic}}{W_{optimal}}$







The End

Conclusions

- Resolve conflicts by Scheduling
- Runtime in $O(n\ log\ n)$

Future Work

- Generalize Constraints
- Histograms instead of Intervals
- Probabilistic Reasoning

Bar-Noy, A., Bar-Yehuda, R., Freund, A., (Seffi) Naor, J., and Schieber, B. (2001).
A unified approach to approximating resource allocation and scheduling.
J. ACM, 48(5):1069–1090.