Resolving Temporal Conflicts in Inconsistent RDF Knowledge Bases

Maximilian Dylla    Mauro Sozio    Martin Theobald

Max Planck Institute for Informatics, Saarbrücken, Germany
Motivation

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.

- Cruz David Beckham: Born in 2005 in Madrid, Spain.
Extracting Facts

\[ \text{Facts} \subset (\text{Relation} \times \text{Entities} \times \text{Entities}) \]

Weight: \( \text{Facts} \rightarrow \mathbb{R}^+ \)

Time-Interval: \( \text{Facts} \rightarrow \text{Intervals} \)

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.

- Cruz David Beckham: Born in 2005 in Madrid, Spain.

Noisy Facts

- bornIn (Beckham, Madrid)
- playsForClub (Beckham, Barcelona)
- bornIn (Beckham, Leytonstone)

\[ \text{1975} \quad 2003 \quad 2008 \]
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

\[
\begin{align*}
\text{bornIn} & (p, l, t_1) \land \\
\text{playsForClub} & (p, c, t_2) \\
\text{playsForClub} & (p, c_1, t_1) \land \\
\text{playsForClub} & (p, c_2, t_2) \land \\
c_1 & \neq c_2
\end{align*}
\]

\[
\rightarrow \text{before}(t_1, t_2) \quad \rightarrow \text{disjoint}(t_1, t_2)
\]
Answering Queries

Goal: Return only consistent Facts

1. Obtain consistent \( F \subseteq Facts \)
   \[
   \max_{F \subseteq Facts} \sum_{f \in F} w(f)
   \]
   where \( F \) fulfills constraints

2. Answer query within \( F \)
   - NP-hard

Query

\( \text{playsForClub}(David \_Beckham, ?, ?) \)

Facts

Constraint

\[
(\text{bornIn}(p, l, t_1) \land \text{playsForClub}(p, c, t_2)) \rightarrow \text{before}(t_1, t_2)
\]
First Approach

Maximum Weight Independent Set
- Binary constraints
- NP-hard
- Heuristics in $\Omega(|Facts|^2)$

Constraints

\[
\begin{align*}
&\text{bornIn}(p, l, t_1) \land \\
&\text{playsForClub}(p, c, t_2) \\
&\implies \text{before}(t_1, t_2)
\end{align*}
\]

\[
\begin{align*}
&\text{playsForClub}(p, c_1, t_1) \land \\
&\text{playsForClub}(p, c_2, t_2) \\
&\land c_1 \neq c_2 \\
&\implies \text{disjoint}(t_1, t_2)
\end{align*}
\]

Facts

- bornIn (Beckham, Madrid)
- playsForClub (Beckham, Barcelona)
- bornIn (Beckham, Leytonstone)
- playsForClub (Beckham, Madrid)
Scheduling Problem

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

Facts

- bornIn (Beckham, Madrid)
- playsForClub (Beckham, Barcelona)

Constraint

\[
\left( \text{bornIn}(p, l, t_1) \land \text{playsForClub}(p, c, t_2) \right) \rightarrow \text{before}(t_1, t_2)
\]
Overview

Input:
- Query
- Facts
- Constraints

System:
- Retrieve Facts
- Scheduling Problem
- Constraint Graph

Output:
- Consistent Answers
Mapping

- Relations → Vertices
- Constraints → Edges

Constraints

\[
\begin{align*}
( & \text{bornIn}(p, l, t_1) \land \\
& \text{playsForClub}(p, c, t_2) ) \\
\rightarrow & \text{before}(t_1, t_2)
\end{align*}
\]

\[
\begin{align*}
( & \text{playsForClub}(p, c_1, t_1) \land \\
& \text{playsForClub}(p, c_2, t_2) \land \\
& c_1 \neq c_2 ) \\
\rightarrow & \text{disjoint}(t_1, t_2)
\end{align*}
\]
Overview

Input:
- Query
- Facts
- Constraints

System:
- Retrieve Facts
- Constraint Graph
- Scheduling Problem

Output:
- Consistent Answers
Constraints handled by Scheduling

- Tripartite graph \((V_1 \cup V_2 \cup V_3, E)\)
- \(V_1 \cup V_3\) must have \textit{mutEx} loops
- \(V_2\) can have \textit{disjoint} loops
- \textit{before} can edges point:
  - from \(V_1\) to \(V_2 \cup V_3\)
  - or from \(V_2\) to \(V_3\)
Constraints handled by Scheduling

- Tripartite graph \((V_1 \cup V_2 \cup V_3, E)\)
- \(V_1 \cup V_3\) must have \textit{mutEx} loops
- \(V_2\) can have \textit{disjoint} loops
- \textit{before} can edges point:
  - from \(V_1\) to \(V_2 \cup V_3\)
  - or from \(V_2\) to \(V_3\)
Constraints handled by Scheduling

- Tripartite graph \((V_1 \cup V_2 \cup V_3, E)\)
- \(V_1 \cup V_3\) must have \textit{mutEx} loops
- \(V_2\) can have \textit{disjoint} loops
- \textit{before} can edges point:
  - from \(V_1\) to \(V_2 \cup V_3\)
  - or from \(V_2\) to \(V_3\)
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
  $\Rightarrow O(|Facts| \cdot |Constraints|)$
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

**Query**

\[ \text{playsForClub}(David\_Beckham, ?, ?) \]

**Constraint Graph**

In General:

- For each shared Argument
  \[ \Rightarrow O(|Facts| \cdot |Constraints|) \]
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:
  \[ O(|\text{Facts}| \cdot |\text{Constraints}|) \]

Query

\[ \text{playsForClub}(\text{David_Beckham}, ?, ?) \]

Constraint Graph

Select * from facts where\rel=\text{playsForClub} and\arg1=\text{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
  $\Rightarrow O(|Facts| \cdot |Constraints|)$
**Facts → Scheduling Jobs**

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

Depending on constraint graph

**In General:**

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

Facts → Scheduling Jobs

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

Depending on constraint graph

In General:

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

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

Depending on constraint graph

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

Facts: Scheduling Jobs

Scheduling Machine:

Constraint Graph:

Bar-Noy et al., 2001
Experiments

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

![Graph of approximation ratio vs. scheduling parameters]

![Graph of occurrences vs. approximation ratio]

![Graph of runtime scheduling vs. relevant facts]

![Graph of runtime MWIS vs. relevant facts]
The End

Conclusions

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

Future Work

• Generalize Constraints
• Histograms instead of Intervals
• Probabilistic Reasoning
A unified approach to approximating resource allocation and scheduling.