A Temporal-Probabilistic Database Model for Information Extraction - Supplementary Material

Maximilian Dylla Iris Miliaraki Martin Theobald

June 20, 2013

This document provides additional material regarding the experiments, which did not fit into the paper due to space constraints. To ease matching the content of this document and the paper, we named this document's sections according to the headlines of the paper's section 7.

1 Extraction and Reasoning Quality

1.1 Data Set

The data set as described in the paper is available for download here[1].

1.2 Deduction Rules

As a first step, we aggregate the different extractions as distinguished by their Ids to one duplicate-free fact:

$$\begin{aligned}
\mathit{AttendedSchool}'(X,Y)_{[T_b,T_e)} &\leftarrow \mathit{AttendedSchool}'(X,Y,Id)_{[T_b,T_e)}\\
\mathit{Born}'(X)_{[T_b,T_e)} &\leftarrow \mathit{Born}(X,Id)_{[T_b,T_e)}\\
\mathit{Died}'(X)_{[T_b,T_e)} &\leftarrow \mathit{Died}(X,Id)_{[T_b,T_e)}\\
\mathit{Divorce}'(X,Y)_{[T_b,T_e)} &\leftarrow \mathit{Divorce}(X,Y,Id)_{[T_b,T_e)}\\
\mathit{Founded}'(X,Y)_{[T_b,T_e)} &\leftarrow \mathit{Founding}(X,Y,Id)_{[T_b,T_e)}\\
\mathit{GraduatedFrom}'(X,Y)_{[T_b,T_e)} &\leftarrow \mathit{GraduatedFrom}(X,Y,Id)_{[T_b,T_e)}\\
\mathit{IsDating}'(X,Y)_{[T_b,T_e)} &\leftarrow \mathit{IsDating}(X,Y,Id)_{[T_b,T_e)}\\
\mathit{MovedTo}'(X,Y)_{[T_b,T_e)} &\leftarrow \mathit{MovedTo}(X,Y,Id)_{[T_b,T_e)}\\
\mathit{Wedding}'(X,Y)_{[T_b,T_e)} &\leftarrow \mathit{Wedding}(X,Y,Id)_{[T_b,T_e)}
\end{aligned}$$

[1] http://www.mpi-inf.mpg.de/~mdylla/tpdbExtracted.zip

Then, we deduce the *AreMarried* relation:

$$AreMarried'(X,Y)_{[T_b,T'_e)} \leftarrow \begin{pmatrix} \begin{matrix} Wedding'(X,Y)_{[T_b,T_e)} \wedge \\ Divorce'(X,Y)_{[T'_b,T'_e)} \wedge \end{matrix} & T_e \leq^T T'_b \end{pmatrix}$$

$$AreMarried'(X,Y)_{[T_b,T'_e)} \leftarrow \begin{pmatrix} Wedding'(X,Y)_{[T_b,T_e)} \wedge \\ \neg Divorce'(X,Y)_{[T'_b,T'_e)} \end{pmatrix}$$

After grounding all deduction rules we query for the relations *Born'*, *Died'*, *Founded'*, *GraduatedFrom'*, *IsDating'*, *MovedTo'*, and *AreMarried'*.

1.3 Constraints

We manually designed the constraints by measuring their effect on the precision-recall values in the training set. The constraints can be divided into three groups, namely irreflexivenes, precedence and disjointness.
Irreflexive. We achieved improved results by constraining *Divorce*:

$$\neg(Divorce'(Y,X)_{[T_b,T_e)} \wedge X = Y)$$

Precedence. The prime target for precendence constraints is the birth date, which should occur before any other event in the life of a person:

$$\begin{gathered}
\neg(Born'(X)_{[T_b,T_e)} \wedge AreMarried'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(Born'(X)_{[T_b,T_e)} \wedge AttendedSchool'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(Born'(X)_{[T_b,T_e)} \wedge Founded'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(Born'(X)_{[T_b,T_e)} \wedge GraduatedFrom'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(Born'(X)_{[T_b,T_e)} \wedge IsDating'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(Born'(Y)_{[T_b,T_e)} \wedge IsDating'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(Born'(X)_{[T_b,T_e)} \wedge MovedTo'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(Born'(X)_{[T_b,T_e)} \wedge Wedding'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(Born'(Y)_{[T_b,T_e)} \wedge Wedding'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b)
\end{gathered}$$

Also, the date of death should occur after any other fact relating to a person:

$$\begin{gathered}
\neg(AreMarried'(X,Y)_{[T_b,T_e)} \wedge Died'(Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(Founded'(X,Y)_{[T_b,T_e)} \wedge Died'(X)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(GraduatedFrom'(X,Y)_{[T_b,T_e)} \wedge Died'(X)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(IsDating'(X,Y)_{[T_b,T_e)} \wedge Died'(X)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(IsDating'(X,Y)_{[T_b,T_e)} \wedge Died'(Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b) \\
\neg(MovedTo'(X,Y)_{[T_b,T_e)} \wedge Died'(X)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b)
\end{gathered}$$

Finally, we require the *IsDating* relation to take place before the couple is married:

$$\neg(\mathit{IsDating}'(X,Y)_{[T_b,T_e)} \wedge \mathit{AreMarried}'(X,Y)_{[T'_b,T'_e)} \wedge \neg T_e \leq^T T'_b)$$

Disjointness. We enforce marriages of a person X to two different persons Y and Z to be temporally disjoint by writing:

$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(X,Y)_{[T_b,T_e)} \wedge \mathit{AreMarried}'(X,Z)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T_b \leq^T T'_b \wedge T'_b \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(X,Y)_{[T_b,T_e)} \wedge \mathit{AreMarried}'(X,Z)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T_b \leq^T T'_e \wedge T'_e \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(X,Y)_{[T_b,T_e)} \wedge \mathit{AreMarried}'(X,Z)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T'_b \leq^T T_b \wedge T_b \leq^T T'_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(X,Y)_{[T_b,T_e)} \wedge \mathit{AreMarried}'(X,Z)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T'_b \leq^T T_e \wedge T_e \leq^T T'_e \end{array}\right)$$

Now, we give the same constraints, but with exchanged order of arguments:

$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(Y,X)_{[T_b,T_e)} \wedge \mathit{AreMarried}'(Z,X)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T_b \leq^T T'_b \wedge T'_b \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(Y,X)_{[T_b,T_e)} \wedge \mathit{AreMarried}'(Z,X)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T_b \leq^T T'_e \wedge T'_e \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(Y,X)_{[T_b,T_e)} \wedge \mathit{AreMarried}'(Z,X)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T'_b \leq^T T_b \wedge T_b \leq^T T'_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(Y,X)_{[T_b,T_e)} \wedge \mathit{AreMarried}'(Z,X)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T'_b \leq^T T_e \wedge T_e \leq^T T'_e \end{array}\right)$$

We continue by restricting that married persons cannot date other persons during their marriage:

$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(X,Y)_{[T_b,T_e)} \wedge \mathit{IsDating}'(X,Z)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T_b \leq^T T'_b \wedge T'_b \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(X,Y)_{[T_b,T_e)} \wedge \mathit{IsDating}'(X,Z)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T_b \leq^T T'_e \wedge T'_e \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(X,Y)_{[T_b,T_e)} \wedge \mathit{IsDating}'(X,Z)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T'_b \leq^T T_b \wedge T_b \leq^T T'_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(X,Y)_{[T_b,T_e)} \wedge \mathit{IsDating}'(X,Z)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T'_b \leq^T T_e \wedge T_e \leq^T T'_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(Y,X)_{[T_b,T_e)} \wedge \mathit{IsDating}'(Z,X)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T_b \leq^T T'_b \wedge T'_b \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(Y,X)_{[T_b,T_e)} \wedge \mathit{IsDating}'(Z,X)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T_b \leq^T T'_e \wedge T'_e \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(Y,X)_{[T_b,T_e)} \wedge \mathit{IsDating}'(Z,X)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T'_b \leq^T T_b \wedge T_b \leq^T T'_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} \mathit{AreMarried}'(Y,X)_{[T_b,T_e)} \wedge \mathit{IsDating}'(Z,X)_{[T'_b,T'_e)} \\ \wedge Y \neq Z \wedge T'_b \leq^T T_e \wedge T_e \leq^T T'_e \end{array}\right)$$

1.4 Results

1.4.1 100% Dataset

In the paper's Figure 2(a) we present the table of F_1 values together with the detailed precision-recall plot in the paper's Figure 2(b). Here in Figure 1, we additionally depict the precision-recall plots for the other 7 relations.

1.4.2 3% Dataset

Since Markov Logic Networks in its implementations by Alchemy[2] and Tuffy[3] did not scale to the full data set, we ran them on a 3% sample of the data set (about 65 facts in total). The corresponding run-times are depicted in Figure 2. Finally, the following table contains the F_1 measure for the 3% of the data set:

	TPDB-c	TPDB+c	Gurobi	Beast-c	Beast+c	MLN-c	MLN+c	Tuffy-c	Tuffy+c
AreMarried	**0.68**	**0.68**	0.48	0.32	0.32	0.67	0.63	0.49	0.51
AttendedSchool	**0.70**	**0.70**	0.54	0.54	0.54	0.5	0.5	0.54	0.54
Born	0.86	**1.0**	0.89	0.69	0.69	0.25	0.25	0.4	0.53
Died	0.56	0.55	**0.74**	0.55	**0.74**	0.5	0.67	0.55	**0.74**
Founded	**1.0**	**1.0**	0.5	0.5	0.5	**1.0**	**1.0**	**1.0**	**1.0**
GraduatedFrom	**0.99**	**0.99**	0.75	0.75	0.75	0.0	0.0	0.75	0.75
IsDating	**0.48**	**0.48**	0.43	0.43	0.43	0.20	0.20	0.37	0.31
MovedTo	0.75	0.75	0.75	0.57	0.59	**1.0**	**1.0**	0.61	0.64

[2] `http://alchemy.cs.washington.edu/`

[3] `http://hazy.cs.wisc.edu/hazy/tuffy/download/`

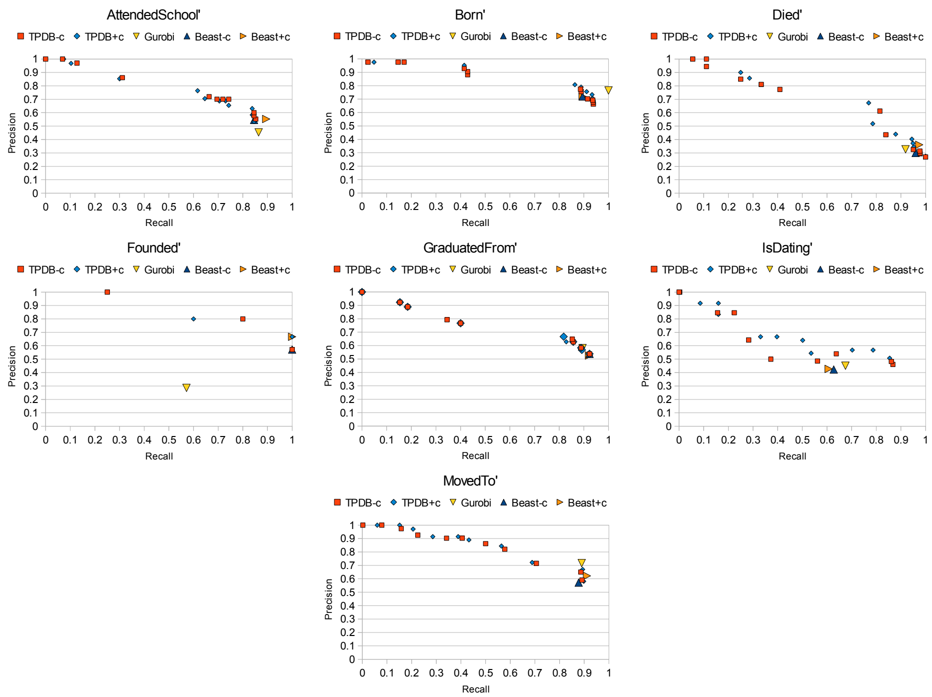

Figure 1: Precision Recall Plots for 100% of the data set

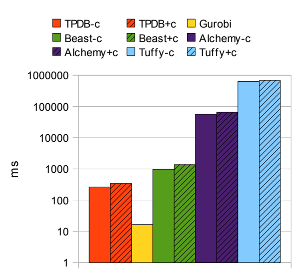

Figure 2: Run-times on 3% of the data set

2 Query Answering Task

2.1 Data Set

We utilized the YAGO2 knowledge base which is freely available for download[4].

2.2 $Q1$ Hierarchical

The hierarchical query forms a union of two join queries and one relation, where the argument of each result is a person's name. We replace *Constant* by 1000 different entities to obtain varying queries and lineage.

$$\begin{array}{rcl} \mathit{Result}(X) & \leftarrow & \mathit{IsMarriedTo}(\mathit{Id}, \mathit{Constant}, X) \\ \mathit{Result}(Y) & \leftarrow & \mathit{Edited}(\mathit{Id}, \mathit{Constant}, X) \wedge \mathit{ActedIn}(\mathit{Id}', Y, X) \\ \mathit{Result}(Y) & \leftarrow & \mathit{IsLeaderOf}(\mathit{Id}, \mathit{Constant}, X) \wedge \mathit{LivesIn}(\mathit{Id}', Y, X) \end{array}$$

The queried relation is *Result*. For MayBMS we run the following query:

```
CREATE TABLE intensional_result_prob AS (
 (SELECT DISTINCT arg0, time_begin, time_end
  FROM
   (SELECT DISTINCT t0.arg2 AS arg0,
                    t0.time_begin AS time_begin,
                    t0.time_end AS time_end
    FROM ismarriedto_prob AS t0,
         dummy_prob AS t1
    WHERE t0.arg1='Constant')
  AS part0)
 UNION
  (SELECT DISTINCT arg0, time_begin, time_end
   FROM
    (SELECT DISTINCT t1.arg1 AS arg0,
                     t0.time_begin AS time_begin,
                     t0.time_end AS time_end
     FROM edited_prob AS t0,
          actedin_prob AS t1
     WHERE t0.arg1='Constant' AND ( t0.arg2=t1.arg2))
  AS part1)
 UNION
```

[4] http://www.mpi-inf.mpg.de/yago-naga/yago/

```sql
  (SELECT DISTINCT arg0, time_begin, time_end
   FROM
     (SELECT DISTINCT t1.arg1 AS arg0,
                      t0.time_begin AS time_begin,
                      t0.time_end AS time_end
      FROM isleaderof_prob AS t0,
           livesin_prob AS t1
      WHERE t0.arg1='Constant' AND t0.arg2=t1.arg2)
   AS part2)
);
SELECT arg0, conf()
FROM intensional_result_prob
GROUP BY arg0;
```

2.3 $Q2$ Read-Once

The query yielding read-once lineage looks as follows:

$$
\begin{array}{lcl}
\mathit{Result}(X) & \leftarrow & \mathit{DiedIn}(Id, X, \mathit{Constant}) \wedge \mathit{HasGivenName}(Id', X, Y) \\
\mathit{Result}(X) & \leftarrow & \mathit{DiedIn}(Id, X, \mathit{Constant}) \wedge \mathit{ActedIn}(Id', X, Y) \\
\mathit{Result}(X) & \leftarrow & \mathit{DiedIn}(Id, X, \mathit{Constant}) \wedge \mathit{WasBornIn}(Id', Y, \mathit{Constant})
\end{array}
$$

Again, we instantiate *Constant* by 1000 different entities to obtain 1000 queries. Also, the queried relation is *Result*. For MayBMS we run the following query:

```sql
CREATE TABLE intensional_result_prob AS (
  (SELECT DISTINCT arg0,
                   time_begin,
                   time_end
   FROM
    (SELECT DISTINCT t1.arg1 AS arg0,
                     t0.time_begin AS time_begin,
                     t0.time_end AS time_end
     FROM diedin_prob AS t0,
          hasgivenname_prob AS t1
     WHERE t0.arg2='Constant' AND t0.arg1=t1.arg1)
   AS part0)
  UNION
  (SELECT DISTINCT arg0,
                   time_begin,
```

```
                    time_end
  FROM
     (SELECT DISTINCT t1.arg1 AS arg0,
                      t0.time_begin AS time_begin,
                      t0.time_end AS time_end
      FROM diedin_prob AS t0,
           actedin_prob AS t1
      WHERE t0.arg2='Constant' AND t0.arg1=t1.arg1)
  AS part1)
 UNION
 (SELECT DISTINCT arg0,
                  time_begin,
                  time_end
  FROM
     (SELECT DISTINCT t0.arg1 AS arg0,
                      t0.time_begin AS time_begin,
                      t0.time_end AS time_end
      FROM diedin_prob AS t0,
           wasbornin_prob AS t1
      WHERE t0.arg2=t1.arg2 AND t1.arg2='Constant')
  AS part2)
);
SELECT arg0, conf()
FROM intensional_result_prob
GROUP BY arg0;
```

2.4 $Q3$ Unsafe

The unsafe query is a boolean query whose lineage alters with every of the 1000 entities we insert for *Constant*:

$$Result(result) \leftarrow \left(\begin{array}{c} ActedIn(Id_0, X, Constant) \wedge WasBornIn(Id_1, X, Y) \\ \wedge DiedIn(Id_2, Z, Y) \wedge WasBornOnDate(Z)_{[T_b, T_e)} \end{array} \right)$$

The queried relation is *Result*. For MayBMS we run the following query:

```
CREATE TABLE intensional_result_prob AS (
  SELECT DISTINCT t0.arg2 AS arg0,
                  t0.time_begin AS time_begin,
                  t0.time_end AS time_end
  FROM actedin_prob AS t0,
```

```
        wasbornin_prob AS t1,
        diedin_prob AS t2,
        wasbornondate_prob AS t3
  WHERE t0.arg2='Constant'
        AND t0.arg1=t1.arg1
        AND t1.arg2=t2.arg2
        AND t2.arg1=t3.arg1);
SELECT arg0, conf()
FROM intensional_result_prob
GROUP BY arg0;
```

3 Knowledge Building Task

3.1 $Q4$ Large Lineage

This query consists of two deduction rules, where the first encodes a $\#\mathcal{P}$-hard subquery, which is used in each of the second deduction rule.

$$\begin{array}{rcl} \mathit{Expensive}(\mathit{result}) & \leftarrow & \left(\begin{array}{c} \mathit{IsLocatedIn}(Id_0, X, Y) \wedge \mathit{WasBornIn}(Id_1, Z, X) \\ \wedge \mathit{LivesIn}(Id_2, Z, U) \end{array} \right) \\ \mathit{Result}(X, Y)_{[T_b, T'_e)} & \leftarrow & \left(\begin{array}{c} \mathit{IsMarriedTo}(Id_0, X, Y) \wedge \mathit{OccursSince}(Id_1, Id_0)_{[T_b, T_e)} \\ \wedge \mathit{OccursUntil}(Id_2, Id_0)_{[T'_b, T'_e)} \wedge \mathit{Expensive}(\mathit{result}) \end{array} \right) \end{array}$$

We query for the relation *Result*. For MayBMS we run the following query:

```
CREATE TABLE intensional_result_prob AS (
  SELECT DISTINCT t0.arg1 AS arg0,
                  t0.arg2 AS arg1,
                  t0.time_begin AS time_begin,
                  t0.time_end AS time_end
  FROM ismarriedto_prob AS t0,
       occurssince_prob AS t1,
       occursuntil_prob AS t2,
       islocatedin_prob AS t4,
       wasbornin_prob AS t5,
       livesin_prob AS t6
  WHERE t0.arg0=t1.arg1 AND
        t1.arg1=t2.arg1 AND
        t4.arg1=t5.arg2 AND
        t5.arg1=t6.arg1);
SELECT arg0, arg1, conf()
```

```
FROM intensional_result_prob
GROUP BY arg0, arg1;
```

3.2 $Q5$ Many Constraints

We use the following deduction rules:

$$
\begin{array}{rcl}
\mathit{Born}(X)_{[T_b,T_e)} & \leftarrow & \mathit{WasBornOnDate}(Id,X)_{[T_b,T_e)} \\
\mathit{Died}(X)_{[T_b,T_e)} & \leftarrow & \mathit{DiedOnDate}(Id,X)_{[T_b,T_e)} \\
\mathit{Divorce}(X,Y)_{[T_b,T_e)} & \leftarrow & \mathit{IsMarriedTo}(Id,X,Y) \wedge \mathit{OccursUntil}(Id',Id)_{[T_b,T_e)} \\
\mathit{HasChild}(X,Y)_{[T_b,T_e)} & \leftarrow & \mathit{HasChild}(Id,X,Y) \wedge \mathit{WasBornOnDate}(Id',Y)_{[T_e,T_b)} \\
\mathit{Wedding}(X,Y)_{[T_b,T_e)} & \leftarrow & \mathit{IsMarriedTo}(Id,X,Y) \wedge \mathit{OccursSince}(Id',Id)_{[T_b,T_e)} \\
\mathit{Marriage}(X,Y)_{[T_b,T_e')} & \leftarrow & \mathit{Wedding}(X,Y)_{[T_b,T_e)} \wedge \mathit{Divorce}(X,Y)_{[T_e',T_b')}
\end{array}
$$

We query for the relations *Born*, *Died*, *HasChild*, and *Marriage*. Also, we apply these constraints to the deduced facts. The precedence constraints are:

$$
\begin{array}{c}
\neg(\mathit{Born}(X)_{[T_b,T_e)} \wedge \mathit{HasChild}(X,Y)_{[T_b',T_e')} \wedge \neg T_e \leq^T T_b') \\
\neg(\mathit{Born}(X)_{[T_b,T_e)} \wedge \mathit{Marriage}(X,Y)_{[T_b',T_e')} \wedge \neg T_e \leq^T T_b') \\
\neg(\mathit{Born}(Y)_{[T_b,T_e)} \wedge \mathit{Marriage}(X,Y)_{[T_b',T_e')} \wedge \neg T_e \leq^T T_b') \\
\neg(\mathit{Marriage}(X,Y)_{[T_b,T_e)} \wedge \mathit{Died}(X)_{[T_b',T_e')} \wedge \neg T_e \leq^T T_b') \\
\neg(\mathit{Marriage}(X,Y)_{[T_b,T_e)} \wedge \mathit{Died}(Y)_{[T_b',T_e')} \wedge \neg T_e \leq^T T_b')
\end{array}
$$

Regarding disjointness we utilize:

$$\neg\left(\begin{array}{c} Marriage(X,Y)_{[T_b,T_e)} \wedge Marriage(X,Z)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b \leq^T T_b' \wedge T_b' \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(X,Y)_{[T_b,T_e)} \wedge Marriage(X,Z)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b \leq^T T_e' \wedge T_e' \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(X,Y)_{[T_b,T_e)} \wedge Marriage(X,Z)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b' \leq^T T_b \wedge T_b \leq^T T_e' \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(X,Y)_{[T_b,T_e)} \wedge Marriage(X,Z)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b' \leq^T T_e \wedge T_e \leq^T T_e' \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(X,Y)_{[T_b,T_e)} \wedge Marriage(Z,X)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b \leq^T T_b' \wedge T_b' \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(X,Y)_{[T_b,T_e)} \wedge Marriage(Z,X)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b \leq^T T_e' \wedge T_e' \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(X,Y)_{[T_b,T_e)} \wedge Marriage(Z,X)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b' \leq^T T_b \wedge T_b \leq^T T_e' \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(X,Y)_{[T_b,T_e)} \wedge Marriage(Z,X)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b' \leq^T T_e \wedge T_e \leq^T T_e' \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(Y,X)_{[T_b,T_e)} \wedge Marriage(Z,X)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b \leq^T T_b' \wedge T_b' \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(Y,X)_{[T_b,T_e)} \wedge Marriage(Z,X)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b \leq^T T_e' \wedge T_e' \leq^T T_e \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(Y,X)_{[T_b,T_e)} \wedge Marriage(Z,X)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b' \leq^T T_b \wedge T_b \leq^T T_e' \end{array}\right)$$
$$\neg\left(\begin{array}{c} Marriage(Y,X)_{[T_b,T_e)} \wedge Marriage(Z,X)_{[T_b',T_e')} \\ \wedge Y \neq Z \wedge T_b' \leq^T T_e \wedge T_e \leq^T T_e' \end{array}\right)$$

The integer linear programs encode the same constraints.

4 Detailed Runtime Analysis

4.1 $Q6$ Deduplication

The query pattern is designed to yield exactly one answer fact, but with a varying number of intervals attached to it. We instantiate *Constant* by 1000 different entities (in form of locations) to obtain the 1000 queries.

$$\begin{aligned}
Born(X)_{[T_b,T_e)} &\leftarrow \begin{pmatrix} WasBornIn(Id, X, Constant)\wedge \\ WasBornOnDate(Id', X)_{[T_b,T_e)} \end{pmatrix} \\
Died(X)_{[T_b,T_e)} &\leftarrow \begin{pmatrix} DiedIn(Id, X, Constant)\wedge \\ DiedOnDate(Id', X)_{[T_b,T_e)} \end{pmatrix} \\
Lives(result)_{[T_b,T'_e)} &\leftarrow Born(X)_{[T_b,T_e)} \wedge Died(X)_{[T'_b,T'_e)} \\
Lives(result)_{[t_{min},T'_e)} &\leftarrow \neg Born(X)_{[T_b,T_e)} \wedge Died(X)_{[T'_b,T'_e)} \\
Lives(result)_{[T_b,t_{max})} &\leftarrow Born(X)_{[T_b,T_e)} \wedge \neg Died(X)_{[T'_b,T'_e)}
\end{aligned}$$

The queried relation is *Lives*. Here, *result* is a constant, such there is only one answer fact per query.