Fuzzy Constraint-based Schema Matching Formulation

Alsayed Algergawy, Eike Schallehn, Gunter Saake
{alshahat|eike|saake}@ovgu.de

Department of Computer Science
University of Magdeburg
Magdeburg, Germany

The First Workshop in Advanced Deep Web 2008
ADW2008
1. Motivations

2. Preliminaries

3. Schema Graphs

4. Schema Matching as an FCOP

5. Summary and Future Work
Motivations

- *Schema matching* is defined as the task of identifying the semantic correspondences from heterogeneous data sources

- **Current Approaches**
  - Lack of formulation
  - Discovering simple mappings
  - Matching Performance
  - Matching Scalability
  - Uncertainty

Therefore, we need a formalization framework that enables us to cope with:

- Discovering complex mappings as well as simple mappings
- Trading-off between two performance aspects—matching effectiveness and matching efficiency
- Dealing with schema matching uncertainty
Motivations

- **Schema matching** is defined as the task of identifying the semantic correspondences from heterogeneous data sources.

- **Current Approaches**
  - Lack of formulation
  - Discovering simple mappings
  - Matching Performance
  - Matching Scalability
  - Uncertainty

- **Therefore, we need a formalization framework that enables us to cope with:**
  - Discovering complex mappings as well as simple mappings
  - Trading-off between two performance aspects—matching effectiveness and matching efficiency
  - Dealing with schema matching uncertainty
1. Motivations

2. Preliminaries

3. Schema Graphs

4. Schema Matching as an FCOP

5. Summary and Future Work
Preliminaries

- Our fuzzy constraint optimization framework is based on:
  - Rooted labeled graphs
  - Constraint programming
Rooted Labeled Graphs

- Schemas to be matched can be modeled as rooted labeled graphs called schema graphs $SG$


- $N_G = \{n_{root}, n_2, \ldots, n_n\} \Rightarrow$ a finite set of nodes
Rooted Labeled Graphs

- Schemas to be matched can be modeled as rooted labeled graphs called schema graphs $SG$


- $N_G = \{n_{\text{root}}, n_2, \ldots, n_n\} \Rightarrow$ a finite set of nodes
- $E_G = \{(n_i, n_j)|n_i, n_j \in N_G\} \Rightarrow$ a finite set of edges,
Rooted Labeled Graphs

- Schemas to be matched can be modeled as rooted labeled graphs called schema graphs $SG$


- $N_G = \{n_{root}, n_2, ..., n_n\} \Rightarrow$ a finite set of nodes
- $E_G = \{(n_i, n_j)| n_i, n_j \in N_G\} \Rightarrow$ a finite set of edges,
- $Lab_G =\{Lab_{NG}, Lab_{EG}\} \Rightarrow$ a finite set of node labels $Lab_{NG}$, and a finite set of edge labels $Lab_{EG}$
Rooted Labeled Graphs

- Schemas to be matched can be modeled as rooted labeled graphs called schema graphs $SG$


- $N_G = \{n_{root}, n_2, \ldots, n_n\}$ ⇒ a finite set of nodes
- $E_G = \{(n_i, n_j)|n_i, n_j \in N_G\}$ ⇒ a finite set of edges,
- $Lab_G = \{Lab_{NG}, Lab_{EG}\}$ ⇒ a finite set of node labels $Lab_{NG}$, and a finite set of edge labels $Lab_{EG}$
- $src$ and $tar$: $E_G \leftrightarrow N_G$ ⇒ two mappings source and target,
Rooted Labeled Graphs

- Schemas to be matched can be modeled as rooted labeled graphs called schema graphs $SG$


- $N_G = \{n_{\text{root}}, n_2, ..., n_n\} \Rightarrow$ a finite set of nodes
- $E_G = \{(n_i, n_j) | n_i, n_j \in N_G\} \Rightarrow$ a finite set of edges,
- $Lab_G = \{Lab_{NG}, Lab_{EG}\} \Rightarrow$ a finite set of node labels $Lab_{NG}$, and a finite set of edge labels $Lab_{EG}$
- $src$ and $tar$: $E_G \mapsto N_G \Rightarrow$ two mappings source and target,
- $l: N_G \cup E_G \mapsto Lab_G \Rightarrow$ a mapping label assigning
A lot of problems in computer science, most notably in AI, can be interpreted as special cases of constraint programming.

Semantic schema matching is an intelligent process.

Therefore, constraint programming is a suitable framework for interpreting and understanding the schema matching problem.
Constraint Programming I

- A lot of problems in computer science, most notably in AI, can be interpreted as special cases of constraint programming.
- Semantic schema matching is an intelligent process
- Therefore, constraint programming is a suitable framework for interpreting and understanding the schema matching problem

- Types of constraint problems
  - Constraint Satisfaction Problem *CSP*
  - Constraint Optimization Problem *COP*
  - Fuzzy Constraint Optimization Problem *FCOP*
• CSP $P$ is a 3-tuple,

$$P = (X, D, C)$$

• $X$ is a finite set of variables
• $D$ is a collection of finite domains
• $C$ is a set of constraints
Constraint Programming II

- CSP $P$ is a 3-tuple,
  $$P = (X, D, C)$$
  - $X$ is a finite set of variables
  - $D$ is a collection of finite domains
  - $C$ is a set of constraints

  Constraint
  $$C_s \subseteq D_1 \times \ldots \times D_r \rightarrow \{0, 1\}$$
  $$S = \{x_1, x_2, \ldots x_r\}$$
• **CSP** $P$ is a 3-tuple,

$$P = (X, D, C)$$

  - $X$ is a finite set of variables
  - $D$ is a collection of finite domains
  - $C$ is a set of constraints

• Constraint

$$C_s \subseteq D_1 \times \ldots \times D_r \rightarrow \{0, 1\}$$

$$S = \{x_1, x_2, \ldots x_r\}$$

• Solution of a **CSP**

An assignment $\Lambda$ is a solution of a **CSP** if it satisfies all the constraints of the problem.
Constraint Programming III

- **COP** COP Q is a 2-tuple, \( Q = (P, g) \)
  - \( P \) is a CSP
  - \( g \) is an objective function
Constraint Programming III

- **COP COP Q** is a 2-tuple, $Q = (P, g)$
  - $P$ is a CSP
  - $g$ is an objective function
- While powerful, both CSP and COP present some limitations
  - ALL constraints are mandatory (CRISP CONSTRAINTS)
• **COP** COP Q is a 2-tuple, $Q = (P, g)$
  • $P$ is a CSP
  • $g$ is an objective function

• While powerful, both **CSP** and **COP** present some limitations
  • ALL constraints are mandatory (**CRISP CONSTRAINTS**)

• Fuzzy Constraints: A fuzzy constraint $C_\mu$ is represented by the fuzzy relation $R_f$, defined by

$$
\mu_R : \prod_{x_i \in \text{var}(C)} D_i \rightarrow [0, 1]
$$
Constraint Programming III

- **COP COP** $Q$ is a 2-tuple, $Q = (P, g)$
  - $P$ is a CSP
  - $g$ is an objective function
- While powerful, both CSP and COP present some limitations
  - ALL constraints are mandatory (**CRISP CONSTRAINTS**)
- Fuzzy Constraints: A fuzzy constraint $C_{\mu}$ is represented by the fuzzy relation $R_f$, defined by
  \[
  \mu_R : \prod_{x_i \in \text{var}(C)} D_i \rightarrow [0, 1]
  \]
- Fuzzy Constraint Optimization Problem FCOP $Q_{\mu}$ is a 4-tuple
  \[
  Q_{\mu} = (X, D, C_{\mu}, g)
  \]
Road Map

1. Motivations

2. Preliminaries

3. Schema Graphs

4. Schema Matching as an FCOP

5. Summary and Future Work
A Unified Schema Matching Framework
Transformation Rules

• Every *prepared matching object* in a schema such as schema, relations, elements, attributes etc. is represented by a *node* in the schema graph.

• The *features* of the prepared matching object are represented by *node labels Lab*$_{NG}$.

• The *relationship* between two prepared matching objects is represented by *an edge* of the schema graph.

• The *features* of the relationship between prepared objects are represented by *edge labels Lab*$_{EG}$.
Relational Schema

Schema S

create table Personnel(
Pno int primary key,
Pname string,
Dept string,
Born date);
Schema Graph Example I

Relational Schema

Schema S
    create table Personnel(
        Pno int primary key,
        Pname string,
        Dept string,
        Born date);
Relational Schema

Schema T

```sql
create table Employee(
    EmpNo int primary key,
    EmpName varchar(20),
    DeptNo int REFERENCES Department,
    Salary int,
    BirthDate date);

create table Department(
    DeptNo int primary key,
    DeptName varchar(30));
```
Relational Schema

Schema T

cREATE TABLE Employee(
    EmpNo INT PRIMARY KEY,
    EmpName VARCHAR(20),
    DeptNo INT REFERENCES Department,
    Salary INT,
    BirthDate DATE);

cREATE TABLE Department(
    DeptNo INT PRIMARY KEY,
    DeptName VARCHAR(30));
Road Map

1. Motivations

2. Preliminaries

3. Schema Graphs

4. Schema Matching as an FCOP

5. Summary and Future Work
The schema matching problem is converted into graph matching

- Graph Morphism; $N_1 \neq N_2$ (schema matching)
- Graph Homomorphism; $N_1 = N_2$
The schema matching problem is converted into graph matching

- Graph Morphism; \( N_1 \neq N_2 \) (schema matching)
- Graph Homomorphism; \( N_1 = N_2 \)

Graph Morphism

\[ \phi : SG_1 \rightarrow SG_2 \]

\[ SG_1 = (N_{GS}, E_{GS}, Lab_{GS}, src_S, tar_S, l_S) \]
\[ SG_2 = (N_{GT}, E_{GT}, Lab_{GT}, src_T, tar_T, l_T) \]
\[ \phi = (\phi_N, \phi_E) \text{ such that } \phi_N : N_{GS} \rightarrow N_{GT}, \phi_E : E_{GS} \rightarrow E_{GT} \]
The schema matching problem is converted into graph matching

- Graph Morphism; \( N_1 \neq N_2 \) (schema matching)
- Graph Homomorphism; \( N_1 = N_2 \)

Graph Morphism

\[
\phi : SG1 \rightarrow SG2
\]

\[
SG1 = (N_{GS}, E_{GS}, Lab_{GS}, src_S, tar_S, l_S)
\]

\[
SG2 = (N_{GT}, E_{GT}, Lab_{GT}, src_T, tar_T, l_T)
\]

\[
\phi = (\phi_N, \phi_E) \text{ such that } \phi_N : N_{GS} \rightarrow N_{GT}, \ \phi_E : E_{GS} \rightarrow E_{GT}
\]

1. \( \forall n \in N_{GS} \exists l_S(n) = l_T(\phi_N(n)) \) (node label preserving)
2. \( \forall e \in E_{GS} \exists l_S(e) = l_T(\phi_E(e)) \) (edge label preserving)
3. \( \forall e \in E_{GS} \exists \) a path \( p' \in N_{GT} \times E_{GT} \) such that \( p' = \phi_E(e) \) and

\[
\phi_N(src_S(e)) = src_T(\phi_E(e)) \land \phi_N(tar_S(e)) = tar_T(\phi_E(e)). \text{(graph structure preserving)}
\]
Graph matching is considered to be one of the most complex problems in computer science. Its complexity is due to two major problems:

- The time complexity
- The fact that all of the algorithms for graph matching found so far can only be applied to two graphs at a time.
Graph matching is considered to be one of the most complex problems in computer science. Its complexity is due to two major problems:-

- The time complexity
- The fact that all of the algorithms for graph matching found so far can only be applied to two graphs at a time.

*To tackle these challenges, as well as the mentioned motivations, we decide to extend graph matching into an FCOP*
Graph Matching as an FCOP

- Graph matching → an FCOP using the following rules:
  - take the *objects of one schema graph* to be matched as the *CPs set of variables*,
  - take the *objects of the other schema graph* to be matched as the *variables domain*
  - find a proper translation of the *conditions that apply to a schema matching* into a *set of constraints*, and
  - form the *objective functions* to be optimized.
The set of variables $X$:

$$X = X_N \cup X_E$$

$$= \{x_{n1}, x_{n2}, x_{n3}, x_{n4}, x_{n5}, x_{n6}\} \cup \{x_{e12}, x_{e23}, x_{e24}, x_{e25}, x_{e26}\}$$

$$= \{x_{n1}, x_{n2}, x_{n3}, x_{n4}, x_{n5}, x_{n6}, x_{e12}, x_{e23}, x_{e24}, x_{e25}, x_{e26}\}$$
Schema Matching as an FCOP: Example

- The set of variables $X$:

  \[ X = X_N \cup X_E \]
  \[ = \{ x_{n1}, x_{n2}, x_{n3}, x_{n4}, x_{n5}, x_{n6} \} \cup \{ x_{e12}, x_{e23}, x_{e24}, x_{e25}, x_{e26} \} \]
  \[ = \{ x_{n1}, x_{n2}, x_{n3}, x_{n4}, x_{n5}, x_{n6}, x_{e12}, x_{e23}, x_{e24}, x_{e25}, x_{e26} \} \]

- The set of domain $D$:

  \[ D = N_{GT} \cup E_{GT} \]
  \[ = \{ D_{n1}, D_{n2}, D_{n3}, D_{n4}, D_{n5}, D_{n6} \} \cup \{ D_{e12}, D_{e23}, D_{e24}, D_{e25}, D_{e26} \} \]
  \[ = \{ D_{n1}, D_{n2}, D_{n3}, D_{n4}, D_{n5}, D_{n6}, D_{e12}, D_{e23}, D_{e24}, D_{e25}, D_{e26} \} \]
Schema Matching as an FCOP: Example

- The set of variables $X$:

  \[ X = X_N \cup X_E \]

  \[ = \{ x_{n1}, x_{n2}, x_{n3}, x_{n4}, x_{n5}, x_{n6} \} \cup \{ x_{e12}, x_{e23}, x_{e24}, x_{e25}, x_{e26} \} \]

  \[ = \{ x_{n1}, x_{n2}, x_{n3}, x_{n4}, x_{n5}, x_{n6}, x_{e12}, x_{e23}, x_{e24}, x_{e25}, x_{e26} \} \]

- The set of domain $D$:

  \[ D = N_{GT} \cup E_{GT} \]

  \[ = \{ D_{n1}, D_{n2}, D_{n3}, D_{n4}, D_{n5}, D_{n6} \} \cup \{ D_{e12}, D_{e23}, D_{e24}, D_{e25}, D_{e26} \} \]

  \[ = \{ D_{n1}, D_{n2}, D_{n3}, D_{n4}, D_{n5}, D_{n6}, D_{e12}, D_{e23}, D_{e24}, D_{e25}, D_{e26} \} \]

  \[ D_{n1} = D_{n2} = D_{n3} = D_{n4} = D_{n5} = D_{n6} = \{ n_1T, n_2T, n_3T, n_4T, n_5T, n_6T, n_7T, n_8T, n_9T, n_{10}T \} \]
Constraint Construction

- Syntactic constraints
  - Domain Constraint
    \[ C_{\mu(x_{ni})}^{\text{dom}} = \{ d_i \in D_{Ni} \} \]
    \[ C_{\mu(x_{ei})}^{\text{dom}} = \{ d_i \in D_{Ei} \} \]

- Structural Constraints
  - Parent Constraint
    \[ C_{\mu(x_{ni}, x_{nj})}^{\text{parent}} = \{ (d_i, d_j) \in D_N \times D_N | \exists e (d_i, d_j) \text{ s.t. } \text{src}(e) = d_i \} \]
  - Child Constraint
    \[ C_{\mu(x_{ni}, x_{nj})}^{\text{child}} = \{ (d_i, d_j) \in D_N \times D_N | \exists e (d_i, d_j) \text{ s.t. } \text{tar}(e) = d_j \} \]
Constraint Construction

- Syntactic constraints
  - Domain Constraint
    \[ C_{\mu(x_{ni})}^{\text{dom}} = \{ d_i \in D_{Ni} \} \]
    \[ C_{\mu(x_{ei})}^{\text{dom}} = \{ d_i \in D_{Ei} \} \]

- Structural Constraints
  - Parent Constraint
    \[ C_{\mu(x_{ni}, x_{nj})}^{\text{parent}} = \{ (d_i, d_j) \in D_N \times D_N | \exists e (d_i, d_j) \text{ s.t. src}(e)=d_i \} \]
  - Child Constraint
    \[ C_{\mu(x_{ni}, x_{nj})}^{\text{child}} = \{ (d_i, d_j) \in D_N \times D_N | \exists e (d_i, d_j) \text{ s.t. tar}(e)=d_j \} \]

- Semantic constraints
  - Labeled Constraints
    \[ C_{\mu(x_i)}^{\text{Lab}} = \{ d_j \in D_N | lsim(l_S(x_i), l_T(d_j)) \geq t \} \]
    \[ C_{\mu(x_i)}^{\text{Lab}} = \{ d_j \in D_E | lsim(l_S(x_i), l_T(d_j)) \geq t \} \]
Objective Function Construction

- is the function associated with the optimization process
- constitutes the implementation of the problem to be solved.
- The input parameters are the object parameters
- The output is the objective value representing the evaluation/quality of the individual

$$g = \min | \max ( \sum_{\text{set of constraint}} f_{\text{cost}} + \sum_{\text{set of assignment}} f_{\text{energy}} )$$
Road Map

1. Motivations

2. Preliminaries

3. Schema Graphs

4. Schema Matching as an FCOP

5. Summary and Future Work
Summary and Future Work

- Building a conceptual connection between the schema matching problem and fuzzy constraint optimization problem
- Developing a formal framework for the SMP, which
  - generic framework; model and domain independent
  - able to handle uncertainty
  - able to cope with complex mappings
- Benefits behind formulation:
  - Increase our understanding of the problem
  - Help mapping of the problem into another well-known problem
  - Open a path to adopt of different existing algorithms
  - Guide the initial design of the schema matching prototype
- Future work?? Implementation, evaluation, and comparison with other mainstream systems
Thank You
Thank You
Questions??