On Intensional Answers to Database Queries

Helmut Horacek

German Research Center for Artificial Intelligence
Language Technology Division
Saarbrücken, Germany
Email: helmut.horacek@dfki.de

SEMAPRO 2021 -
The 15. International Conference on Advances in Semantic Processing
Barcelona, October 3, 2021 - October 7, 2021
**Position**

Researcher at DFKI Saarbrücken, Language Technology
Lecturer at University of the Saarland

**Research interests**

Natural language processing, specifically generation
Dialog systems
Proof presentation
Computational models of natural argument
Intelligent tutoring systems
Game playing
Projects I have contributed (back > 10 years)

FABEON - NL proof presentation (DFG project)
DIALOG - part of the Collaborative Research Centre on Resource-Adaptive Cognitive Processes (SFB 378) at University of the Saarland

Recent Projects at Language Technology DFKI

QT21: Quality Translation 21
EU Council Presidency Translator
Deep Learning for End-to-End-Applications in Language Technology
feature learning without a priori knowledge about the language to be translated
Answers to Database Queries

Extensional answers (normal)
an enumeration of the data satisfying the query

Intensional answers (specific)
a discriminatory description of this data

Examples for intensional answers

„Which states have a capital?“
„All German states.“ (Integrity check, limited DB modeling)

„Which state does Missouri river pass through”?
„All the states that border South Dakota except Wyoming.“
   including exception handling (Montana, South Dakota, North Dakota, Nebraska, Iowa, Kansas, Missouri)
Motivation

Expected benefits

Avoids potentially long enumerations
Highlights commonalities among items
May provide new (hopefully) useful information

Difficulties/ problem areas

Limited scope - potential extensions recently explored
Computational complexity
Situational suitability of descriptions may be problematic
Content

Research context
  Cooperative answers
  Deductive intentional answers (classical)
  Inductive intentional answers (experimental)

Inductive intentional answers
  Technical approaches
  Heuristics, complexity, limitations

An equivalent linguistic task
  The task - generating referring expressions
  Approaches, algorithmic issues

The future
  Algorithmic transfer
  Issues of conceptual suitability
Cooperative Answers

Adequate treatment of empty responses
Detection of presupposition failures in queries
Stating the intermediate empty result

Answering slightly modified queries
Using knowledge bases
Approaching desired results by handling near misses, slight variations (numerical, conceptual)

Overanswering
Anticipating follow-up questions
Adding conceptually related data
**Intensional answers (classical)**

**Definitions**
- Circumscription of the semantics of the query
- Sufficient conditions for the truth of the query
  - independent of content of the database

**Issues**
- Exploiting subsumption, aiming at simplifications
- Using vocabulary of interest to users
- Based on reduced integrity constraints, deductive rules
Categories of Intensional Answers

Properties
Only intensional descriptions versus mixed forms
Independency/dependency of database content
Completeness in describing the extensional data

Approaches discussed here
Purely based on extensional data
Combinations (exceptions, near misses)
Extensions motivated by linguistic tasks
Intensional Answers built from Extensions

General approach

Based on inductive logic programming

Learning a concept that subsumes only the extensions in the answer

Descriptions of extensions are built and incrementally generalized

Choices in descriptions are made to reduce the complexity

Properties

Mixed - may include extensional data

Partial - not all/complete intensional descriptions generated
The Basic Idea (Hartfiel, Cimiano)

Ingredients of the approach

A knowledge base, with categories and properties of items

Procedural steps

Building a least general generalization, if possible
Eliminating redundant clauses
Checking coverage of only positive items

Possible results

An intensional description
Failure to produce a (simple) description
Generalization Algorithm

1: \( \text{Ans} = \text{ExtensionalAnswers}.\text{getNext}() \);
2: \( C = \text{constructClause}(\text{Ans}, KB) \);
3: \( \text{if } |\text{ExtensionalAnswers}| < 2 \text{ then} \)
4: \( \text{return } C \);
5: \( \text{while not all answers have been processed do} \)
6: \( \text{Ans'} = \text{ExtensionalAnswers}.\text{getNext}() \);
7: \( C' = \text{constructClause}(\text{Ans'}, KB) \);
8: \( c = \text{LGG}(C, C') \);
9: \( \text{Answers'} = \text{evaluateQuery(query(c), KB)} \);
10: \( \text{if } \text{ExtensionalAnswers} \cap \text{Answers'} \subseteq \text{Answers'} \text{ then} \)
11: \( \text{return } \phi \); \( \triangleright \) no consistent clause can be found
12: \( C'' = \text{reduceClause}(c, KB, \text{ExtensionalAnswers}) \);
13: \( \text{Answers''} = \text{evaluateQuery(query(c''), KB)} \);
14: \( \text{if } \text{Answers''} = \text{Answers} \text{ then} \); \( \triangleright \) i.e. \( C'' \) covers all answers
15: \( \text{return } C'' \);
16: \( \text{return } \text{reduceClause}(c, KB, \text{Answers}) \);
General Architecture

Diagram showing the following steps:

1. Question
2. Preprocessing
3. Query
4. Database
5. Output
6. Extensional Answers
7. Intensional answer clause
8. LGG
9. Generalization Algorithm
Reduction Algorithm

ReduceClause(Clause c, KnowledgeBase KB, Set Answers)

1. List literals = orderedLiterals(c); ▷ (in increasing order)
2. for i=1 to Literals do
3.     c′ = remove(c, L_i);
4.     Answers′ = evaluateQuery(query(c′), KB);
5.     if Answers ∩ Answers′ = Answers′ then ▷ i.e. c′ is consistent
6.         c = c′;
7. return c;
Heuristics Used

Ordering literals
- According to arity
- According to commonly appearing variables
- Domain-specific ordering, according to query type

Usage in reduction step
- Literals with higher arity are checked with priority
  (e.g., „lake length“ prior to „lake name“, …)

Aims at producing more compact descriptions
Some Examples (Hartfiel)

”Which rivers flow through states which Saarland borders?”
„All that flow through Koblenz.“ (Mosel, Rhein)

”Which states does the Main flow through?”
„All that border Baden Wuerttemberg and Thueringen“

”Which countries border a state which the Main flows through?”
„All countries that border Bayern.“ (Austria, Czech Republic)

”Which cities have more than 1500000 inhabitants ?”
„All that are located at the A24 and which a river flows through.“

”Which rivers flow through more than 5 cities ?“
„All that flow through Duisburg.“ (Rhein, Ruhr)
The Situation so far

Some promising results

Limited success
not always an intentional description available

A related field
Let us check linguistic approaches
  generating referring expressions
Generating Referring Expressions (GRE)

Given
A set of objects,
described in terms of entries in a knowledge base
(focusing on semantics, abstracting from surface)

Goal specification
A referring expression that identifies (uniquely)
the intended referent(s) most naturally

Comparison
Exactly the same task (approaches are quite different!)
Basic Ingredients in GRE

Algorithmic issues

Incrementally builds a description out of components that apply to the intended referent(s) (attributes, limited uses of relations)

Coverage of referents

Mostly elaborated/used for single referents

Works also for sets of referents with a joint description

Comparison

Orthogonal search organization, same limitation
Some Psychological Insights

Human preferences

- Basic categories (dog as opposed to poodle)
- Easy perceivable attributes (color quite prominent)
- Redundant expressions (some extra attributes)

Personal preferences

- Using location attributes or not
- More or less redundancy

Salience is the driving factor
Search organization

Database context

Increasingly covering positive items, one-by-one
Starting with relatively large descriptions
Combining descriptions, eliminating components
Finishing if all positive items are covered

GRE context

Increasingly adding descriptors/attributes, one-by-one
Starting with a simple description (e.g., object category)
Adding descriptors, checking termination
An Extension in GRE - for Sets of Objects

The basic idea (van Deemter 2000)

Boolean combinations of attributes, not only single attributes

Search technique proposed

Increasingly complex combinations considered
(single attributes, combinations of two attributes, ... )

An important theoretical result

Identifying description (intentional answer) is guaranteed
(if it exists; it must exist in the database context)
### An Example Scenario

<table>
<thead>
<tr>
<th>descriptors/objects</th>
<th>$x_0$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
<th>$x_{10}$</th>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>vehicle</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>car</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>sportscar</td>
<td></td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>truck</td>
<td></td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>blue</td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>red</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>white</td>
<td>⬤</td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>center</td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>left</td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>right</td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>big</td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>small</td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>new</td>
<td>⬤</td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>old</td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
</tbody>
</table>
Descriptions with Boolean Combinations

From the perspective of intensional answers to databases

No single answer possible (no set of total commonalities)
Several partially covering answers, varying complexity
At worst only an enumeration, equal to extensional answers

Dealing with near-misses becomes of interest

Exclusions: “the vehicles on the right, but not the red truck”

Boolean combinations need transformations (using distributivity)

“the vehicles that are a sport car or small and either a truck or not red” ->
“the sport cars that are not red and the small trucks”
Recasting Descriptions

Techniques

Partitioning a description by descriptors and referents
Simplifications by eliminating non-existing combinations

Example

\{x_5, x_7, x_8, x_{12}\} identified by (sportscar \lor small) \land (truck \lor \neg red)

3 possible partitionings, according to subexpression chosen and objects it covers

1. (sportscar \land (truck \lor \neg red)) \lor (small \land (truck \lor \neg red)) for \{x_{12}\}, \{x_5, x_7, x_8\}

2. (sportscar \land (truck \lor \neg red)) \lor (small \land (truck \lor \neg red)) for \{x_5, x_{12}\}, \{x_7, x_8\}

3. (truck \land (sportscar \lor small)) \lor (\neg red \land (sportscar \lor small)) for \{x_7, x_8\}, \{x_5, x_{12}\}

2. and 3. (not 1.) can be simplified to (truck \land small) \lor (\neg red \land sportscar)
Extension for Sets of Objects - Critique

Consequences of the (incremental) search technique proposed
Solution quality may be very low
(expressions generated may get very complex/redundant)
Strong commitment:
A priori inclusion of structurally simpler combinations
Turns out to be computationally expensive

Methods proposed
Constraint-Based searching (Gardent 2002)
Best-first Searching (Horacek 2003, …)
Issues of Complexity

<table>
<thead>
<tr>
<th></th>
<th>Number of distractors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td>max. search tree size</td>
<td>9</td>
</tr>
<tr>
<td>min. search time (msec)</td>
<td>10</td>
</tr>
<tr>
<td>avg. search time (msec)</td>
<td>116</td>
</tr>
<tr>
<td>max. search time (msec)</td>
<td>490</td>
</tr>
</tbody>
</table>

Extreme Example

“the cars which are not blue, are old or stand in the center, are new or stand on the right side, are big or not white, and are small or not red”

108110 msec, identifying $x_3$, $x_4$, and $x_6$ out of 25 vehicles
Transfer to Database Answers (Ambadi 2018)

Algorithmic Extension of the inductive logic approach

Using boolean extensions incrementally

Complexity may be considerable: 24526 combinations for “Through which states does Mississippi river flow?” (11 extensions, takes 249 secs)

Conceptual use

Combining with single extensional answers (near misses)

Additions

Intensional answer plus single extensional answer

Exceptions

Intensional answer except single extensional answer
Extended Algorithm

Disjunctive descriptions (Extensional answers, KnowledgeBase KB)

1. for all (Extensional Answers) do
2. \( \text{Ans} = \text{ExtensionalAnswers.getNext();} \)
3. \( C = \text{constructClause(Ans, KB);} \quad \triangleright \text{(first answer clause)} \)
4. \( \text{Ans'} = \text{ExtensionalAnswers.getNext();} \)
5. \( C' = \text{constructClause(Ans', KB);} \quad \triangleright \text{(second answer clause)} \)
6. Coverage\((L_i, V_i)\)
7. if Coverage < answers set.size() then
8. \( \text{for } ((L_i, V_i) \text{ in } C) \quad \triangleright L_i, V_i = \text{literal value pair in first clause} \)
9. \( \text{for } ((L_j, V_j) \text{ in } C') \quad \triangleright L_j, V_j = \text{literal value pair in first clause} \)
10. Query = BoolCombQuery\((L_i, V_i, L_j, V_j)\) \quad \triangleright \text{boolean combinations} \\
11. check consistency of the combination
12. if output set = answers set then
13. return (literal, value) pair; \quad \triangleright \text{Intensional answer clause}
Some Examples (Ambadi 2018)

“Which state does Missouri river pass through”?

„All the states that border South Dakota except Wyoming“

“Give me the states that border with Arizona”

„Colorado and all states that are located west and through which Colorado r, Truckee rivers flow“

“What states are next to Oregon?”

“All states which either has
    BorderName as [Arizona, Nevada, Oregon] and
    landelevationName as [borah peak and snake river] or
    has Location as [West] and LakeName as [Tahoe]”

Describes Idaho + (California, Nevada)
Future Research

Algorithmic issues

Improving efficiency of the approach pursued
(use of heuristics, technical improvements)
Investigating the referring expressions approach
(completely unexplored yet)

Psychological/coherence issues

Interpreting psychological insights for database answers
Developing relations between query types and suitable answers components (focus, coherence)
Reusing results for heuristics in the algorithmic part
References


