Constructing and Refining Operationalized Ontologies within Explanation-Capable Agents.

Michael Minock
The University of Umeå, Sweden
mjm@se.umu.se

Abstract

In this paper a technique that enables the dialog-based construction and refinement of operationalized ontologies is proposed. An agent learns classification knowledge through a dialog with a teacher. In this dialog the agent is forced to provide logical explanations of why it classifies (or fails to classify) an instance to various ontology concepts. The teacher is able to give immediate and very focused feedback through these dialogs. This enables the agent to be taught classification knowledge through the treatment of only a handful of examples. And the teacher's feedback is through conversational point, click, and field-filling interactions.

What differentiates this approach is that this work leverages off of, and builds on previous work in which a complete system of concept equations may be solved over a practical subset of relational schemas. The ability to syntactically compute *query difference* enables the calculation of concept intersections and differences, and the determination of concept subsumption and disjointness. The knowledge repair algorithms outlined in this paper require this capability.

Keywords: Knowledge Acquisition, Query Difference Operator, Explanation, Universal Relation.

1 Introduction

Unfortunately real world data does not usually come with class membership data attached. Much real world data simply describes low-level situations or events. To employ similarity-based learning techniques, a large, representative sample of the data must first be manually labeled by an expert. Moreover, if the target concepts are subjective (e.g. `GoodRestaurant` or `BadCreditRisk`), then this labeling phase will need to be performed by multiple individuals to suit their own unique purposes.

Though the idea is not new, it still stands that using *prior knowledge* is helpful in the task of acquiring concrete classification knowledge. Through *operationalizing* such background knowledge, it is possible to teach an agent how to classify events and situations using just a handful of examples.

This is especially pertinent when we consider that *ontologies* will capture more and more high-level background knowledge, but will often capture such knowledge in general forms that will need to be mapped down to the actual databases available in practice.

The notion of an ontology in this paper, is that of a lattice of general and domain concepts and their relationships. These concepts may have logical definitions connecting them to the extensional schema in which instances (facts) are represented. Alternatively the ontology may not have operationalized definitions. In the later case the techniques here attempt to aid in the *construction* of an operationalized ontology. In the former case, these techniques may be employed to *refine* an existing, operationalized ontology.

Let us start this paper out with a very simple example of an agent trying to build the concept of speeding [1]. Assume a simple schema or `record`, along with a target concept `speeding`:

```
report(vin, speed, color, type, date)
speeding(vin, date)
```

This schema also comes annotated, informing us that `vin` (vehicle identification number) and `date` are the keys, and that `vin`, `color`, and `type` are no-orders, and `date` and `speed` are total orders.

Aside from access to this input information, let us assume that the agent is a virtual tableau rosa. There is no knowledge of what speeding is - except for the fact that it is assigned to specific vehicles on specific dates.

```
report(001, 78 mph, white, mazda, 01-01)
```

(1) Agent: *This is not speeding.*
(2) Teacher: *Wrong classification.*
(3) Agent: *OK - This is speeding because the speed is 78 mph, color is white, and the type is mazda.*
(4) Teacher: *The color and type are not relevant to why this car was judged to be speeding.*
(5) Agent: *This is speeding because the speed is 78 mph*
(6) Teacher: *Correct.*

```
report(002, 74 mph, black, BMW, 01-02)
```

(7) Agent: *This is not speeding because the speed is not 78 mph.*
(8) Teacher: *Wrong classification*
(9) Agent: *OK - This is speeding because the speed is less than or equal to 78 mph.*

[1] More complex examples will be discussed below, but this example illustrates the basic approach being proposed here.

(10) Teacher: *No - Wrong condition.*
(11)Agent: *OK - This is speeding because the speed is greater than or equal to 74 mph.*
(12)Teacher: *Correct.*

`report(003, 60 mph, brown, Ford, 01-03)`

(13) Agent: *This is not speeding because the speed is less than 74 mph.*
(14) Teacher: *Correct - but you mean 65 mph*

`report(004, 90 mph, white, Police, 01-03)`

(15) Agent: *This is speeding because the speed is greater than 65 mph.*
(16) Teacher: *Wrong classification*
(17) Agent: *OK - This is not speeding because the the color is white, and the type is Police Car and Speed is 90 mph.*
(18) Teacher: *The color and the speed are not relevant to why this car was judged not to be speeding.*
(19) Agent: *OK - This is not speeding because the type is Police.*
(20) Teacher: *correct.*

(21) Agent: *Is it always the case that it is not speeding when speed is less than 65 mph?*
(22) Teacher: *Yes.*
(23) Agent: *Is it never the case that it is speeding when car type is police?*
(24) Teacher: *Yes.*
(25) Agent: *Is it always the case that it is speeding when car type is not police and the speed is greater than 65 mph?*
(26) Teacher: *Yes.*

(27) Agent: *I fully understand the concept of speeding.*

1.1 Organization of this paper

Section 2 illustrates the basic approach through showing the workings of the agent over the example dialog above. Section 3 discusses scale up to larger domains. Section 4 discusses the formal under-pining of this work - the *Query Difference Operator*. Section 5 discusses this work in relation to some prior and ongoing work in knowledge acquisition and ontology learning. Section 6 proposes the further plans for the development and refinement of this system.

2 The Approach

2.1 The Conversational Flow

There is a semi-rigid protocol that is observed between the teacher and the agent. Teacher input is constrained to be in a very simple form. This obviates the need for sophisticated front-end interpretors over the teachers input and feedback. In addition all expressions that the agent produces are in natural language. This is possible because the representations within the agent are forms from which it is relatively easy to generate lucid, non-ambiguous natural language.

Figure 1 shows the conversational flow between the agent and the teacher. There are three distinct phases of the conversation. The first phase is the *observation phase* and it always occurs. In this phase an example instance and target concept are presented to the agent. The agent calculates whether the example is a member of the target concept and then provides a positive (or negative) minimal explanation of its classification back to the teacher. If the teacher agrees with the classification decision and also finds the explanation adequate, then the process concludes and the agent will await the next trial.

If the teacher disagrees with the classification, then the agent is informed of this with a disagreement indication. This opens the *correction phase* of the conversation. In this phase the teacher directs the agent to repair its knowledge structures and to then provide a new (correct) classification and explanation based on the repaired knowledge structure. This, when the teacher has been less than consistent in prior training, may

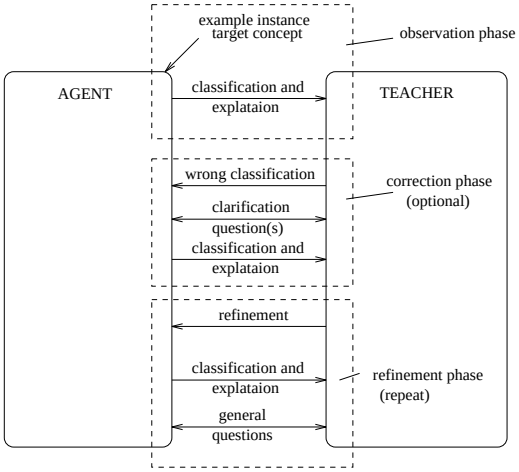

Figure 1: Conversational Flow

generate a (set of) clarifying question(s) from the agent. After this phase the agent will be classifying the example properly[2] and the *refinement phase* is entered.

Upon entering the refinement phase, the agent will be properly classifying the example but it may be relying on irrelevant conditions in the example to perform the deduction. The explanation provided from the observation or correction phase starts out the interaction. The teacher may identify conditions in the explanation which are irrelevant or incorrect. The teacher may provide repairs to incorrect conditions or simply let the agent search out an alternative condition. Alternatively the teacher may request that a deeper explanation, with more conditions, be provided. During the refinement phase the agent may also ask general questions about the domain. These true/false questions, if answered by the teacher, enable the agent to further constrain the hypothesis space. At the end of the refinement phase the agent will still be able to account for the example (and past examples) but will, through, interaction with the teacher, have more specific, closer to the truth, knowledge structures. This type of confidence building through dialog takes us a step toward more complete and consistent ontologies[3].

2.2 Hypothesis Space Representation

The approach here is a mix of version-spaces[8], explanation-based learning[9][10], and description logics[1] inspired relational systems[7]. Let us illustrate the basic representations and processing by treating the example above.

As mentioned in the introduction, the knowledge input to the system is a schema, and a set of target concepts that should be learned. In this case the relation `report(vin, speed, color, type, date)` plus the concept `speeding(vin, date)` is provided. In addition we know whether an attributes domain is either a *total-order* or a *no-order*. Finally we have functional dependency knowledge that `vin` and `date`, together, functionally determine all the other attributes.

[2]Unless, in the face of the clarifying questions, the teacher decides to abort the trial.

We associate the target concept c_i to be learned with three expressions:$C_i^{\perp}$,$C_i^{\top}$, and C_i. As is customary, $C_i^{\perp}$ indicates the logically most specific possible version of c_i, $C_i^{\top}$ the most general possible version of the concept, and C_i the current, best guess, as to what c_i is.

Note that all of these expressions are in the form that will be defined in section 4. The important point to note here is that these expressions resemble relational algebra (over a universal relation [2]) and are closed under a difference operator. From this it follows that we are able to compute conceptual differences, intersections, and compliments. In addition we are able to compute subsumption, disjointness, and concept equivalence. The operations of the agent, sketched in section 2.4, require these capabilities.

The following tables report the contents of the agent's knowledge-base after each step in the dialog[3]. Note that the j-th instance added is t_j.

First we consider the knowledge the agent has about $C_1^{\top}$ and $C_1^{\perp}$ through the dialog.

	$C_1^{\top}$	$C_1^{\perp}$
0	$\pi_{vin,date}$	$\emptyset$
1	$\pi_{vin,date} \ominus \pi_{vin,date}\{t_1\}$	$\emptyset$
3	$\pi_{vin,date}$	$\pi_{vin,date}\{t_1\}$
7	$\pi_{vin,date} \ominus \pi_{vin,date}\{t_2\}$	$\pi_{vin,date}\{t_1\}$
9	$\pi_{vin,date}$	$\pi_{vin,date}\{t_1,t_2\}$
13	$\pi_{vin,date}\ominus$ $\pi_{vin,date}\{t_3\}$	$\pi_{vin,date}\{t_1,t_2\}\ominus$ $\pi_{vin,date}\{t_3\}$
15	$\pi_{vin,date}\ominus$ $\pi_{vin,date}\{t_3\}$	$\pi_{vin,date}\{t_1,t_2,t_4\}\ominus$ $\pi_{vin,date}\{t_3\}$
17	$\pi_{vin,date}\ominus$ $\pi_{vin,date}\{t_3,t_4\}$	$\pi_{vin,date}\{t_1,t_2\}\ominus$ $\pi_{vin,date}\{t_3,t_4\}$
22	$\pi_{vin,date}\ominus$ $\pi_{vin,date}\sigma_{speed<65}\ominus$ $\pi_{vin,date}\{t_4\}$	$\pi_{vin,date}\{t_1,t_2\}\ominus$ $\pi_{vin,date}\sigma_{speed<65}\ominus$ $\pi_{vin,date}\{t_4\}$
24	$\pi_{vin,date}\ominus$ $\pi_{vin,date}\sigma_{speed<65}\ominus$ $\pi_{vin,date}\sigma_{type=Police}$	$\pi_{vin,date}\{t_1,t_2\}\ominus$ $\pi_{vin,date}\sigma_{speed<65}\ominus$ $\pi_{vin,date}\sigma_{type=Police}$
26	$\pi_{vin,date}\ominus$ $\pi_{vin,date}\sigma_{speed<65}\ominus$ $\pi_{vin,date}\sigma_{type=Police}$	$\pi_{vin,date}\sigma_{speed\geq 65}\ominus$ $\pi_{vin,date}\sigma_{type=Police}$

As we can see, until the agent issues the general questions to the teacher, there is little, other than the actual instances themselves, that constrain the hypothesis space. The equivalence of $C_1^{\top}$ and $C_1^{\perp}$ is established by verifying that $C_1^{\top} \ominus C_1^{\perp} = \emptyset$ and $C_1^{\perp} \ominus C_1^{\top} = \emptyset$. Section 4 discusses how this is achieved. The equivalence of $C_1^{\perp}$ and $C_1^{\top}$ at step 26 licenses the agent to make the statement at step 27.

Next we show the calculated guess that the agent has about the actual concept C_1. We will discuss below how the system arrives at these guesses below. Note that these guesses are always legal. That is $C_1 \supseteq C_1^{\perp}$ and $C_1^{\top} \supseteq C_1$.

	C_1
0	$\emptyset$
1	$\emptyset$
2	$\pi_{vin,date}\{t_1\}$
3	$\pi_{vin,date}\sigma_{speed=78 \wedge color=white \wedge type=mazda}$
4	$\pi_{vin,date}\sigma_{speed=78}$
8	$\pi_{vin,date}\sigma_{speed=78} \oplus \pi_{vin,date}\{t_2\}$
9	$\pi_{vin,date}\sigma_{speed<78}$
10	$\pi_{vin,date}\sigma_{speed>74}$
13	$\pi_{vin,date}\sigma_{speed>74}$
14	$\pi_{vin,date}\sigma_{speed>65}$
15	$\pi_{vin,date}\sigma_{speed>65}$
16	$\pi_{vin,date}\sigma_{speed>65} \ominus \pi_{vin,date}\{t_4\}$
17	$\pi_{vin,date}\sigma_{speed>65} \ominus \pi_{vin,date}\sigma_{color=white \wedge \ldots}$
18	$\pi_{vin,date}\sigma_{speed>65} \ominus \pi_{vin,date}\sigma_{type='Police'}$

[3]Operations are applied from left to right. $A \ominus B \ominus C \equiv (A \ominus B) \ominus C$

2.3 Relevance Representation

There is an additional data structure R_i associated with each concept. This structure records the relevance feedback that has been gathered on the concept. This structure consists of a set of relevant attribute sets paired with query expressions. This indicates that the variables within the set are relevant to determining the concept c_i when the instance falls under the associated query expression. The following table shows the contents of R_1 for the example in the introduction.

	R_1
0	$\emptyset$
4	$\{speed\} : \sigma_{color=white,type=mazda,speed=78}$
12	$\{speed\} : \sigma_{color=white,type=mazda,speed=78}\oplus$ $\sigma_{color=black,type=BMW,speed=74}$
14	$\{speed\} : \sigma_{color=white,type=mazda,speed=78}\oplus$ $\sigma_{color=black,type=BMW,speed=74}\oplus$ $\sigma_{color=brown,type=Ford,speed=60}$
17	$\{speed\} : \sigma_{color=white,type=mazda,speed=78}\oplus$ $\sigma_{color=black,type=BMW,speed=74}\oplus$ $\sigma_{color=brown,type=Ford,speed=60}\ominus$ $\sigma_{color=white,type=Police,speed=90}$
18	$\{speed\} : \sigma_{color=white,type=mazda,speed=78}\oplus$ $\sigma_{color=black,type=BMW,speed=74}\oplus$ $\sigma_{color=brown,type=Ford,speed=60}\ominus$ $\sigma_{color=white,type=Police,speed=90}$ $\{type\} :$ $\sigma_{speed=90,color=white,type=Police}$

Note that this structure is a logical consequence of the operations carried out by the teacher.

2.4 Operations

The teacher, who guides the sessions, is able to apply operations. The structures C_i, $C_i^{\top}$, $C_i^{\perp}$, and R_i are accessed and altered during these operations.

Operations of the Observation Phase

The `Classify` operation simply tests whether the `instance` meets the current definition of the `concept`. If it does then the minimal set of attribute values that make it so are reported. This is the *positive explanation* for why the instance is a member of the concept. If the instance is not a member of the concept then the minimal set of attribute values, which, if changed, would make it a member of the concept, are reported. This is the *negative explanation* for why the instance is not a member of the concept. These techniques are described in [6]. Explanation services for general description logic systems are discussed in [5]. The results from this operation are shown in every agent utterance before step 21 in the introductory dialog.

Operations of the Correction Phase

The operation `WrongClassification` starts the correction phase in which the agent's conceptual structures are repaired. If the disagreement is with a positive explanation, then the initial repair is to subtract the instance from all three C_i, $C_i^{\top}$, and $C_i^{\perp}$ structures (e.g. step 16). If the disagreement is with a negative explanation then the initial repair is to add the instance to the three structures (e.g. steps 2 and 8). In either case for the C_i expression the agent immediately adopts a value-oriented view of the instance, expressing the instance as a conjunction of conditions, one for each *relevant* attribute. The relevance of an attribute is decided by consulting the relevance structure R_i. See *heuristic #2* below to see how relevant attributes are decided. This produces an

immediate generalized repair to the structure. However this repair will be generalized only enough to apply to the current instance, so re-testing prior instances is not necessary.

Note that contradictions are possible when the teacher has given inconsistent responses. However this is only the case when the agent has issued a set of general questions to the teacher to constrain $C_i^\top$ and $C_i^\perp$. When a correction violates the boundaries of the $C_i^\top$ or $C_i^\perp$ the agent will issue a (set of) clarifying question(s) to re-extend the boundary.

Operations of the Refinement Phase

In all the operations of the refinement phase, the structure C_i is altered. If it is generalized then it is necessary to re-test all of the prior negative examples. If it is specialized then all prior positive examples need to be re-tested. By reasoning over the concept and relevance expressions one does not, in practice, need to do this full calculation at each step. Still, logically, the new structure must account for all past examples. In the case of a conflict the operation is disallowed, along with an explanation of why. Refinements that carry C_i outside the boundary determined by $C_i^\top$ and $C_i^\perp$ generate clarifying questions to the teacher to ask if the boundary may be extended.

The operation `Irrelevant` identifies when and where an explanation is using an irrelevant `condition`. In the case of a positive explanation, the condition is removed (e.g. step 4) from C_i. In the case of a negative explanation, `Irrelevant`, we remove conditions in a way similar to the case for positive explanations, but in this case for terms which are being subtracted within C_i (e.g. step 18).

The operation `Wrong` identifies a condition that has been generalized improperly (e.g. step 10). This then precipitates an alternative generalization strategy to be applied to the condition. See *heuristic #3* below. The operation `Change` lets the teacher directly alter a constant value within a condition in the knowledge structure (e.g. step 14).

The operation `MoreReasons` is a call for more specificity around why the instance was (or was not) classified to the concept. In either case this leads to the full (i.e. all conditions) value-oriented version of the instance to be either added to or subtracted from C_i.

After the new version of C_i is confirmed consistent with all the prior examples, the `Classify(instance)` operation is re-run and the correct classification and its explanation are reported to the teacher. This may in turn result in another round of refinement.

Heuristics

The above operations employ the following heuristics:

Heuristic #1: The values of keys are not relevant to classifying the instances. Though this heuristic could easily be discarded, it seems to hold for many potential domains.

Heuristic #2: A relevant attribute of an instance is an attribute whose query expression in R_i does not preclude the instance. If no relevant attributes may be determined, then all of the non-key attributes are deemed relevant. The intuition here is that the repairs in the correction phase must only apply to the current instance, yet should be generalized, and should exploit prior determinations of relevant attributes. Using this the agent concludes that all the non-key attributes are relevant in step 2, only speed after step 8, and again all non-key attributes after step 16.

Heuristic #3: For total orders, if more than one value is true (or false) the agent will attempt to generalize the condition over the entire domain of the total order. Currently the agent (arbitrarily) decides to generalize by assuming that the attribute is $\leq$ to the highest element in the set of known values (e.g. step 9). If this fails the agent generalizes by assuming that the attribute is $\geq$ to the lowest element in the set (e.g. step 11). Next the agent attempts to generalize over the range from the lowest to the highest. If all these generalizations fail, then the agent falls back to just known values.

3 Scale-up

3.1 Multiple concepts and Hierarchical Knowledge

In the presence of already known concepts we convert the concept values to boolean attributes on new instances. The system will then neglect the attributes that were deemed relevant to decide the known concept. Of course the teacher may force a more specific analysis that include the concept variables. Still this initial treatment helps promote the building of hierarchical knowledge structures in the agent. Note that this makes the order of the concepts presented to the agent very important - where each concept builds on the subsequent ones. Techniques from concept exploration[11] may be the key to managing this complexity.

3.2 Incorrect Guidance

The above system can accommodate an example where the teacher gives incorrect or misleading instruction. The approach is simply to state the conflict to the teacher. The response of the teacher decides how and if the knowledge structure is repaired.

3.3 More Expressive Knowledge Representations

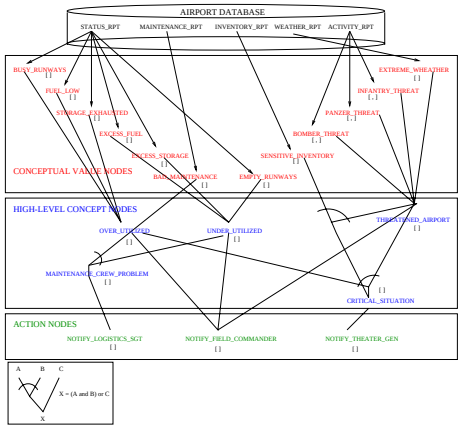

Figure 2: Knowledge structure of limited, though practical, expressivity.

It may be that this approach in this paper is only possible within knowledge representations of the form described below. Still, even within these systems, interesting and practical ontologies may be operationalized. Work is currently be

MOVIE(title, type, rating, year)

REVIEW(title, source, eval)

THEATER(theater, address, city)

SHOW(title, theater , time, price)

Table 1: The MOVIE schema.

conducted to employ these methods to teach agents structures such as the one within figure 2.

4 The Query Difference Operator

In an earlier paper[7] it is shown that the set difference of the queries q_1 and q_2 may be computed as a syntactic manipulation of the expressions q_1 and q_2 for a well defined subset of the relational algebra over a restricted class of relational schemas. With this, one may, without materializing data, take the expressions for q_1 and q_2, apply the query difference formula to yield q_3, and be guaranteed that q_3 is logically equivalent to $q_1 - q_2$. With query set difference handled, the ability to compute query intersection, subsumption, disjointness, and equivalence follow.

To illustrate let us take as an example the movie schema in table 1. Consider q_1 being the query *"Give the show times in Austin for G or PG-13 movies with 4 or 5 star evaluations"* and q_2 being the query *"Give the show times in Austin or San Antonio for PG-13 or R movies with 3 or 4 star evaluations."* The logical result of $q_1 - q_2$ is the query *"Give show times in Austin for G movies with 4 or 5 star evaluations plus show times in Austin for PG-13 movies with 5 star evaluations."* Similarly, $q_1 \cap q_2$ is *"Give show times in Austin for PG-13 movies with 4 star evaluations."* Clearly neither query subsumes the other, nor are they equivalent.

4.1 Query and Schema Representation and Restrictions

Assume a set of relations $\{R_1, ..., R_m\}$ where we have, a priori, decided on a set of equi-joins that connect these relations together so that there are no cycles in the resulting graph. In the example in table1 we pick the equi-join between `MOVIE` and `SHOW` through the attribute `title`, the equi-join of `SHOW` and `THEATER` through the attribute `Theater`, and the equi-join of `MOVIE` and `REVIEW` through `title`. When we outer join all the relations together, we get the universal relation R.

A query is represented as an expression of the form $\Pi_X \sigma_{c_1 \wedge ... \wedge c_n}$ where X identifies a set of sub-relations from R. The conditions $c_1, ..., c_n$ are simple, non-join conditions. For example $\Pi_{Show} \sigma_{city=Austin}$ retrieves the show-times for those movies playing in Austin.

Differences with the standard Relational Algebra

Though these queries have a similar appearance and semantics to simple relational algebra, they differs in several key ways. (1) The "relation" that these queries are applied over is always R, the outer join of all the relations $\{R_1, ..., R_m\}$. Because this is always the case, 'R' is omitted from the notation. And because of this single implied relation argument, self-join queries are precluded. (2) The projection sets in queries here are sub-relations of the universal relation R. In the Relation Algebra projections are over sets of attributes – this requirement could easily be relaxed, but it helps simplify some applications. (3) We extend the notion of difference and union over what in the standard relational algebra are non-union compatible projection sets. The expression $q_1 \oplus q_2$ indicates this extended form of union. In the standard case $\Pi_{Movie} \sigma_{city=Austin} \oplus \Pi_{Movie} \sigma_{city=LA}$ is equivalent to $\Pi_{Movie} \sigma_{city=Austin} \cup \Pi_{Movie} \sigma_{city=LA}$. But in the language here you can also express queries like $\Pi_{Movie} \sigma_{city=Austin} \oplus \Pi_{Playing\_at} \sigma_{city=Austin}$. The same extension occurs for the difference operator. In Relational Algebra this again requires that queries have equivalent projection sets. Not so here, the operator $\ominus$ expresses this extended notion of query difference. For example $\Pi_{Movie} \sigma_{city=Austin} \ominus \Pi_{Movie,Show} \sigma_{city=Austin}$ yields the empty set. (4) Finally there is a special negation semantics of such queries where the query $\Pi_{Movie} \sigma_{city <>' Austin'}$ is not equivalent to the query $\Pi_{people} \sigma_{\neg(city =' Austin')}$. The former query gives the Movies which play somewhere other than Austin. The later query expresses the query giving the movies that do not play anywhere in Austin. This later query is expressing a form of universal quantification through a not-exists constraint.

4.2 Algebraic Theorems

Here follows the central theorem that we may use under the assumptions above. The query difference formula says that for the simple queries of the type proposed, one may solve for query difference by manipulating query expressions directly.

Theorem 1 *(Query Difference Formula)*

$$\Pi_X \sigma_{c_1 \wedge ... \wedge c_n} \ominus \Pi_Y \sigma_{c'_1 \wedge ... \wedge c'_{n'}} = \Pi_{X \cap Y} \sigma_{c_1 \wedge ... \wedge c_n \wedge \neg c'_1} \oplus ... \oplus \Pi_{X \cap Y} \sigma_{c_1 \wedge ... \wedge c_n \wedge \neg c'_{n'}} \oplus \Pi_{X - Y} \sigma_{c_1 \wedge ... \wedge c_n}$$

Let us consider some examples of this operator in use: First consider, $\Pi_{Movie} \sigma_{city=Austin} \ominus \Pi_{Movie,Show} \sigma_{city=Austin} = \Pi_{Movie} \sigma_{city=Austin \wedge \neg(city=Austin)} = \emptyset$. Here we have a non-union compatible difference that yields the an empty expression. In the expression, $\Pi_{Movie} \sigma_{year>1950} \ominus \Pi_{Movie} \sigma_{year \geq 1960} = \Pi_{Movie} \sigma_{year>1950 \wedge year<1960}$. Here we have a simple union compatible query that yield the movies made in the fifties.

$\Pi_{Movie} \sigma_{city =' LA'} \ominus \Pi_{Movie} \sigma_{city <>' LA'} = \Pi_{Movie} \sigma_{city =' LA' \wedge \neg(city <>' LA')}$ Notice that in this query that what we obtain are the movies that are playing only in LA. This illustrates that rules govern when conditions may be simplified. The approach is to apply the $\neg$ operator to the condition c (thus changing an '=' to '<>', or '<' to a '>=', etc.) if and only if the condition's attribute is functionally dependent on the key of each sub-relation in the projection set, and the attribute is not a proper subset of a candidate key.

Theorem 2 *(Query Difference is distributive over compound queries)*

$$(q_1 \oplus q_2) \ominus (q_3 \oplus q_4) = ((q_1 \ominus q_3) \ominus q_4) \oplus ((q_2 \ominus q_3) \ominus q_4)$$

This theorem enables us to apply the difference formula over compound queries. This is necessary to compute intersections. Intersection follows from $q_1 \cap q_2 = q_1 \ominus (q_1 \ominus q_2)$.

Note that subsumption follows from $q_1 \supseteq q_2 \Leftrightarrow q_2 \ominus q_1 = \emptyset$ and equivalence follows from $q_1 \equiv q_2 \Leftrightarrow q_2 \ominus q_1 = \emptyset \wedge q_1 \ominus q_2 = \emptyset$. Queries are disjoint if their intersection is empty.

The following three theorems assist in the simplification of query expressions. In many applications queries will be conveyed to users. No matter the method (e.g. Natural language, graphical display, etc.), we should like to minimized the number of terms in the expression. These three theorems give a complete method by which to search for simpler, but equivalent, query expressions.

Theorem 3 *(Absorption)* $\Pi_X \sigma_C \oplus \Pi_Y \sigma_{C'} = \Pi_Y \sigma_{C'}$ *if* $C \Rightarrow C'$ *and* $X \subseteq Y$.

Theorem 4 *(Horizontal Merge)* $\Pi_X \sigma_C \oplus \Pi_Y \sigma_C = \Pi_{X \cup Y} \sigma_C$

Theorem 5 *(Vertical Merge)* $\Pi_X \sigma_{C \wedge c_i} \oplus \Pi_X \sigma_{C \wedge c_a} = \Pi_X \sigma_{C \wedge c_k}$ *where* $c_i \vee c_a$ *may be written as the simple condition* c_k.

5 Previous and contemporary work

The work here follows in the same spirit as propose-and-revise systems such as SALT system[4]. Approaches such as Expect[13] also relate - especially in the quest to facilitate knowledge acquisition through explanatory dialogues. However the focus here is on acquiring knowledge of a much simpler form than systems acquiring knowledge for expert systems. The simplicity of knowledge representation leads, however, toward an acquisition tool that may be efficiently instructed on any (or many) possible domain specific 'views' over a universal relational.

This work shares some goals with[12] in that it may start with a simple attribute-value structure (a Universal Relation[2] in the case here), and build an immediate problem solver, while keeping as a further goal the extraction of a reusable ontology from the efforts. In contrast to building the problem solver from ripple down rules, the approach here builds a structured, though not highly expressive problem solver as a by-product of only a few instances of the problem. The teacher identifies relevance information by interacting with agent explanations.

6 Future plans

A prototype of this system exists, however this prototype needs to be further developed. The current system is able to account for dialogs such as the one described in the introduction. Still there are some further heuristics that need to be added to reduce instance order sensitivity and to properly trigger the agent to ask general questions of the teacher. But work continues and progress is being made. I look forward to receiving feedback from the community to focus and guide the scale-up of this prototype.

In addition I will be investigating how these techniques may applied over widely used ontology representations (OIL, F-Logic, OKBC, etc.). Once a suitable match is found this prototype will be ported to that representation language and larger scale experiments with ontology operationalization will be undertaken.

7 Conclusion

In this paper proposed an approach towards the complete acquisition of conceptual knowledge from a teacher. At its heart the approach relies on the query difference operator to solve concept equations.

8 Bibliography

References

[1] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Studies in Logic, Language and Information*, pages 193–238, 1996.

[2] R. Fagin, A. Mendelzon, and J. Ullman. A simplified universal relation assumption and its properties. *ACM Transactions on Database Systems, 7*, 1982.

[3] A. Gomez-Perez. Evaluation of taxonomic knowledge in ontologies and knowledge bases. *Proceedings of KAW'99*, 1999.

[4] S. Marcus and J. McDermott. Salt: A knowledge acquisition language for propose-and-revise systems. *Artificial Intelligence 39 (1)*, 1989.

[5] D. McGuinness. Explaining reasoning in description logics. phd thesis. *Department of Computer Science, Rutgers University*, 1996.

[6] M. Minock and W. Chu. Explanation over inference hierarchies in active mediation applications. *Journal of Applied Intelligence*, 13:101–112, 2000.

[7] Michael Minock, Marek Rusinkiewicz, and Brad Perry. The identification of missing information resource agents by using the query difference operator. In *COOPIS '99*, Edenburgh, Scotland, September 1999. IEEE Computer Society Press.

[8] T. Mitchell. Version spaces: A candidate elimination approach to rule learning. *IJCAI*, pages 305–310, 1977.

[9] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: a unifying view. Technical report, SUNJ Rutgers U, 1985.

[10] Michael J. Pazzani. Integrating explanation-based and empirical learning methods in occam. *EWSL*, pages 147–165, 1988.

[11] G. Stumme. Concept exploration - a tool for creating and exploring conceptual hierarchies. *In Proceedings of the 5th International Conference on Conceptual Structures, Seattle, USA.*, 1997.

[12] Hendra Suryanto and Paul Compton. Learning classification taxonomies from a classification knowledge based system. In *ECAI 2000, Workshop on Ontology Learning. Berlin, Germany.*, 2000.

[13] B. Swartout and Y. Gill. Flexible knowledge acquisition through explicit representation of knowledge roles. *In 1996 AAAI Spring Symposium on Acquisition, Learning, and Demonstration: Automating Tasks for Users.*, 1996.