Toward Question Answering in Travel Domains: locations of participants and cardinalities of groups

Yana Todorova and Michael Gelfond

February 12, 2012

Abstract

The paper gives Answer Set Prolog axiomatization of knowledge needed to answer questions about the content of simple travel stories. We assume that the background knowledge needed to understand travel stories contains a hierarchy of classes relevant to it and concentrate on answering questions about locations of movers and cardinalities of different groups of movers. The axiomatizations allow to properly answer such questions for stories with various forms of incompleteness.

1 Introduction

This paper belongs to the line of work, which investigates the applicability of Answer Set Programming (ASP) [9, 15, 17] to question answering from natural language ([2]). We are interested in reasoning about simple travel stories, with the emphasis on the locations of participants and the cardinality of groups, located in different areas at different stages of the travel. The previous work [3, 4] on answering questions about travel in ASP was able to answer rather sophisticated questions about locations of participants, given complete information about the travel domain. No attempt, however, was made to deal with cardinality and with stories containing only partial information. In this work, we remove these restrictions. In addition, we assume that the background knowledge, needed to understand travel stories, contains a hierarchy of classes1 of movable objects described by the stories. Movement of one such object into some area changes not only its position, but also the cardinality of the sets of objects from different classes located in this (and other) areas. Thus, this addition adds an extra level of complexity, which was not present in the previous work. To better understand the problem we present several examples, which will be used throughout the paper for illustrative purposes.

Example 1 [Basic Travel Story]

Professor D and two of his students, A and B, entered the empty room. They were immediately followed by professor C. Is A in the room? How many people are in the room? How many students? How many professors?

The hierarchy needed to understand this story and the relevant questions consists of the classes person, professor, and student with natural subclass relations between them. The story describes a trajectory of a discrete dynamic system2. Initially, the system is in the state in which the room is empty. After the first action, the state changes. Now the room contains A, B, and D. After the second action, they are joined by C. The questions, which refer to the last state of the trajectory, are answered by yes, 4, 2 and 2, respectively. Since the story gives complete information about locations of people and their positions in the inheritance hierarchy, the cardinality questions are answered by a single number each. A small modification of existing axioms was sufficient to find the desired trajectory and to answer the

1Here, by hierarchy we mean a collection of class names connected by arrows, which indicate the subclass relation between the classes; instance of a hierarchy is a set of objects called the universe of the hierarchy, and a mapping of class names into subsets of the universe, which respects the hierarchy’s subclass relation.

2We assume that all the possible trajectories of such system are described by a transition diagram, whose nodes represent possible states of the system, and whose directed arcs are labeled by actions.
corresponding questions. The next example shows that the situation changes when we do not have a complete knowledge about our domain.

Example 2 [Uncertain group affiliation]
Professor D and two of his students, A and B, entered the empty room. They were immediately followed by professor C. How many freshmen are in the room?

The hierarchy of classes, pertinent to the story from Example 2, includes classes freshman, sophomore, junior, and senior which are subclasses of the class student. In contrast to the previous example, we do not have a complete knowledge about classes to which the entities discussed in the text belong to (in particular, we do not know if A and B are freshmen). This implies that there is a number of trajectories compatible with the story. Analysis of these trajectories allows us to produce an expected answer: at most two. The next example illustrates another type of incompleteness. This time, we do not even know the exact number and names of people, initially located in the room.

Example 3 [Uncertainty in the initial sizes of groups]
There were less than two people in the room. How many people are in the room, after it has been entered by a student named John?

Again the story is compatible with multiple trajectories. Their analysis allows us to produce the correct answer: between 1 and 3. Finally, the following example:

Example 4 [Uncertainty in the number of movers]
There were two professors and four students in the room. They were joined by at least two more students. How many professors, students, freshman, and people are in the room?

The story only specifies the lower bound of people of a given class involved in the movement. There are multiple models of this story, containing multiple moving objects and multiple trajectories, but the questions can be uniquely answered by 2, between 6 and max_size, between 0 and max_size, and between 8 and max_size.

The above stories have several characteristic features. All of them describe trajectories of some dynamic system and are concerned with questions about positions of objects and about cardinality (number of objects from a class of some inheritance hierarchy located in a given area). The last three stories also deal with various types of incompleteness of knowledge. Traditionally, these features cause difficulties for the reasoning parts of existing question answering systems. The method discussed in this paper is able to overcome these difficulties in the context of simple travel stories. We hope that this method will be applicable to more general domains.

The paper is organized as follows. In the next section, we introduce a formal representation of a text and a question (often referred to as a text-question pair), which would serve as an input to our reasoning system. This is followed by a section discussing question answering in the situation when the story contains all the relevant information. After that, we present three sections dealing with travel stories with different types of incompleteness.

2 Travel Logic Form

In this paper we follow a logic based approach to question answering, in which a text-question pair \((T, Q)\) is translated into a suitable logical representation, called logic form of \((T, Q)\). The logical representation (sometimes together with the background knowledge) is then used to find answer to the question. A specific feature of our approach is the use of a particular logic form, suitable for reasoning about travel stories. We refer to it as travel logic form (TLF).

Definition 1 [TLF]
A TLF consists of:
1. Description of the relevant hierarchy:
   (a) A tree of classes:
   \[
   \text{root}(C) \quad \text{class}(C) \quad \text{c.link}(C_1, C_2)
   \]
   The last statement says that in the class hierarchy of the story, class \(C_1\) is a child of class \(C_2\);
   (b) An instance of the hierarchy:
   \[
   \text{object}(O) \quad \text{m.link}(O, C)
   \]
   The last statement places an object \(O\) into a class \(C\) of the hierarchy;
   (c) Cardinality of the universe:
   \[
   \text{card} = k
   \]

2. Collection of areas:
   \[
   \text{area}(A)
   \]
   We refer to the part of logic form consisting of statements described above as static.

3. History of the travelers’ domain described by the story, which is represented by collection of state-
ments of the form:
   \[
   \text{hpd}(a, i) \quad \text{obs}(f, v, i)
   \]
   where \(v\) is a boolean value; the first statement means that action \(a\) happened at step \(i\) of the system’s trauma;
   the second one states that at step \(i\) fluent \(f\) was observed to have truth value \(v\). (For simplicity we restrict ourselves to simple histories, which only contain observations of fluents made
   at time 0).
   The actions of a simple travel domain are of the form
   \[
   \text{enter}(O, A)
   \]
   read as object \(O\) enters area \(A\), and
   \[
   \text{enter}(X, N, C, A)
   \]
   where \(X\) is min, max, or equal, meaning that minimum, maximum, or exactly \(N\) members of class
   \(C\) entered area \(A\). Similarly for actions leave. The fluent
   \[
   \text{in}(O, A)
   \]
   read as object \(O\) is located in area \(A\) describes a property of an individual object. Other fluents
   describe properties of populations — groups of object of class \(C\) located in area \(A\). We will need:
   \[
   \text{lower-bound}(P, N)
   \]
   \[
   \text{upper-bound}(P, N)
   \]
   read as \(N\) is the lower (upper) bound of the size of population \(P\). Action \(\text{enter}(O, A)\) and fluent
   \(\text{in}(O, A)\) will be called basic. Other actions and fluents will be called extended.

4. Collection of queries of the form:
   \[
   \text{query}(\text{in}(O, A)) \quad \text{Is object } O \text{ in the area } A?\]
   \[
   \text{query}(\text{where}(O)) \quad \text{Where is } O?\]
   \[
   \text{query}(\text{who}(A)) \quad \text{Who is located in area } A?\]
   \[
   \text{query}(\text{how many}(C, A)) \quad \text{How many members of class } C \text{ are in } A?\]
   The questions are referring to the values of the fluents at the end of the trajectory.

---

\(^{3}\)Fluent is a property of the domain, whose truth value can be changed by actions. Other properties are called statics.

\(^{4}\)Note that in our logic program below, we will use 0 for min, 1 for equal, and 2 for max.
A TLF can be viewed as a theory defining a collection of models. To define this notion, let us consider a travel logic form LF. By $H(LF)$ we denote the hierarchy described by statements (1a) of LF. An instance, $I(LF)$, of this hierarchy consists of the universe $U$ of cardinality $k$ containing all the objects defined in LF, and a mapping from classes of $H(LF)$ to subsets of $U$. The mapping is such that, for every statement $c_{\text{link}}(c_1, c_2)$ of LF, class $c_1$ is mapped into a subset of $c_2$ and the root is mapped into $U$. By encoding of $I$ with the universe $U$, we mean the set

$$e(I) = \{\text{object}(o) : o \in U\}.$$  

A model of LF consists of an instance $I$ of the LF’s hierarchy, a transition diagram $T$ (describing effects of actions $\text{enter}(O, A)$ and $\text{leave}(O, A)$ on locations of movers from $I$), and a collection $P$ of paths of $T$, compatible with the TLF’s history. The $T$ and $P$ will be defined by an instance $I$ and an ASP program, $M^n$, where $n$ is the maximum number of steps in the story’s history. The program contains general knowledge about travel and about inheritance hierarchies.

**Definition of $M^n$:** The program consists of the following rules.

**Fluents and steps:**

$$step(0...n),$$

$$\text{fluent}(\text{inertial}, \text{in}(O,A)).$$

(Here and below, we use possibly indexed variables $O$, $A$, and $I$ for objects, areas, and steps of the system’s trajectory, respectively; $n$ is the length of the story’s history);

**Dynamic causal law:**

$$\text{holds}(\text{in}(O,A), I + 1) \leftarrow \text{occurs}(\text{enter}(O,A), I).$$

which says that $\text{in}(O,A)$ is a direct effect of action $\text{enter}(O,A)$;

**State constraint:**

$$\neg \text{holds}(\text{in}(O,A_2), I) \leftarrow \text{holds}(\text{in}(O,A_1), I),$$

$$A_1 \neq A_2.$$  

which says that an object can not be located in two disjoint areas;

**Executability condition:**

$$\neg \text{occurs}(\text{enter}(O,A), I), \quad \text{holds}(\text{in}(O,A), I).$$

which says that it is impossible for $O$ to enter area $A$, if it is already there;

**Inertia axioms:**

$$\text{holds}(F,I + 1) \leftarrow \text{fluent}(\text{inertial}, F),$$

$$\text{holds}(F,I),$$

$$\neg \text{holds}(F,I + 1) \leftarrow \neg \text{fluent}(\text{inertial}, F),$$

$$\neg \text{holds}(F,I),$$

$$\neg \text{holds}(F,I + 1)$$

which say that inertial fluents tend to stay unchanged;

**Membership of the class hierarchy:**

$$\text{member}(O,C) \leftarrow \text{m}_{\text{link}}(O,C),$$

$$\text{member}(O,C_2) \leftarrow \text{c}_{\text{link}}(C_1,C_2),$$

$$\text{member}(O,C_1).$$

which defines member as the transitive closure of relation $m_{\text{link}}$.
Disjointness axiom:
\[
\neg \text{member}(O, C2) \leftarrow \text{member}(O, C1),
\]
\[
c\text{link}(C1, C),
\]
\[
c\text{link}(C2, C),
\]
\[
C1 \neq C2
\] (7)

stating that the children of a class are disjoint;

Completeness of the hierarchy axioms:
\[
\neg \text{leaf}(C2) \leftarrow c\text{link}(C1, C2).
\]
\[
\text{leaf}(C) \leftarrow \neg \text{leaf}(C).
\]
\[
is\_\text{defined}(O) \leftarrow \text{leaf}(C),
\]
\[
\neg \text{is\_defined}(O).
\] (8)

stating that every object of the domain belongs to a leaf of the hierarchy; and

Axioms for observations in the initial situation:
\[
\text{holds}(F, 0) \leftarrow \text{obs}(F, true, 0).
\]
\[
\neg \text{holds}(F, 0) \leftarrow \text{obs}(F, false, 0).
\] (9)

This completes the definition of \(M^\mu\).

Now we define the transition diagram \(T\), given by an instance \(I\) of a travel logic form \(LF\). The signature of \(T\) consists of the objects of the universe of \(U\), inertial fluent \(in(O, A)\), statics \(object(O)\), area \(A\), root \(C\), class \(C\), m\_link \((O, C)\), c\_link \((C_1, C_2)\), and member \((O, C)\), as well as actions enter \((O, A)\) and leave \((O, A)\).

A state of \(T\) consists of the encoding \(e(I)\), the collection, stat, of static statements of \(LF\), and a complete and consistent collection of literals formed by relation \(in(O, A)\), which satisfies rule 3 of \(M^1\).

A triple \((\sigma_0, a, \sigma_1)\) is a transition of \(T\) iff there exists an answer set \(S\) of the program:
\[
M^1 \cup e(I) \cup \{\text{holds}(f, 0) : f \in \sigma_0\} \cup \{\neg\text{holds}(f, 0) : \neg f \in \sigma_0\} \cup \{\text{occurs}(e, 0) : e \in a\},
\]
such that

\[
\sigma_1 = \text{stat} \cup \{f : \text{holds}(f, 1) \in S\} \cup \{\neg f : \text{holds}(f, 1) \in S\}.
\]

Finally, we define the set \(P\) of all possible trajectories of \(T\), which are compatible with the recorded history \(H_n\) of \(LF\) (we refer to such trajectories as models of \(H_n\)). We say that a trajectory \(M = \langle \sigma_0, a_0, \sigma_1, \ldots, a_n, \sigma_n \rangle\) of \(T\) is a model of history \(H_n\) (with respect to \(T\), if:

- if \(\text{obs}(f, true, i) \in H_n\) then \(f \in \sigma_i\).
- if \(\text{obs}(f, false, i) \in H_n\) then \(\neg f \in \sigma_i\).
- \(hpd(e, i) \in H_n\) iff \(e \in a_i\).

A history is consistent, if it has a model. This completes our definition of a model \(\langle I, T, P \rangle\) of a travel logic form \(LF\).

We also say that a travel story is consistent, if its logic form is consistent, i.e., has a model. A statement \(Q\) is entailed by a travel story with travel logic form \(LF\), if \(Q\) is true in all models of \(LF\).

3 Complete Travel Stories

In this section we assume that the logic form, constructed from a text-question pair of a travel story is complete. This will be defined as follows:
Definition 2 [Complete TLF]
A travel logic form \( LF \) is called complete, if it has the following properties:

1. \( \text{card} = k \in LF \iff k = |\{o : \text{object}(o) \in LF\}|. \)
2. Every movable object \( O \) of \( LF \) is linked by \( m\text{-link} \) relation with the leaf of the story’s class hierarchy.
3. For every movable object \( O \) and area \( A \), such that \( O \) is initially located in \( A \), the logic form contains \( \text{obs}(in(O, A), \text{true}, 0) \).
4. The only fluents and actions mentioned in the logic form are basic.

A travel story is called complete, if it has a complete \( TLF \).

It is not difficult to show that:

Proposition 1 [Uniqueness of model for complete TLF]
A complete and consistent \( TLF \) has a unique model.

As usually in ASP, computing this model and answering our queries will be reduced to finding answer set of a logic program. To describe this process, we need the following definitions:

1. We say that a relation \( p \) is defined by a set of ground literals \( S \), if:
   
   • \( \bar{a} \in p \iff p((\bar{a})) \in S \)
   
   • \( \bar{a} \notin p \iff \neg p((\bar{a})) \in S \)

2. A trajectory \( \langle \sigma_0, a_0, \sigma_1, \ldots, a_{n-1}, \sigma_n \rangle \) of a dynamic system is defined by \( S \), if:
   
   • \( \sigma_i = \{f \mid \text{holds}(f, i) \in S\} \cup \{\neg f \mid \neg \text{holds}(f, i) \in S\} \), for any \( 1 \leq i \leq n \); and
   
   • \( a_i = \{a \mid \text{occurs}(a, i) \in S\} \), for any \( 1 \leq i < n \).

Now let us expand our program \( \mathcal{M}^n \) (where \( n \) is the length of story’s history), by the rules defining answers to our queries. Here are some examples:

\[
\begin{align*}
\text{answer}(O, \text{is} \_ \text{in}, A) & \leftarrow \text{query}(\text{where}(O), n), \\
& \quad \text{holds}(in(O, A), n).
\end{align*}
\]

\[
\begin{align*}
\text{answer}(O, \text{is} \_ \text{not} \_ \text{in}, A) & \leftarrow \text{query}(\text{where}(O), n), \\
& \quad \neg \text{holds}(in(O, A), n).
\end{align*}
\]

\[
\begin{align*}
\text{answer}(\text{there} \_ \text{are}, N, \text{members} \_ \text{of} \_ \text{class}, C, \text{in}, A) & \leftarrow \text{query}(\text{how} \_ \text{many}(C, A), n), \\
& \quad N = \#\text{count}\{\text{holds}(\text{in}(X, A), n) : \text{member}(X, C)\}.
\end{align*}
\]

(Here we use an extension of the original ASP by aggregates (such as \( \#\text{count} \)). (See, for instance, [16].)

Other queries can be added in a similar way. We denote the resulting program by \( P^n_1 \); by \( P^n_1(LF) \), we denote the union of \( P^n_1 \) and the travel logic form \( LF \) of a travel story.

Proposition 2 [Correctness of reasoning with complete stories]
The model of a complete and consistent travel story \( LF \) is defined by the unique answer set of program \( P^n_1(LF) \). Moreover, the answer literal, contained in this answer set, gives the correct answer to the story’s query.

This proposition can be used to answer queries from the first example of the introduction.
Example 5  [Example 1, revisited]
The logic form of the example story will contain description of movable objects $a$, $b$, $c$, and $d$, area room, together with statements:

\[
\begin{align*}
\text{card} &= 4 \\
\text{root(person).} \\
\text{c\_link(prof,person).} \\
\text{c\_link(student,person).} \\
\text{m\_link(a, student).} \\
\text{m\_link(b, student).} \\
\text{m\_link(c, prof).} \\
\text{m\_link(d, prof).}
\end{align*}
\]

describing the instance of the hierarchy; the initial situation:

\[
\text{obs(in(O, room), false, 0)}.
\]

and actions:

\[
\begin{align*}
\text{hpd(enter(d, room), 0).} \\
\text{hpd(enter(a, room), 0).} \\
\text{hpd(enter(b, room), 0).} \\
\text{hpd(enter(c, room), 1).}
\end{align*}
\]

Consider a text-question pair with the question: \textit{How many students are in the room?} The corresponding logic form will contain:

\[
\text{query(how\_many(student, room), 2).}
\]

Combination of this logic form and $\mathcal{M}^2$ constitutes program $\mathcal{P}_1^{\mathcal{L}F}$. As expected, the answer set of this program contains:

\[
\text{answer(there\_are, 2, members\_of\_class, student, in, room).}
\]

An answer set solver (in our case cling [7]), returns the answers instantaneously. The other questions from Example 1 can be answered in a similar way.

4 Uncertain Group Affiliation

In this section, we discuss answering questions about travel stories, similar to those considered in Example 2. The story in this example contains complete knowledge of the class hierarchy, the locations of objects in the initial state, and the actions up to the moment $n$. However, it has only \textit{partial knowledge about the class membership of objects}. In other words, the class $C$ in a statement $m\_link(O, C)$ is not necessarily a leaf and hence, the story’s hierarchy may have multiple instances. More precisely,

\textbf{Definition 3 [TLF with incomplete class membership]}

A logic form is incomplete with respect to class membership of the hierarchy if:

1. \text{card} = k \in LF \text{ iff } k = |\{o: object(o) \in LF\}|.

2. For every object $O$ from the logic form, there is a unique class $C$ of the hierarchy, such that $m\_link(O, C)$ belongs to the logic form.

3. For every movable object $O$ and area $A$, such that $O$ is initially located in $A$, the logic form contains $\text{obs(in}(O, A), true, 0)$.

4. The only fluents and actions mentioned in the logic form are basic.

This only differs from Definition 2 by clause (2) – objects of the domain do not have to be linked to the leaves of the hierarchy. To illustrate the definition, consider the logic form for Example 2. It includes
statements from the logic form for Example 1, together with the new classes *freshman*, *sophomore*, 
*junior*, and *senior*, included in the hierarchy by the statements:

\[
c_{\text{link}}(\text{freshman, student}).
\]
\[
c_{\text{link}}(\text{sophomore, student}).
\]
\[
c_{\text{link}}(\text{junior, student}).
\]
\[
c_{\text{link}}(\text{senior, student}).
\]

The corresponding question has the form:

\[
\text{query}(\text{how many}(\text{freshman, room}), 2).
\]

Since \( m_{\text{link}} \) does not connect students \( a \) and \( b \) with leaves of the hierarchy, the previous question answering algorithm, justified by Proposition 4, is not applicable. Indeed, the attempt to run the corresponding program, will return the answer set with 0 freshmen located in the room. The answer of course can be true if, say, both \( a \) and \( b \) were juniors, but it can also be false. The problem is that even though the history of Example 2 is the same as that of Example 1 (and hence is represented by the encoding described in Example 5), it does not have a unique model. Instead, it has 16 models, corresponding to 16 possible instances of the hierarchy compatible with our knowledge. All models have the same universe \( \{a, b, c, d\} \), but the objects can be placed into different classes. For instance, in the first instance, both \( a \) and \( b \) can be juniors, in the second, \( a \) can be a freshman and \( b \) a sophomore, etc.

Let us notice that models of travel logic form \( LF \), incomplete with respect to class membership of the hierarchy, coincide on atoms of the form \( \text{holds}(in(O, A), n) \), but differ on atoms of the form \( \text{member}(O, C) \).

So the definition of answers to queries containing “where” and “who” are not going to change. But the answer to a query containing “how many” should be redefined. This time the answer should be given by an interval, defining lower and upper bound for the cardinality of the corresponding set. More precisely,

**Definition 4** [Answer to a query how many for stories with multiple models] Let \( LF \) be a travel logic form with multiple models. We say that an interval \([l, u]\) is the \( LF \)’s answer to a query of the type \( \text{how many}(C, A) \) for a class \( C \) and area \( A \) if:

1. For every model of \( LF \) the number \( K \) of elements of the class \( C \) located in area \( A \) at step \( n \) in this model belongs to the interval \([l, u]\).
2. There is a model of \( LF \), such that the number of elements of the class \( C \), located in area \( A \) at step \( n \) in this model is \( l \).
3. There is a model of \( LF \), such that the number of elements of the class \( C \), located in area \( A \) at step \( n \) in this model is \( u \).

Our next step is to present a logic program \( P_n^2 \), which would be able to properly answer our questions.

**Definition of \( P_n^2 \):** The program will consists of:

Rules of \( M^n \).

Definition of the notion of *population* (The population of elements of class \( C \), located in \( A \), will be denoted by a term \( r(C, A) \)).

\[
\begin{align*}
\text{population}(r(C, A)).
\text{population}_\text{class}(r(C, A), C).
\text{population}_\text{area}(r(C, A), A).
\end{align*}
\]

(10)

Definition of new fluents – *lower* and *upper* bounds:

\[
\begin{align*}
\text{fluent}(\text{defined, lower bound}(P, N)).
\text{fluent}(\text{defined, upper bound}(P, N)).
\end{align*}
\]

(11)
Here \( N \) is the value of the lower and the upper bound, respectively. In rules 17 and 18, these fluents will be defined in terms of other relations – hence, the term defined. Such fluents are not subject to the inertia axiom. Instead, they require

The Closed World Assumption for defined fluents:

\[
\neg \text{holds}(F,T) \leftarrow \text{fluent}(\text{defined}, F), \\
\text{not holds}(F,T).
\] (12)

The uniqueness of value rule

\[
\neg \text{holds}(\text{lower bound}(P,N2),I) \leftarrow \text{holds}(\text{lower bound}(P,N1),I), \\
N1 \neq N2.
\]

\[
\neg \text{holds}(\text{upper bound}(P,N2),I) \leftarrow \text{holds}(\text{upper bound}(P,N1),I), \\
N1 \neq N2.
\] (13)

which states that the lower bound is a function, i.e., it has a unique value \( N \). Similarly for the upper bound.

Another collection of rules deals with the incompleteness of our stories.

Class membership uncertainty is captured by relation \( \text{maybe}(O,C) \), which holds if there is an instance of the hierarchy, compatible with the membership relation, defined by the logic form of our story,

\[
\text{maybe}(O,C) \leftarrow \text{member}(O,C).
\]

and relation \( \text{maybe member}(O,r(C,A)),I \), which holds if at step \( I \), object \( O \) might be a member of class \( C \) and is located in area \( A \):

\[
\text{maybe member}(O,r(C,A),0) \leftarrow \text{maybe}(O,C), \\
\text{holds}(\text{in}(O,A),0).
\] (15)

and

Relation subclass:

\[
\text{subclass}(C1,C2) \leftarrow \text{c link}(C1,C2).
\]

\[
\text{subclass}(C1,C2) \leftarrow \text{c link}(C1,C3), \\
\text{subclass}(C3,C2).
\] (16)

The next collection of rules defines the upper bound of the set of members of \( C \), initially located in \( A \).

\[
\text{holds}(\text{upper bound}(P,N),T) \leftarrow N = \#\text{count}\{\text{maybe member}(Y,P,T) : \text{object}(Y)\}.
\] (17)

The lower bound is given by the rule:

\[
\text{holds}(\text{lower bound}(P,N),T) \leftarrow \text{population class}(P,U), \\
\text{population area}(P,V), \\
N = \#\text{count}\{\text{holds}(\text{in}(Y,V),T) : \text{member}(Y,U)\}.
\] (18)

Finally, we need an auxiliary rule:

\[
\text{occurs}(X,T) \leftarrow \text{hpd}(X,T).
\] (19)
and the definition of answer for our new domain. This consists of the old definitions for queries not containing “how many” and the rule:

\[
\text{answer}(\text{between}, N1, \text{and}, N2, C, \text{in}, A) \leftarrow \text{query}(\text{how\_many}(C, A), n), \\
\text{holds}(\text{lower\_bound}(r(C, A), N1), n), \\
\text{holds}(\text{upper\_bound}(r(C, A), N2), n).
\]

(20)

This completes the construction of program \(P_{n2}^n\). As before, by \(P_{n2}^n(LF)\) we denote the union of \(P_{n2}^n\) and the logic form \(LF\).

**Proposition 3** [Correctness of reasoning with incomplete information about class membership]

If a consistent travel story with the logic form \(LF\) is incomplete with respect to class membership of the story’s hierarchy, then program \(P_{n2}^n(LF)\) has exactly one answer set, \(S\), and:

1. For every class \(c\) and area \(a\), interval \([l, u]\) is the answer to a question \(\text{how\_many}(c, a)\) iff \(\text{answer}(\text{between}, l, \text{and}, u, c, \text{in}, a) \in S\) iff \(\text{holds}(\text{lower\_bound}(r(c, a), l), n) \in S\) and \(\text{holds}(\text{upper\_bound}(r(c, a), u), n) \in S\);

2. For every object \(o\), area \(a\) is the answer to a query \(\text{where}(o, a)\) iff \(\text{answer}(o, \text{is\_in}, a) \in S\). Similarly for the other queries.

**Example 6** [Example 2, revisited]

It is not difficult to check that the answers to queries:

- \(\text{query}(\text{how\_many}(\text{person}, \text{room}), 2)\).
- \(\text{query}(\text{how\_many}(\text{professor}, \text{room}), 2)\).
- \(\text{query}(\text{how\_many}(\text{freshman}, \text{room}), 2)\).
- \(\text{query}(\text{how\_many}(\text{student}, \text{room}), 2)\).

are

- \(\text{answer}(\text{there\_are\_between}, 4, \text{and}, 4, \text{person, in\_the, room})\).
- \(\text{answer}(\text{there\_are\_between}, 2, \text{and}, 2, \text{professor, in\_the, room})\).
- \(\text{answer}(\text{there\_are\_between}, 0, \text{and}, 2, \text{freshman, in\_the, room})\).
- \(\text{answer}(\text{there\_are\_between}, 2, \text{and}, 2, \text{student, in\_the, room})\).

5 Uncertain Identity of Objects in the Initial Situation

In this section, we consider stories similar to that of Example 3. In addition to uncertainty about the class membership of John (he can be a freshman, a sophomore, etc.), this story does not have information about identity of other movers and the initial size of the domain’s populations. This means that the closed domain assumption is not any longer applicable, the corresponding hierarchy will have instances with multiple universes, and hence, multiple, universe dependent models. The initial situation of such stories can be represented using extended fluents \(\text{lower\_bound}\) and \(\text{upper\_bound}\). But, in contrast to the previous section \(\text{lower\_bound}\) and \(\text{upper\_bound}\) will not be defined in terms of fluent \(\text{in}\). Instead, they will be described in the story’s history and changed by actions. Therefore they will be defined as inertial. The TFL for such stories will be defined as follows:

**Definition 5** [TFL with incomplete identity of object in initial situation]

A travel logic form is incomplete with respect to identity of movers if:

1. If \(\text{card} = k \in LF\), then \(k\) is sufficiently big to be compatible with the story (i.e., it is bigger than the number of objects explicitly or implicitly mentioned in the story).

2. For every object \(O\) from the logic form, there is a unique class \(C\) of the hierarchy, such that \(m\_link(O, C)\) belongs to the logic form.
3. The initial situation is given by observations of fluents \textit{in}(O,A), \textit{lower}\_bound(P,N), and \textit{upper}\_bound(P,N).

4. The only actions mentioned in the logic form are basic.

5. There are no concurrent actions.

(This last requirement is inessential. It is added to simplify the presentation).

To illustrate this case let us consider again Example 3.

\textbf{Example 7 [Example 3, revisited]}

The logic form for this example consists of statements:

\begin{verbatim}
card = 2.
object(john).
m_link(john,student).
obs(lower_bound(r(person, room), 0), true, 0).
obs(upper_bound(r(person, room), 1), true, 0).
hpd(enter(john, room), 0).
\end{verbatim}

\textit{together with representation of the area and the hierarchy (as in Example 2). (Note that the cardinality can be any number larger than 2. This will not change answers to our queries in which this number will be replaced by symbolic constant max\_size.)}

The answers to queries for travel stories incomplete with respect to identity of movers will be computed using a logic program \(P^n_3(LF)\) defined as follows:

\textbf{Definition of} \(P^n_3\): The program consists of rules of \(P^n_3\) with rules 11 and 12 replaced by

\begin{equation}
\begin{aligned}
\text{fluent}(\text{inertial, lower}\_bound(P,N)). \\
\text{fluent}(\text{inertial, upper}\_bound(P,N)).
\end{aligned}
\end{equation}

\textit{together with the following rules.}

\begin{equation}
\begin{aligned}
\text{occurs(leave(O, A_2), I)} & \leftarrow \text{occurs(enter(O, A_1), I)}, \\
& \text{holds(in(O, A_2), I)}. \\
\text{holds(upper_bound(r(C, A), N_2), I + 1)} & \leftarrow \text{occurs(enter(O, A), I)}, \\
& \text{maybe(O, C)}, \\
& \text{holds(upper_bound(r(C, A), N_1), I)}, \\
& N_2 = N_1 + 1.
\end{aligned}
\end{equation}

\begin{equation}
\begin{aligned}
\text{holds(lower_bound(r(C, A), N_2), I + 1)} & \leftarrow \text{occurs(enter(O, A), I)}, \\
& \text{member(O, C)}, \\
& \text{holds(lower_bound(r(C, A), N_1), I)}, \\
& N_2 = N_1 + 1.
\end{aligned}
\end{equation}

\begin{equation}
\begin{aligned}
\text{holds(upper_bound(r(C, A), N_2), I + 1)} & \leftarrow \text{occurs(leave(O, A), I)}, \\
& \text{member(O, C)}, \\
& \text{holds(upper_bound(r(C, A), N_1), I)}, \\
& N_1 > 0, \\
& N_2 = N_1 - 1.
\end{aligned}
\end{equation}
holds(lower_bound(r(C, A), N2), I + 1) ← occurs(leave(O, A), I), maybe(O, C),
holds(lower_bound(r(C, A), N1), I), N1 > 0, N2 = N1 - 1.  

(26)

In addition, the definition 20 of answer will be replaced by a new definition which accommodates the unknown cardinality of the domain.

\[
\begin{align*}
\text{answer(between, } & N1, \text{ and, } N2, C, \text{ in, } A) \leftarrow \\
& \text{query(how_many(C, A), n), holds(lower_bound(r(C, A), N1), n), holds(upper_bound(r(C, A), N2), n), N2 = k} \\
\text{answer(between, } & N1, \text{ and, max_size, C, in, } A) \leftarrow \\
& \text{query(how_many(C, A), n), holds(lower_bound(r(C, A), N1), n), holds(upper_bound(r(C, A), N2), n), N2 = k}
\end{align*}
\]

where \( k \) is the cardinality of the domain.

This completes the definition of \( P^n_3 \).

**Proposition 4** [Correctness of reasoning with incomplete identity of movers]

If a consistent travel story with the logic form \( LF \) is incomplete with respect to class membership of the story’s hierarchy, then program \( P^n_3(LF) \) has exactly one answer set, \( S \), and:

1. For every class \( c \) and area \( a \), interval \([l, u]\) is the answer to a question \( \text{how_many}(c, a) \) iff \( \text{answer(between, } l, \text{ and, } u, c, \text{ in, } a) \in S \) iff 
   \( \text{holds(lower_bound}(r(c, a), l), n) \in S \) and \( \text{holds(upper_bound}(r(c, a), u), n) \in S \).

2. For every object \( o \), area \( a \) is the answer to a query \( \text{where}(o, a) \) iff \( \text{answer(o, is_in, } a) \in S \). Similarly for the other queries.

It is not difficult to verify that the answers to queries:

query(how_many(person, room), 1).
query(how_many(professor, room), 1).
query(how_many(freshman, room), 1).
query(how_many(student, room), 1).

are

answer(there_are_between, 1, and, max_size, person, in_the, room).
answer(there_are_between, 0, and, 1, professor, in_the, room).
answer(there_are_between, 0, and, max_size, freshman, in_the, room).
answer(there_are_between, 1, and, max_size, student, in_the, room).

6 Uncertainty in the Number of Movers

Finally, we consider stories similar to those given in Example 4. Such stories have neither information about identities of movers in the domain nor their precise number. All we know is the limits for the sizes of the corresponding populations in the initial situation and the number of people from such populations involved in the moves. In other words we are interested in stories with the following logic form:
Definition 6 [TLF with uncertainty in the number of movers.]
A travel logic form is incomplete with respect to number of movers if:

1. If $\text{card} = k \in LF$ then $k$ is sufficiently big to be compatible with the story.
2. The logic form contains no basic actions and fluents.
3. There are no concurrent actions.

We illustrate this by example Example 4.

Example 8 [Example 4 revisited]
The logic form will have representation of the areas and the hierarchy (as in Example 2) together with statements:

\[
\begin{align*}
\text{card} &= 10 \\
\text{obs}(\text{lower bound}(r(\text{student}, \text{room}), 4), \text{true}, 0) \\
\text{obs}(\text{upper bound}(r(\text{student}, \text{room}), 4), \text{true}, 0) \\
\text{obs}(\text{lower bound}(r(\text{professor}, \text{room}), 2), \text{true}, 0) \\
\text{obs}(\text{upper bound}(r(\text{professor}, \text{room}), 2), \text{true}, 0) \\
\text{hpd}(\text{enter}(0, 2, \text{student}, \text{room}), 0)
\end{align*}
\]

(Recall that $\text{enter}(0, 2, \text{student}, \text{room})$ reads as at least two students entered the room.) Note also that, as in the previous example, the value of $\text{card}$ is arbitrary. It can be any number greater than or equal to 8.

This is the first time when we consider a logic form of travel story containing extended actions. This seems like a substantial change which may require us to change our previous framework. We do not even have a notion of model defined for such logic forms — so far we only considered transition diagrams describing effects of actions $\text{enter}(O, A)$ and $\text{leave}(O, A)$. To deal with the problem we translate a logic form $LF$ with uncertainty in the number of movers into a collection of complete logic forms, called instances of $LF$ whose models will be viewed as models of $LF$. First we need the following definition.

Definition 7 [Instance of a logic form with extended history]
Let $LF$ be logic form with extended history. Let $\text{inst}(LF)$ be a logic form obtained from $LF$ by

1. Expanding the signature of $LF$ by new constant symbols $x_1, x_2, \ldots, x_m$ where $m$ is the cardinality of the domain.
2. Adding statements $\text{object}(x_1), \ldots, \text{object}(x_m)$ and a collection of statements of the form $\text{m_link}(x_i, c)$ such that for every $0 \leq i \leq m$ object $x_i$ is allocated into some leaf $c$ of the domain hierarchy,
3. Adding a collection of statement of the form $\text{obs}(\text{in}(x_i, \text{area}), \text{true}, 0)$ placing each $x_i$ into some area in the initial situation.
4. Replacing every occurrence of a statement $\text{hpd}(a, i)$ where $a$ is and extended action $\text{enter}(\text{rel}, k, c, \text{area})$ by a collection of basic actions
   \[
   \text{occurs}(\text{enter}(x_1, \text{area}), i), \ldots, \text{occurs}(\text{enter}(x_n, \text{area}), i)
   \]
   where $x_1, \ldots, x_n$ are elements of class $c$ and $n$ and $k$ satisfy the relation $\text{rel}$.
5. Similarly for $\text{leave}$.
6. Removing all statements containing occurrences of $\text{lower bound}$ and $\text{upper bound}$.

It is easy to see that $\text{inst}(LF)$ is a complete logic form. We say that $\text{inst}(LF)$ is an instance of $LF$ if it is consistent and satisfies all the observations of $LF$ made in the initial situation.

By models of $LF$ we mean models of its instances.
The addition is sufficient. The definition of answer from 4 becomes applicable and we only need to worry about computing these answers. This will be done with the help of logic program $P^a_4(LF)$ defined as follows:

**Definition of $P^a_4$:**
The program consists of rules of $P^a_3$ together with the following rules.

Definition of extended actions:

\[
\text{action}(enter(0,N,C,A)). \\
\text{action}(enter(1,N,C,A)). \\
\text{action}(enter(2,N,C,A)).
\] (28)

Similarly for leave.

Direct effect of $occurs(enter(0,N1,C1,A))$ (At least $N1$ movers of class $C$ entered $A$):

\[
\text{holds}(\text{lower\_bound}(r(C2,A),N),I+1) \leftarrow occurs(enter(0,N1,C1,A),I), \\
\text{subclass}(C1,C2), \\
\text{holds}(\text{lower\_bound}(r(C2,A),N2),I), \\
N = N1 + N2.
\] (29)

\[
\text{holds}(\text{upper\_bound}(r(C2,A),k),I+1) \leftarrow occurs(enter(0,N1,C1,A),I), \\
\text{subclass}(C1,C2).
\] (30)

where $k$ is the cardinality of the domain.

Direct effect of $occurs(enter(1,N1,C1,A))$ ($N1$ movers of class $C$ entered $A$):

\[
\text{holds}(\text{lower\_bound}(r(C2,A),N),I+1) \leftarrow occurs(enter(0,N1,C1,A),I), \\
\text{holds}(\text{lower\_bound}(r(C2,A),N2),I), \\
N = N1 + N2.
\] (31)

\[
\text{holds}(\text{upper\_bound}(r(C2,A),N),I+1) \leftarrow occurs(enter(0,N1,C1,A),I), \\
\text{holds}(\text{upper\_bound}(r(C2,A),N2),I), \\
N = N1 + N2.
\] (32)

Similarly for other actions.

The rules of $P^a_3$ about basic fluents and actions can be dropped from the program.

This completes the definition of $P^a_4$.

**Proposition 5** [Correctness of reasoning with unknown number of movers]
If a consistent travel story with the logic form $LF$ is incomplete with respect to number of movers then program $P^a_4(LF)$ has exactly one answer set, $S$, and for every class $c$ and area $a$, interval $[l,u]$ is the answer to a question $\text{how\_many}(c,a)$ iff $\text{answer}(\text{between}, l, \text{and}, u, c, \text{in}, a) \in S$ iff $\text{holds}(\text{lower\_bound}(r(c,a),l),n) \in S$ and $\text{holds}(\text{upper\_bound}(r(c,a),u),n) \in S$.

**Example 9** [Example 4 revisited]
It is not difficult to verify that the answers to queries:

query($\text{how\_many}(\text{person}, \text{room}), 1$).
query($\text{how\_many}(\text{professor}, \text{room}), 1$).
query($\text{how\_many}(\text{freshman}, \text{room}), 1$).
query($\text{how\_many}(\text{student}, \text{room}), 1$).

are

answer($\text{there\_are\_between}, 8, \text{and}, \text{max\_size}, \text{person, in\_the, room}$).
answer($\text{there\_are\_between}, 2, \text{and}, 2, \text{professor, in\_the, room}$).
answer($\text{there\_are\_between}, 0, \text{and}, \text{max\_size}, \text{freshman, in\_the, room}$).
answer($\text{there\_are\_between}, 6, \text{and}, \text{max\_size}, \text{student, in\_the, room}$).
7 Discussion

The axiomatization of travel domains presented in this paper is based on extensive research on the relationship between reasoning about actions and Answer Set Prolog. This work, which started in [10, 20], led to the development of rich theory and multiple applications [6]. Reasoning about various types of motion, including travel, was often used to test this theory. In the state of the art approach the background knowledge about action is normally represented in action languages [11]. In fact, there is a small collection of modules containing axioms for various actions built in modular action languages (see, for instance, [13, 14, 1]) These theories are automatically translated into ASP which allows solutions of various problems using efficient answer set solvers like smodels [18], dlv [5], clingo [8] and others. In our extended work we also use a modular action language ALM [12] but this was omitted to save the space. We believe that results presented in this paper will, eventually, be useful for the design of real question-answering systems. Of course to make this happen we need to develop translations from natural language which will be able to classify a text into a travel story and translate it into the ASP based logic form. First steps towards this were made in [19] in which this was done for a simple controlled language. However, much more research is needed to properly combine natural language processing and reasoning parts of such future systems. Still we believe that our work helps to better understand different types of incompleteness and their impact on question answering which contributes to this effort.

References


5http://downloads.sourceforge.net/potassco/guide.pdf


