Toward Question Answering in Travel Domains: locations of participants and cardinalites 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 classes[1] 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 system[2]. 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

[1] Here, 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.

[2] We 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 $\langle T, Q\rangle$ is translated into a suitable logical representation, called *logic form* of $\langle T, Q\rangle$. 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:

   $$root(C) \qquad class(C) \qquad c\_link(C_1, C_2)$$

   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:

   $$object(O) \qquad m\_link(O, C)$$

   The last statement places an object O into a class C of the hierarchy;

   (c) Cardinality of the universe:

   $$card = k$$

2. Collection of areas:

   $$area(A)$$

   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 statements of the form:

   $$hpd(a, i) \qquad obs(f, v, i)$$

   where v is a boolean value; the first statement means that action a happened at step i of the system's trajectory; the second one states that at step i fluent[3] 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

   $$enter(O, A)$$

   read as *object O enters area A*, and

   $$enter(X, N, C, A)$$

   where X is min, max, or $equal$, meaning that *minimum, maximum, or exactly N members of class C entered area A*[4]. Similarly for actions *leave*. The fluent

   $$in(O, A)$$

   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:

   $$lower\_bound(P, N)$$

   $$upper\_bound(P, N)$$

   read as *N is the lower (upper) bound of the size of population P*. Action $enter(O, A)$ and fluent $in(O, A)$ will be called *basic*. Other actions and fluents will be called *extended*.

4. Collection of queries of the form:

   $$query(in(O, A)) \quad -- \quad \textit{Is object } O \textit{ in the area } A?$$

   $$query(where(O)) \quad -- \quad \textit{Where is } O?$$

   $$query(who(A)) \quad -- \quad \textit{Who is located in area } A?$$

   $$query(how\_many(C, A)) \quad -- \quad \textit{How many members of class } C \textit{ are in } A?$$

   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 $\mathcal{H}(LF)$ we denote the hierarchy described by statements (1a) of LF. An instance, $\mathcal{I}(LF)$, of this hierarchy consists of the universe U of cardinality k containing all the objects defined in LF, and a mapping from classes of $\mathcal{H}(LF)$ to subsets of U. The mapping is such that, for every statement $c\_link(c_1, c_2)$ of LF, class c_1 is mapped into a subset of c_2 and the root is mapped into U. By *encoding* of $\mathcal{I}$ with the universe U, we mean the set

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

A model of LF consists of an instance $\mathcal{I}$ of the LF's hierarchy, a *transition diagram* T (describing effects of actions $enter(O, A)$ and $leave(O, A)$ on locations of movers from $\mathcal{I}$), and a collection P of paths of T, compatible with the TLF's history. The T and P will be defined by an instance $\mathcal{I}$ and an ASP program, $\mathcal{M}^n$, 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 $\mathcal{M}^n$: The program consists of the following rules.

Fluents and *steps*:

$$\begin{array}{l} step(0...n). \\ fluent(inertial, in(O, A)). \end{array} \tag{1}$$

(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:

$$holds(in(O, A), I + 1) \leftarrow occurs(enter(O, A), I). \tag{2}$$

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

State constraint:

$$\begin{array}{rcl} \neg holds(in(O, A_2), I) & \leftarrow & holds(in(O, A_1), I), \\ & & A_1 \neq A_2. \end{array} \tag{3}$$

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

Executability condition:

$$\begin{array}{rcl} \leftarrow & & occurs(enter(O, A), I), \\ & & holds(in(O, A), I). \end{array} \tag{4}$$

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

Inertia axioms:

$$\begin{array}{rcl} holds(F, I + 1) \leftarrow & & fluent(inertial, F), \\ & & holds(F, I), \\ & & \text{not } \neg holds(F, I + 1) \\ \neg holds(F, I + 1) \leftarrow & & fluent(inertial, F), \\ & & \neg holds(F, I), \\ & & \text{not } holds(F, I + 1) \end{array} \tag{5}$$

which say that inertial fluents tend to stay unchanged;

Membership of the class hierarchy:

$$\begin{array}{rcl} member(O, C) & \leftarrow & m\_link(O, C). \\ member(O, C2) & \leftarrow & c\_link(C1, C2), \\ & & member(O, C1). \end{array} \tag{6}$$

which defines *member* as the *transitive closure of relation* m_link;

Disjointness axiom:

$$\begin{array}{lcl} \neg member(O, C2) & \leftarrow & member(O, C1), \\ & & c\_link(C1, C), \\ & & c\_link(C2, C), \\ & & C1 \neq C2 \end{array} \tag{7}$$

stating that the children of a class are disjoint;

Completeness of the hierarchy axioms:

$$\begin{array}{lcl} \neg leaf(C2) & \leftarrow & c\_link(C1, C2). \\ leaf(C) & \leftarrow & \text{not } leaf(C). \\ is\_defined(O) & \leftarrow & leaf(C), \\ & & member(O, C). \\ & \leftarrow & \text{not } is\_defined(O). \end{array} \tag{8}$$

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

Axioms for observations in the initial situation:

$$\begin{array}{lcl} holds(F, 0) & \leftarrow & obs(F, true, 0). \\ \neg holds(F, 0) & \leftarrow & obs(F, false, 0). \end{array} \tag{9}$$

This completes the definition of $\mathcal{M}^n$.

Now we define the transition diagram T, given by an instance $\mathcal{I}$ 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 $e(\mathcal{I})$, 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 $\mathcal{M}^1$.

A triple $\langle \sigma_0, a, \sigma_1 \rangle$ is a *transition* of T iff there exists an answer set S of the program:

$$\mathcal{M}^1 \cup e(\mathcal{I}) \cup \{holds(f, 0) : f \in \sigma_0\} \cup \{\neg holds(f, 0) : \neg f \in \sigma_0\} \cup \{occurs(e, 0) : e \in a\},$$

such that

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

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 $M = \langle \sigma_0, a_0, \sigma_1, \ldots, a_{n-1}, \sigma_n \rangle$ of T is a *model* of history H_n (with respect to T), if:

- if $obs(f, true, i) \in H_n$ then $f \in \sigma_i$.
- if $obs(f, false, i) \in H_n$ 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 $\langle \mathcal{I}, T, P \rangle$ 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. $card = k \in LF$ iff $k = |\{o : object(o) \in LF\}|$.
2. Every movable object O of LF is linked by m_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 $obs(in(O, A), true, 0)$.
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:
   - $\sigma_i = \{f \mid holds(f, i) \in S\} \cup \{\neg f \mid \neg holds(f, i) \in S\}$, for any $1 \leq i \leq n$; and
   - $a_i = \{a \mid occurs(a, i) \in S\}$, 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:

$$\begin{array}{rcl} answer(O, is\_in, A) & \leftarrow & query(where(O), n), \\ & & holds(in(O, A), n). \\ answer(O, is\_not\_in, A) & \leftarrow & query(where(O), n), \\ & & \neg holds(in(O, A), n). \end{array}$$

$$\begin{array}{rcl} answer(there\_are, N, members\_of\_class, C, in, A) & \leftarrow & \\ & & query(how\_many(C, A), n), \\ & & N = \#count\{holds(in(X, A), n) : member(X, C)\}. \end{array}$$

(Here we use an extension of the original ASP by aggregates (such as $\#count$). (See, for instance, [16].) Other queries can be added in a similar way. We **denote the resulting program by $\mathcal{P}_1^n$; by $\mathcal{P}_1^n(LF)$, we denote the union of $\mathcal{P}_1^n$ 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 $\mathcal{P}_1^n(LF)$. 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:

$card = 4$
$root(person)$.
$c\_link(prof, person)$.
$c\_link(student, person)$.
$m\_link(a, student)$.
$m\_link(b, student)$.
$m\_link(c, prof)$.
$m\_link(d, prof)$.

describing the instance of the hierarchy; the initial situation:

$obs(in(O, room), false, 0)$.

and actions:

$hpd(enter(d, room), 0)$.
$hpd(enter(a, room), 0)$.
$hpd(enter(b, room), 0)$.
$hpd(enter(c, room), 1)$.

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

$query(how\_many(student, room), 2)$.

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

$answer(there\_are, 2, members\_of\_class, student, in, room)$.

An answer set solver (in our case *clingo* [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 *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,

Definition 3 *[TLF with incomplete class membership]*
A logic form is *incomplete with respect to class membership of the hierarchy* if:

1. $card = k \in LF$ 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 $obs(in(O, A), true, 0)$.
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:

$c\_link(freshman, student)$.
$c\_link(sophomore, student)$.
$c\_link(junior, student)$.
$c\_link(senior, student)$.

The corresponding question has the form:

$query(how\_many(freshman, room), 2)$.

Since m_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 $holds(in(O, A), n)$, but differ on atoms of the form $member(O, C)$. 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 $how\_many(C, A)$ 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 $\mathcal{P}_2^n$, which would be able to properly answer our questions.

Definition of $\mathcal{P}_2^n$: The program will consists of:

Rules of $\mathcal{M}^n$.

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

$$\begin{array}{l} population(r(C, A)). \\ population\_class(r(C, A), C). \\ population\_area(r(C, A), A). \end{array} \tag{10}$$

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

$$\begin{array}{l} fluent(defined, lower\_bound(P, N)). \\ fluent(defined, upper\_bound(P, N)). \end{array} \tag{11}$$

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*:

$$\begin{array}{lcl} \neg holds(F,T) & \leftarrow & fluent(defined,F), \\ & & \text{not } holds(F,T). \end{array} \tag{12}$$

The *uniqueness of value* rule

$$\begin{array}{lcl} \neg holds(lower\_bound(P,N2),I) & \leftarrow & holds(lower\_bound(P,N1),I), \\ & & N1 \; != \; N2. \\ \neg holds(upper\_bound(P,N2),I) & \leftarrow & holds(upper\_bound(P,N1),I), \\ & & N1 \; != \; N2. \end{array} \tag{13}$$

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 $maybe(O,C)$, which holds if there is an instance of the hierarchy, compatible with the membership relation, defined by the logic form of our story,

$$\begin{array}{lcl} maybe(O,C) & \leftarrow & member(O,C). \\ maybe(O,C_1) & \leftarrow & subclass(C_1,C_2), \\ & & m\_link(O,C_2). \end{array} \tag{14}$$

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

$$\begin{array}{lcl} maybe\_member(O,r(C,A),0) & \leftarrow & maybe(O,C), \\ & & holds(in(O,A),0). \end{array} \tag{15}$$

and

Relation *subclass*:

$$\begin{array}{lcl} subclass(C1,C2) & \leftarrow & c\_link(C1,C2). \\ subclass(C1,C2) & \leftarrow & c\_link(C1,C3), \\ & & subclass(C3,C2). \\ subclass(C,C). & & \end{array} \tag{16}$$

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

$$holds(upper\_bound(P,N),T) \quad \leftarrow \quad N = \#count\{maybe\_member(Y,P,T) : object(Y)\}. \tag{17}$$

The lower bound is given by the rule:

$$\begin{array}{lcl} holds(lower\_bound(P,N),T) & \leftarrow & population\_class(P,U), \\ & & population\_area(P,V), \\ & & N = \#count\{holds(in(Y,V),T) : member(Y,U)\}. \end{array} \tag{18}$$

Finally, we need an auxiliary rule:

$$occurs(X,T) \quad \leftarrow \quad hpd(X,T). \tag{19}$$

and *the definition of answer* for our new domain. This consists of the old definitions for queries not containing "how many" and the rule:

$$\begin{array}{lll} answer(between, N1, and, N2, C, in, A) & \leftarrow & \\ & & query(how\_many(C, A), n), \\ & & holds(lower\_bound(r(C, A), N1), n), \\ & & holds(upper\_bound(r(C, A), N2), n). \end{array} \quad (20)$$

This completes the construction of program $\mathcal{P}_2^n$. As before, by $\mathcal{P}_2^n(LF)$ we denote the union of $\mathcal{P}_2^n$ 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 $\mathcal{P}_2^n(LF)$ has exactly one answer set, S, and:

1. For every class c and area a, interval $[l, u]$ is the answer to a question $how\_many(c, a)$ iff $answer(between, l, and, u, c, in, a) \in S$ iff
   $holds(lower\_bound(r(c, a), l), n) \in S$ and $holds(upper\_bound(r(c, a), u), n) \in S$;
2. For every object o, area a is the answer to a query $where(o, a)$ iff $answer(o, is\_in, a) \in S$. Similarly for the other queries.

Example 6 *[Example 2, revisited]*
It is not difficult to check that the answers to queries:

$query(how\_many(person, room), 2).$
$query(how\_many(professor, room), 2).$
$query(how\_many(freshman, room), 2).$
$query(how\_many(student, room), 2).$

are

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

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 *lower_bound* and *upper_bound.* But, in contrast to the previous section *lower_bound* and *upper_bound* will not be defined in terms of fluent *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 *[TLF with incomplete identity of object in initial situation]*
A travel logic form is *incomplete with respect to identity of movers* if:

1. If $card = k \in LF$, 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 $in(O, A)$, $lower\_bound(P, N)$, and $upper\_bound(P, N)$.

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.

Example 7 *[Example 3, revisited]*
The logic form for this example consists of statements:

$card = 2.$
$object(john).$
$m\_link(john, student).$
$obs(lower\_bound(r(person, room), 0), true, 0).$
$obs(upper\_bound(r(person, room), 1), true, 0).$
$hpd(enter(john, room), 0).$

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 $\mathcal{P}_3^n(LF)$ defined as follows:

Definition of $\mathcal{P}_3^n$: The program consists of rules of $\mathcal{P}_3^n$ with rules 11 and 12 replaced by

$$\begin{array}{l} fluent(inertial, lower\_bound(P, N)). \\ fluent(inertial, upper\_bound(P, N)). \end{array} \tag{21}$$

together with the following rules.

$$\begin{array}{lcl} occurs(leave(O, A_2), I) & \leftarrow & occurs(enter(O, A_1), I), \\ & & holds(in(O, A_2), I). \end{array} \tag{22}$$

$$\begin{array}{lcl} holds(upper\_bound(r(C, A), N2), I+1) & \leftarrow & occurs(enter(O, A), I), \\ & & maybe(O, C), \\ & & holds(upper\_bound(r(C, A), N1), I), \\ & & N2 = N1 + 1. \end{array} \tag{23}$$

$$\begin{array}{lcl} holds(lower\_bound(r(C, A), N2), I+1) & \leftarrow & occurs(enter(O, A), I), \\ & & member(O, C), \\ & & holds(lower\_bound(r(C, A), N1), I), \\ & & N2 = N1 + 1. \end{array} \tag{24}$$

$$\begin{array}{lcl} holds(upper\_bound(r(C, A), N2), I+1) & \leftarrow & occurs(leave(O, A), I), \\ & & member(O, C), \\ & & holds(upper\_bound(r(C, A), N1), I), \\ & & N1 > 0, \\ & & N2 = N1 - 1. \end{array} \tag{25}$$

$$
\begin{array}{lll}
holds(lower\_bound(r(C,A),N2),I+1) & \leftarrow & occurs(leave(O,A),I), \\
 & & maybe(O,C), \\
 & & holds(lower\_bound(r(C,A),N1),I), \\
 & & N1>0, \\
 & & N2=N1-1.
\end{array} \tag{26}
$$

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

$$
\begin{array}{lll}
answer(between,N1,and,N2,C,in,A) & \leftarrow & \\
 & & query(how\_many(C,A),n), \\
 & & holds(lower\_bound(r(C,A),N1),n), \\
 & & holds(upper\_bound(r(C,A),N2),n), \\
 & & N2 \neq k \\
answer(between,N1,and,max\_size,C,in,A) & \leftarrow & \\
 & & query(how\_many(C,A),n), \\
 & & holds(lower\_bound(r(C,A),N1),n), \\
 & & holds(upper\_bound(r(C,A),N2),n), \\
 & & N2=k
\end{array} \tag{27}
$$

where k is the cardinality of the domain.
This completes the definition of $\mathcal{P}_3^n$.

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 $\mathcal{P}_3^n(LF)$ has exactly one answer set, S, and:

1. For every class c and area a, interval $[l,u]$ is the answer to a question $how\_many(c,a)$ iff $answer(between,l,and,u,c,in,a) \in S$ iff
   $holds(lower\_bound(r(c,a),l),n) \in S$ and $holds(upper\_bound(r(c,a),u),n) \in S$;
2. For every object o, area a is the answer to a query $where(o,a)$ iff $answer(o,is\_in,a) \in S$. 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 $card = k \in LF$ 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:

$card = 10$
$obs(lower\_bound(r(student, room), 4), true, 0)$
$obs(upper\_bound(r(student, room), 4), true, 0)$
$obs(lower\_bound(r(professor, room), 2), true, 0)$
$obs(upper\_bound(r(professor, room), 2), true, 0)$
$hpd(enter(0, 2, student, room), 0)$

(Recall that $enter(0, 2, student, room)$ reads as *at least two students entered the room.*) Note also that, as in the previous example, the value of *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 $enter(O, A)$ and $leave(O, A)$. 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 $inst(LF)$ be a logic form obtained from LF by

1. Expending 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 $object(x_1), \ldots, object(x_m)$ and a collection of statements of the form $m\_link(x_i, c)$ 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 $obs(in(x_i, area), true, 0)$ placing each x_i into some area in the initial situation.
4. Replacing every occurrence of a statement $hpd(a, i)$ where a is and extended action $enter(rel, k, c, area)$ by a collection of basic actions
   $\{occurs(enter(x_1, area), i), \ldots occurs(enter(x_n, area), i)\}$
   where $x_1, \ldots, x_n$ are elements of class c and n and k satisfy the relation rel.
5. Similarly for *leave*.
6. Removing all statements containing occurrences of *lower_bound* and *upper_bound*.

It is easy to see that $inst(LF)$ is a complete logic form. We say that $inst(LF)$ 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 $\mathcal{P}_4^n(LF)$ defined as follows:

Definition of $\mathcal{P}_4^n$:
The program consists of rules of $\mathcal{P}_3^n$ together with the following rules.

Definition of extended actions:

$$\begin{array}{l} action(enter(0,N,C,A)).\\ action(enter(1,N,C,A)).\\ action(enter(2,N,C,A)). \end{array} \tag{28}$$

Similarly for *leave*.

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

$$\begin{array}{lcl} holds(lower\_bound(r(C2,A),N),I+1) & \leftarrow & occurs(enter(0,N1,C1,A),I),\\ & & subclass(C1,C2),\\ & & holds(lower\_bound(r(C2,A),N2),I),\\ & & N = N1+N2. \end{array} \tag{29}$$

$$\begin{array}{lcl} holds(upper\_bound(r(C2,A),k),I+1) & \leftarrow & occurs(enter(0,N1,C1,A),I),\\ & & subclass(C1,C2). \end{array} \tag{30}$$

where k is the cardinality of the domain.

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

$$\begin{array}{lcl} holds(lower\_bound(r(C2,A),N),I+1) & \leftarrow & occurs(enter(0,N1,C1,A),I),\\ & & holds(lower\_bound(r(C2,A),N2),I),\\ & & N = N1+N2. \end{array} \tag{31}$$

$$\begin{array}{lcl} holds(upper\_bound(r(C2,A),N),I+1) & \leftarrow & occurs(enter(0,N1,C1,A),I),\\ & & holds(upper\_bound(r(C2,A),N2),I),\\ & & N = N1+N2. \end{array} \tag{32}$$

Similarly for other actions.
The rules of $\mathcal{P}_3^n$ about basic fluents and actions can be dropped from the program.

This completes the definition of $\mathcal{P}_4^n$.

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 $\mathcal{P}_4^n(LF)$ has exactly one answer set, S, and for every class c and area a, interval $[l,u]$ is the answer to a question $how\_many(c,a)$ iff $answer(between,l,and,u,c,in,a) \in S$ iff $holds(lower\_bound(r(c,a),l),n) \in S$ and $holds(upper\_bound(r(c,a),u),n) \in S$;

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

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

are

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

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 $\mathcal{ALM}$ [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

[1] Varol Akman, Selim T. Erdogan, Joohyung Lee, Vladimir Lifschitz, and Hudson Turner. Representing the zoo world and the traffic world in the language of the causal calculator. *Artif. Intell.*, 153(1-2):105–140, 2004.

[2] M. Balduccini, C. Baral, and Y. Lierler. *Knowledge Representation and Question Answering, chapter 1.* Handbook of Knowledge Representation, Elsevier, 2007.

[3] C. Baral, M. Gelfond, G. Gelfond, and R. Scherl. Textual Inference by Combining Multiple Logic Programming Paradigms. In *AAAI'05 Workshop on Inference for Textual Question Answering*, 2005.

[4] C. Baral, M. Gelfond, and R. Scherl. Using answer set programming to answer complex queries. In *Workshop on Pragmatics of Question Answering at HLT-NAAC2004*, 2004.

[5] Francesco Calimeri, Tina Dell'Armi, Thomas Eiter, Wolfgang Faber, Georg Gottlob, Giovanbattista Ianni, Giuseppe Ielpa, Christoph Koch, Nicola Leone, Simona Perri, Gerard Pfeifer, and Axel Polleres. The DLV System. In Sergio Flesca and Giovanbattista Ianni, editors, *Proceedings of the 8th European Conference on Artificial Intelligence (JELIA 2002)*, Sep 2002.

[6] C.Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving.* Cambridge University Press, Chapter 5, 2003.

[7] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele. A User's Guide to gringo, clasp, clingo, and iclingo. Unpublished draft, 2008. Available at URL[5].

[8] M. Gebser, B. Kaufman, A. Neumann, and T. Schaub. Conflict-deriven answer set enumeration. In C. Baral, G. Brewka, and J. Schlipf, editors, *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, volume 3662 of *lnai*, pages 136–148. Springer, 2007.

[9] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[10] M. Gelfond and V. Lifschitz. Representing Actions in Extended Logic Programs. In *Joint International Conference and Symposium on Logic Programming*, pages 559–573. MIT Press, 1992.

[5] `http://downloads.sourceforge.net/potassco/guide.pdf`

[11] M. Gelfond and V. Lifschitz. Action Languages. *Electronic Transactions on AI*, 3, 1998.

[12] Michael Gelfond and Daniela Inclezan. Yet another modular action language. In *In Proceedings of the Second International Workshop on Software Engineering for Answer Set Programming10*, pages 64–78, 2009.

[13] Vladimir Lifschitz, Norman McCain, Emilio Remolina, and Armando Tacchella. Getting to the airport: the oldest planning problem in ai. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, pages 147–165. Kluwer, 2000.

[14] Vladimir Lifschitz and Wanwan Ren. A modular action description language. In *AAAI*, 2006.

[15] Victor W. Marek and Miroslaw Truszczynski. *Stable models and an alternative logic programming paradigm*, pages 375–398. The Logic Programming Paradigm: a 25-Year Perspective. Springer Verlag, Berlin, 1999.

[16] I. Niemela and P. Simons. Extending the Smodels System with Cardinality and Weight Constraints. In *Logics in Artificial Intelligence*. Kluwer Academic Publishers, 2000.

[17] Ilkka Niemela. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. In *Proceedings of the Workshop on Computational Aspects of Nonmonotonic Reasoning*, pages 72–79, Jun 1998.

[18] Ilkka Niemela and Patrik Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR'97)*, volume 1265 of *Lecture Notes in Artificial Intelligence (LNCS)*, pages 420–429, 1997.

[19] Yana Todorova. Answering questions about dynamic domains from natural language using ASP. PhD Dissertation, 2011.

[20] H. Turner. Representing Actions In Logic Programs And Default Theories. *Journal of Logic Programming*, pages 245–298, 1997.