TLP **13** (4–5): Online Supplement, July 2013. © 2013 [ROLF SCHWITTER]
URL: http://dx.doi.org/10.1017/S1471068413000306

The jobs puzzle: Taking on the challenge via controlled natural language processing

ROLF SCHWITTER

Macquarie University, Department of Computing, Sydney NSW 2109, Australia
(*e-mail:* `Rolf.Schwitter@mq.edu.au`)

submitted 10 April 2013; accepted 23 May 2013

Abstract

In this paper we take on Stuart C. Shapiro's challenge of solving the Jobs Puzzle automatically and do this via controlled natural language processing. Instead of encoding the puzzle in a formal language that might be difficult to use and understand, we employ a controlled natural language as a high-level specification language that adheres closely to the original notation of the puzzle and allows us to reconstruct the puzzle in a machine-processable way and add missing and implicit information to the problem description. We show how the resulting specification can be translated into an answer set program and be processed by a state-of-the-art answer set solver to find the solutions to the puzzle.

KEYWORDS: Controlled natural language, knowledge representation, answer set programming

1 Introduction

In a recent paper (Shapiro 2011), the author identified the Jobs Puzzle (Wos *et al.* 1984) as a *challenge for logical expressibility and automated reasoning* and presented three formalisations that require a less difficult and less tedious encoding than the original formalisation by Wos and his co-authors. The Jobs Puzzle is hard to solve automatically because the problem description in natural language is not complete and contains ambiguous information:

1. There are four people: Roberta, Thelma, Steve, and Pete.
2. Among them, they hold eight different jobs.
3. Each holds exactly two jobs.
4. The jobs are: chef, guard, nurse, telephone operator, police officer (gender not implied), teacher, actor, and boxer.
5. The job of nurse is held by a male.
6. The husband of the chef is the telephone operator.
7. Roberta is not a boxer.
8. Pete has no education past the ninth grade.
9. Roberta, the chef, and the police officer went golfing together.

Question: Who holds which jobs?

If the missing and implicit information is not uncovered, and the ambiguity is not cleared, then the puzzle cannot be solved by an automated reasoner (Wos *et al.* 1984). Shapiro's challenge to the community is three-fold (Shapiro 2011):

(a) formalise the puzzle in a way that is neither difficult nor tedious,
(b) formalise the puzzle so that it adheres closely to the English statements of the puzzle, and
(c) have an automated general-purpose commonsense reasoner that solves the puzzle efficiently.

Shapiro discusses three formalisations of the puzzle: one in TPTP syntax (Sutcliffe 2009) that relies on equality predicates and special-purpose axioms in order to deal with the unique name assumption; one in SNePSLOG syntax (Shapiro *et al.* 2010) that uses general quantifiers and set arguments but requires the translation of some statements into contrapositives; and one in Lparse syntax (Syrjänen 2000) that relies on an extended logic programming notation with cardinality expressions and integrity constraints. None of these formalisations satisfy the requirement (b) because there is still a considerable conceptual gap between the formal notations and the English statements of the puzzle.

In this paper we show that it is possible to specify the relevant knowledge that is necessary to solve the puzzle as a series of statements in a controlled natural language (White and Schwitter 2009; Schwitter 2010; Schwitter 2012). The resulting specification is then translated automatically into an Answer Set Program (ASP) (Gelfond and Lifschitz 1988; Lifschitz 2008; Brewka *et al.* 2011) and executed by a state-of-the-art ASP solver (Gebser *et al.* 2011), similar to the Lparse solution. Note that there is no need to encode the puzzle in a formal notation; the focus here is on the linguistic reconstruction of the puzzle rather than on its formal encoding.

Our approach is different to (Baral and Dzifcak 2012) who try to learn suitable ASP representations for clues of combinatorial logic puzzles from training data. The authors assume that the domain of a puzzle is given and that the representation of this domain follows a specific ontological structure that is common to most puzzles. They manually preprocess and simplify the clues of the puzzles and focus only on the accurate translation of these clues. They observe that many clues are very specific and that it is hard to figure them out automatically using additional background knowledge. Their approach does not meet requirement (b) since their formal representation is designed to exploit the similarity of clues of other puzzles and therefore does not adhere closely to the English statements of the puzzles. Other researchers (Lierler and Görz 2006; Balduccini *et al.* 2008; Todorova 2009) have used ASP for reasoning from natural language but not for solving logic puzzles. Constraint Lingo (Finkel *et al.* 2004) is another approach that has been used for solving logic puzzles but it requires the programmer to convert a puzzle into a tabular representation that abstracts away from the problem description in natural language.

The rest of this paper is organised as follows: In Section 2, we introduce and discuss Shaprio's Lparse/Smodels solution to the puzzle. In Section 3, we show

how the puzzle can be reconstructed in controlled natural language and provide the corresponding Lparse translation. In Section 4, we discuss a number of possible improvements to our solution. In Section 5, we focus on the language processor that is used to translate the controlled natural language via discourse representation structures into an ASP program. Finally, in Section 6, we summarise the advantages of our approach and conclude.

2 Shapiro's solution via Lparse/Smodels

Let us first have a look at Shapiro's solution to the Jobs Puzzle that came closest to meeting the challenge. This solution is based on Lparse syntax and the ASP tools Lparse (Syrjänen 2000) and Smodels (Niemelä *et al.* 2000). Lparse is a grounding tool for the answer set solver Smodels and accepts extended normal logic programs in Lparse syntax as input.

The formalisation of sentence 1 of the puzzle results in a single atom with alternative terms pooled together (this notation is equivalent to four individual atoms):

```
A.1.0  person(roberta ; thelma ; steve ; pete).
```

Note that the four names imply the gender of the persons: Roberta and Thelma are female, and Steve and Pete are male. This implicit knowledge is not directly available from the problem description and needs to be made explicit in order to solve the puzzle by a reasoning tool (Wos *et al.* 1984):

```
A.1.1  female(roberta ; thelma).
A.1.2  male(steve ; pete).
```

The problem description also assumes that humans know that no person can be both male and female; this implicit knowledge can be represented via an integrity constraint:

```
A.1.3  :- person(X), male(X), female(X).
```

Sentence 2 states that the four persons have eight different jobs among them. Shapiro's formalisation does not mention the total number of jobs since these jobs are enumerated in sentence 4. Instead the formalisation states that for every job there is exactly one person who holds that job, and this is represented via a choice rule with a cardinality expression:

```
A.2.0  1 {hasJob(X,Y) : person(X)} 1 :- job(Y).
```

Sentence 3 specifies that each person holds exactly two jobs; this knowledge is again formalised via a choice rule with a cardinality expression:

```
A.3.0  2 {hasJob(X,Y) : job(Y)} 2 :- person(X).
```

Sentence 4 enumerates the eight job names; this knowledge can be represented by a single atom where the alternative job names are pooled together:

```
A.4.0  job(chef ; guard ; nurse ; operator ; police ; teacher ;
          actor ; boxer).
```

Sentence 5 specifies that if a person has a job as nurse then that person is male; this is formalised via a basic rule:

```
A.5.0  male(X) :- person(X), hasJob(X,nurse).
```

It is further assumed that humans can – due to their linguistic knowledge – make the inference that actors are male; this knowledge is formalised in a similar way as A.5.0:

```
A.5.1  male(X) :- person(X), hasJob(X,actor).
```

Sentence 6 states that the husband of the chef is the telephone operator; this is formalised via a basic rule:

```
A.6.0  hasHusband(Y,X) :- person(X ; Y), hasJob(Y,chef),
          hasJob(X,operator).
```

Since a person X has another person Y as a husband, it is further assumed that humans can infer that X must be female and that Y must be male. This inference is made explicit in Shapiro's solution via the following choice rule:

```
A.6.1  2 {female(X), male(Y)} 2 :- person(X ; Y), hasHusband(X,Y).
```

Sentence 7 states that Roberta is not a boxer, this knowledge is formalised via an integrity constraint that excludes the corresponding atom from any satisfying model:

```
A.7.0  :- hasJob(roberta,boxer).
```

Sentence 8 basically says that Pete is not educated; this is again formalised via an integrity constraint:

```
A.8.0  :- educated(pete).
```

The problem description further assumes that humans know due to their cultural knowledge that the jobs of nurse, police officer and teacher require more than a ninth-grade education. In Shapiro's solution this knowledge is expressed via a basic rule with a cardinality expression in the body (this is a way of putting a disjunction in the body of an ASP rule). To count as educated at least one or at most two common instances of `hasJob/2` must occur in a satisfying model:

```
A.8.1  educated(X) :- 1 {hasJob(X,nurse), hasJob(X,police),
          hasJob(X,teacher)} 2, person(X).
```

Sentence 9 is ambiguous and has to be interpreted as Roberta, [and] the chef, and the police officer went golfing together so that a human can infer that Roberta is neither the chef nor the police officer. This knowledge can be formalised with a choice rule that excludes the two corresponding atoms:

```
A.9.1  0 {hasJob(roberta,chef), hasJob(roberta,police)} 0.
```

Since the chef and the police officer went golfing together, a human can also infer that the same person cannot be chef and police officer at the same time. In Shapiro's solution this is modeled via a choice rule that makes sure that only one instance of a `hasJob/2` atom for a given person and the identified two jobs can occur in a satisfying model:

```
A.9.2   0 {hasJob(X,chef), hasJob(X,police)} 1 :- person(X).
```

Finally, the `hide` declaration below marks all atoms of a model as hidden, and the `show` declaration specifies which atoms of the model should be included in the output:

```
#hide.
#show hasJob(X,Y).
```

After grounding the ASP program with Lparse, the ASP solver Smodels generates a satisfying model and displays the following instances for `hasJob/2` as solutions:

```
hasJob(pete,operator)
hasJob(pete,actor)
hasJob(steve,nurse)
hasJob(steve,police)
hasJob(thelma,chef)
hasJob(thelma,boxer)
hasJob(roberta,guard)
hasJob(roberta,teacher)
```

3 Solution via controlled natural language processing

Shapiro's solution requires an encoding of the puzzle that makes the missing and implicit knowledge explicit in a formal language. We show in this section that a controlled natural language can be used as a high-level specification language instead of a formal language to specify the same kind of knowledge but in a familiar notation that adheres – as required in the challenge – closely to the English statements of the puzzle.

We start our reconstruction of the puzzle using the **simplest possible** subset of the controlled natural language PENG Light (White and Schwitter 2009). This subset allows us to express just the relevant facts, rules, and constraints of the puzzle. We will then extend this subset in the subsequent section and further align it with the ASP language in order to gain more flexibility and achieve a more compact representation of the puzzle. Additional linguistic structures that we do not discuss here but belong to the controlled natural language PENG Light are necessary to process other puzzles (Schwitter 2012).

In our case the controlled natural language consists of simple sentences, complex sentences, and questions. The simple sentences can be introduced – for our purpose – with the help of eight linguistic patterns that are displayed in Table 1 together with corresponding examples. Note that `PNoun` stands for a proper noun that represents a

Table 1. *Syntax of simple sentences*

Pattern	Example
`PNoun` **is a** `CNoun`.	Roberta is a person.
`PNoun` **is** `Adjective`.	Roberta is female.
There is a `CNoun`.	There is a job.
A `CNoun` **is** `Adjective`.	A person is female.
A `OrdNumber CNoun` **is a** `CNoun` **of a** `OrdNumber CNoun`.	A first person is a husband of a second person.
A `CNoun Verb` **a** `CNoun` **as** `PNoun`.	A person holds a job as nurse.
`CardRest CNoun Verb` **a** `CNoun`.	Exactly one person holds a job.
A `CNoun Verb CardRest CNoun`.	A person holds exactly two jobs.

unique entity and `CNoun` stands for a common noun that represents a class of entities. Note also that only a single cardinality restriction (`CardRest`) can occur in a simple sentence. These simple sentences build the starting point for the reconstruction of the puzzle. Complex sentences are built from simple sentences with the help of coordination and the following two patterns:

1. **If** `Simple Sentence` [**and** `Simple Sentence`]* **then** `Simple Sentence`.
2. **Exclude that** `Simple Sentence` [**and that** `Simple Sentence`]*.

Questions are derived from simple sentences using the usual formation rules. Only specific forms of anaphoric expressions are allowed in our controlled natural language: definite descriptions can be used to link back to an indefinite noun phrases (e.g., *a person* ← *the person*). Ordinal numbers can be used in these noun phrases to distinguish between different instances that have the same form and to support the anaphora resolution process (e.g., *a first person* ... *a second person* ... *the first person*).

Using our simple controlled natural language, the reconstruction of sentence 1 of the puzzle results in the following simple sentences in B.1.0. The implicit knowledge about the gender of these persons is expressed by the simple sentences in B.1.1 and B.1.2. All these sentences are then translated automatically into individual atoms. The implicit knowledge that a person cannot be male and female at the same time is expressed by the complex sentence B.1.3 whose translation results in an integrity constraint:

B.1.0 Roberta is a person. Thelma is a person. Steve is a person. Pete is a person.
```
person(roberta). person(thelma). person(steve). person(pete).
```

B.1.1 Roberta is female. Thelma is female.
```
female(roberta). female(thelma).
```

B.1.2 Steve is male. Pete is male.
```
male(steve). male(pete).
```

B.1.3 Exclude that a person is male and that the person is female.

```
:- person(X), male(X), female(X).
```

The reconstruction of sentences 2 and 3 leads to the following two conditional sentences B.2.0 and B.3.0. These sentences are then translated into choice rules with cardinality expressions similar to A.2.0 and A.3.0; however, we use the original name of the verb (*hold*) as a predicate name:

B.2.0 If there is a job then exactly one person holds the job.

```
1 {hold(X,Y) : person(X)} 1 :- job(Y).
```

B.3.0 If there is a person then the person holds exactly two jobs.

```
2 {hold(X,Y) : job(Y)} 2 :- person(X).
```

The reconstruction of sentence 4 is similar to that of sentence 1 and gives rise to a series of simple sentences in B.4.0 that are finally translated into single atoms:

B.4.0 Chef is a job. Guard is a job. Nurse is a job. ...

```
job(chef). job(guard). job(nurse). ...
```

The reconstruction of sentence 5 leads to a conditional sentence (B.5.0) as well as to additional knowledge that identifies an actor as male (B.5.1). Note that in contrast to A.5.0, the translation of *a job as nurse* in B.5.0 results in a different representation since the formalisation closely follows the linguistic surface structure of the controlled language. The same is the case for the translation of sentence B.5.1 and all the subsequent sentences that contain this structure:

B.5.0 If a person holds a job as nurse then the person is male.

```
male(X) :- person(X), job(nurse), hold(X,nurse).
```

B.5.1 If a person holds a job as actor then the person is male.

```
male(X) :- person(X), job(actor), hold(X,actor).
```

The reconstruction of sentence 6 leads to the conditional sentence B.6.0 as well as to the specification of additional knowledge that identifies the subject of a husband as male in B.6.1 and the object as female in B.6.2. Note that the translation of this additional knowledge gives – for the time being – rise to two rules (in contrast to the formalisation in A.6.1):

B.6.0 If a first person holds a job as chef and a second person holds a job as telephone
operator then the second person is a husband of the first person.

```
husband(Y,X) :- person(X), job(chef), hold(X,chef), person(Y),
   job(operator), hold(Y,operator).
```

B.6.1 If a first person is a husband of a second person then the first person is male.

```
male(X) :- person(X), person(Y), husband(X,Y).
```

B.6.2 If a first person is a husband of a second person then the second person is female.
```
female(Y) :- person(X), person(Y), husband(X,Y).
```

The reconstruction of sentence 7 results in the complex sentence B.7.0 that is translated into an integrity constraint:

B.7.0 Exclude that Roberta holds a job as boxer.
```
:- job(boxer), hold(roberta,boxer).
```

Similarly, the reconstruction of sentence 8 results in a complex sentence (B.8.0). The inferred knowledge about jobs and the required level of education is then specified via three conditional sentences (B.8.1, B.8.2 and B.8.3). This is different to A.8.1 where only one rule is used instead of three:

B.8.0 Exclude that Pete is educated.
```
:- educated(pete).
```

B.8.1 If a person holds a job as nurse then the person is educated.
```
educated(X) :- person(X), job(nurse), hold(X,nurse).
```

B.8.2 If a person holds a job as police officer then the person is educated.
```
educated(X) :- person(X), job(police), hold(X,police).
```

B.8.3 If a person holds a job as teacher then the person is educated.
```
educated(X) :- person(X), job(teacher), hold(X,teacher).
```

The reconstruction of the inferred knowledge of sentence 9 leads to three complex sentences that are all translated into integrity constraints. In contrast to A.9.1, the two sentences B.9.1^a and B.9.1^b are used to invoke integrity constraints:

B.9.1^a Exclude that Roberta holds a job as chef.
```
:- job(chef), hold(roberta,chef).
```

B.9.1^b Exclude that Roberta holds a job as police officer.
```
:- job(police), hold(roberta,police).
```

B.9.2 Exclude that a person holds a job as chef and that the person holds a job as police officer.
```
:- person(X), job(chef), hold(X,chef), job(police),
   hold(X,police).
```

In contrast to Shapiro's solution, we also process questions and translate them into basic rules with a predefined answer literal (`answer/1`) as head:

Question: Who holds which jobs?
```
answer(hold(X,Y)) :- job(Y), hold(X,Y).
```

We then display these generic answer literals in the ASP program using the `show` declaration.

4 Towards a more compact specification

We now introduce a number of additional linguistic constructions that are part of the controlled natural language PENG Light and show how these constructions contribute to a more compact specification of the puzzle on the level of the controlled language. These constructions consist of the universal quantifier *every*, relative clauses and additional forms of coordination. It is important to note that in PENG Light, conditional sentences and universally quantified sentences are interpreted in the same way. The textual order of a quantifier defines its scope, and the occurrence of a quantifier in a sentence opens its scope that extends to the end of the sentence. As we will see, quantifier raising can then be used to move a designated quantified noun phrase to a different position in a sentence.

As a first step towards a more compact representation, we first relate job names in a systematic way to their professional categories (e.g., *chef* to *a chef*). We do this with the help of a universal quantified sentence together with a subject-modifying relative clause and specify for each professional category the necessary conditions that must hold for a person to be a member of that category, for example:

C.0.1 Every person who holds a job as chef is a chef.

```
chef(X) :- person(X), job(chef), hold(X,chef).
```

As we will see shortly, these categories will allow us to write the specification in a more compact way, but let's discuss the suggested improvements in the order of the given sentences.

The sentences in B.1.0, B.1.1 and B.1.2 are all simple sentences and have a similar structure. We can reformulate these sentences in a more compact way by coordinating the names in the subject position. The translation of these sentences then results in a representation with pooled arguments within a single atom:

C.1.0 Roberta, Thelma, Steve, and Pete are persons.

```
person(roberta ; thelma ; steve ; pete).
```

C.1.1 Roberta and Thelma are female.

```
female(roberta ; thelma).
```

C.1.2 Steve and Pete are male.

```
male(steve ; pete).
```

Sentence B.1.3 coordinates two simple relative sentences and uses an anaphoric expression in the second one. We can replace this construction by coordinating two verb phrases instead and end up after the translation with the same representation as before:

C.1.3 Exclude that a person is male and is female.

```
:- person(X), male(X), female(X).
```

Sentence B.2.0 is a conditional sentence that contains an anaphoric expression in the consequence. We can replace this conditional sentence by a universally quantified sentence; but this requires quantifier raising using the predefined expressions *for every* and *there is* and a relative clause to get the correct scope of the quantifiers (and this results after translation in the same representation as B.2.0):

C.2.0 For every job there is exactly one person who holds the job.
```
1 {hold(X,Y) : person(X)} 1 :- job(Y).
```

Sentence B.3.0 is another conditional sentence that can be replaced by a universally quantified one (and finally results after translation in the same representation as B.3.0):

C.3.0 Every person holds exactly two jobs.
```
2 {hold(X,Y) : job(Y)} 2 :- person(X).
```

The simple sentences in B.4.0 can be replaced by one that coordinates all job names in subject position:

C.4.0 Chef, guard, nurse, telephone operator, police officer, teacher, actor, and boxer are jobs.
```
job(chef ; guard ; nurse ; operator ; police ; teacher ;
   actor ; boxer).
```

The two conditional sentences in B.5.1 and B.5.2 can be expressed as a single universally quantified sentence with embedded disjunctive relative clauses. Note that we can now use the common nouns *nurse* and *actor* that we introduced in C.0.1. The translation of this sentence leads to two basic rules because the relative clauses are coordinated by a disjunction:

C.5.0 Every person who is a nurse or who is an actor is male.
```
male(X) :- person(X), nurse(X).
male(X) :- person(X), actor(X).
```

Sentence B.6.0 introduces the relational noun *husband* but it turns out that only the information in C.6.1 and C.6.2 that specify the gender of chef and telephone operator are necessary to solve the puzzle. Sentence C.6.0 provides additional information about the husband relationship but does not contribute anything to the solution:

C.6.0 If there is a telephone operator and there is a chef then the telephone operator is the husband of the chef.
```
husband(X,Y) :- operator(X), chef(Y).
```

C.6.1 Every person who is a chef is female.
```
female(X) :- person(X), chef(X).
```

C.6.2 Every person who is a telephone operator is male.
```
male(X) :- person(X), operator(X).
```

Sentence B.7.0 is similar to sentence C.7.0, but we can now use the professional categories that we introduced in C.0.1 and specify the relevant knowledge in a more compact way:

C.7.0 Exclude that Roberta is a boxer.
```
:- boxer(roberta).
```

One could argue that *Roberta is not a boxer* is a more compact notation but this would not lead to the expected result since our controlled natural language distinguishes between integrity constraints (*exclude that*), strong negation (*not*) and weak negation (*not provably*) on the surface level. ASP rules with negation as failure in the head are strongly equivalent to constraints; therefore we would have to state *Roberta is not provably a boxer* instead of *Roberta is not a boxer*. The same applies to sentence C.8.0 that is identical to B.8.0:

C.8.0 Exclude that Pete is educated.

```
:- educated(pete).
```

The three conditional sentences in B.8.1, B.8.2 and B.8.3 can be expressed as a single universally quantified sentence that contains three disjunctive relative clauses. The translation of this disjunction triggers three different rules (and is easier to generate automatically than the representation in A.8.1):

C.8.1 Every person who is a nurse or who is a police officer or who is a teacher is educated.

```
educated(X) :- person(X), nurse(X).
educated(X) :- person(X), police(X).
educated(X) :- person(X), teacher(X).
```

The two sentences in B.9.1[a] and B.9.2[b] can be reformulated and expressed as C.9.1 using sentence coordination. The translation of this complex sentence results in two integrity constraints (and not in a choice rule as in A.9.1):

C.9.1 Exclude that Roberta is a chef and exclude that Roberta is a police officer.

```
:- chef(roberta).
:- police(roberta).
```

Sentence B.9.2 uses two coordinated sentences; we can now express the same information in a more compact way with the help of two coordinated verb phrases:

C.9.2 Exclude that a person is a chef and is a police officer.

```
:- person(X), chef(X), police(X).
```

The question *Who holds which jobs?* is translated in the same way as before. In contrast to Shapiro who relies on Lparse/Smodels, we use the answer set tool *clingo* (Gebser *et al.* 2011) that combines the grounder and the solver in a single program and communicates with the controlled language processor in order to compute the answers to the question:

```
answer(holds(thelma,chef))
answer(holds(roberta,guard))
answer(holds(steve,nurse))
answer(holds(pete,operator))
answer(holds(steve,police))
answer(holds(roberta,teacher))
answer(holds(pete,actor))
answer(holds(thelma,boxer))
```

We were not able to measure any differences in wall-clock times between Shapiro's ASP program, our first ASP program and our second ASP program that is based on a more compact specification. However, a closer inspection of the internal representation in *clingo* reveals that Shapiro's ASP program results in twice as much choices as our first ASP program and that our second ASP program results in no choices at all and can be solved without any search.

5 Controlled natural language processing

The reconstruction of the puzzle in controlled language is supported by a predictive text editor (Schwitter *et al.* 2003) that enforces the restrictions of the controlled language during the writing process. For each word form that the author enters, the language processor generates look-ahead information that is displayed in the text editor and informs the author which categories and word forms can follow the current input. In this way the author can only construct syntactically correct sentences that are part of the controlled language.

The language processor consists of a chart parser, a unification-based grammar and a lexicon, and communicates with the predictive editor and the answer set tool *clingo* (Gebser *et al.* 2011). The chart parser processes the controlled language specification of the Jobs Puzzle incrementally, resolves anaphoric references on the fly and generates an (extended) discourse representation structure (DRS) in the spirit of (Kamp and Reyle 1993; van Eijck and Kamp 2011) during the parsing process.

In our case, a DRS is a term of the form `drs(U,C)`. The first argument `U` is a list of discourse referents (i.e. quantified variables), and the second argument `C` is a list of simple and complex conditions for these discourse referents. Simple conditions are logical atoms and complex conditions are built from other DRSs with the help of logical connectors (i.e. negation, disjunction, implication) and a non-standard operator for constraints. Our DRS uses a reified notation for the logical atoms together with a small number of predefined predicates. This DRS can be translated automatically into an ASP program, whenever a new sentence has been added to the specification. For example, after processing the first two sentences C.0.1 and C.1.0, the DRS will be as follows:

```
[A,B,C,D,E,F]
   [G,H,I]
   object(G,person)
   object(H,job)
   named(H,chef)
   predicate(I,hold,G,H)
   ==>
      [J,K]
      object(J,chef)
      predicate(K,isa,G,J)
named(A,roberta)
named(B,thelma)
named(C,steve)
named(D,pete)
object(E,person)
predicate(F,isa,(A;B;C;D),E)
```

The implicative condition derived from the universally quantified sentence is then translated into a basic ASP rule (see C.0.1), and the simple conditions derived from the second sentence are translated into a single atom where the alternative names are pooled together (see C.1.0).

Also the two universally quantified sentences C.2.0 and C.3.0 result both in an implicative condition but an additional condition (`cardinal/3`) is used to represent the cardinality restriction as the example for C.3.0 illustrates:

```
[M2]
object(M2,person)
==>
   [N2,O2]
   cardinal(N2,eq,2)
   object(N2,job)
   predicate(O2,hold,M2,N2)
```

The subsequent translation of this implicative condition results in a choice rule with a cardinality expression. The two universally quantified sentences C.5.0 and C.8.1 with embedded disjunctive relative clauses lead both to an implicative condition with disjunctive conditions in the antecedent. This complex structure is then split up and translated into basic ASP rules as discussed in the last section.

Finally, for the representation of those sentences that introduce a constraint in ASP, we use a complex condition in the DRS involving a non-standard operator (CSTR). For example, sentence C.9.1 triggers two constraints via sentence coordination and translates into the following two complex conditions:

```
CSTR
   [T3,U3]
   object(T3,chef)
   predicate(U3,isa,A,T3)

CSTR
   [V3,W3]
   object(V3,police)
   predicate(W3,isa,A,V3)
```

During the translation of these DRSs into integrity constraints, the variable `A` is unified with the name `roberta` that has been introduced as a condition in the top DRS.

6 Conclusion

In this paper we took on Shapiro's challenge to formalise the Jobs Puzzle in a way that the representation to an automated reasoning program is neither difficult nor tedious to construct and that the specification adheres closely to the English statements of the original version of the puzzle. We achieved this by reconstructing the puzzle in a controlled natural language that consists of a well-defined subset of English. This reconstruction is necessary for the following four reasons: (a) we need to bring the problem description of the puzzle into a form that is machine-processable; (b) we need to add the missing information to the problem description; (c) we need to make the implicit knowledge explicit, and (d) we need to clear the ambiguity.

As we have seen, the reconstruction of the puzzle in controlled natural language can be made more or less compact depending on the approved linguistic constructions that we have at hand. The resulting specification can then be translated automatically into an answer set program in order to generate the answers to the problem as part of a satisfying model. To make this possible, we have implemented a new grammar and a utility that translates the Jobs Puzzle (and similar combinatorial puzzles that use additional linguistic constructions) first into a discourse representation structure and then into an executable answer set program. To the best of our knowledge, this is the first controlled natural language that can serve as a high-level specification language to answer set programs.

In summary: we addressed Shapiro's challenge by using a controlled natural language as a high-level specification language that looks at first glance informal and is easy to understand but has the same expressive power as the resulting answer set program.

References

Balduccini, M., Baral, C. and Lierler, Y. 2008. Knowledge representation and question answering. In *Handbook of Knowledge Representation*, F. van Harmelen, V. Lifschitz and B. Porter, Eds. Elsevier B. V., 779–819.

Baral, C. and Dzifcak, J. 2012. Solving puzzles described in english by automated translation to answer set programming and learning how to do that translation. In *Proceedings of KR 2012*, 573–577.

Brewka, G., Eiter, T. and Truszczyński, M. December, 2011. Answer set programming at a glance. *Communications of the ACM 54*, 12.

Finkel, R. Marek, V. W. and Truszczyński, M. 2004. Constraint Lingo: Towards high-level constraint programming. *Software: Practice and Experience 34*, 15, 1481–1504.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T. and Schneider, M. 2011. Potassco: The potsdam answer set solving collection. *AI Communications 24*, 2, 105–124.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of International Logic Programming Conference and Symposium*, R. Kowalski and K. Bowen, Eds. 1070–1080.

Kamp, H. and Reyle, U. 1993. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Dordrecht.

Lierler, Y. and Görz, G. 2006. Model generation for generalized quantifiers via answer set programming. In *Proceedings of 8th Conference on Natural Language Processing (KONVENS)*, 101–106.

Lifschitz, V. 2008. What is answer set programming? In *Proceedings of AAAI'08*, vol. 3, 1594–1597.

Niemelä, I., Simons, P. and Syrjänen, T. 2000. Smodels: A system for answer set programming. In *CoRR*, Vol. cs.AI/0003033.

Schwitter, R. 2010. Controlled natural language for knowledge representation. In *Proceedings of COLING 2010*, 1113–1121.

Schwitter, R. 2012. Answer set programming via controlled natural language processing. In *CNL 2012*, T. Kuhn and N. E. Fuchs, Eds., LNCS 7427, Springer, 26–43.

Schwitter, R., Ljungberg, A. and Hood, D. 2003. ECOLE - A look-ahead editor for a controlled language. In *Proceedings of EAMT-CLAW03*, May 15–17, Dublin City University, Ireland, 141–150.

Shapiro, S. C. 2011. The jobs puzzle: A challenge for logical expressibility and automated reasoning. In *Logical Formalizations of Commonsense Reasoning*, E. Davis, P. Doherty and E. Erdem, Eds., AAAI Press, Menlo Park, CA, 96–102.

Shapiro, S. C. and the SNePS Implementation Group. December 8, 2010. *SNePS 2.7.1 User's Manual*. Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY.

Sutcliffe, G. 2009. The TPTP problem library and associated infrastructure: The FOF and CNF parts, v3.5.0. *Journal of Automated Reasoning*, *43*, 4, 337–362.

Syrjänen, T. 2000. *Lparse 1.0, User's Manual*. Laboratory for Theoretical Computer Science, Helsinki University of Technology.

Todorova, Y. 2011. Answering questions about dynamic domains from natural language using ASP. *Dissertation*, Texas Tech University.

van Eijck, J. and Kamp, H. 2011. Discourse representation in context. In *Handbook of Logic and Language*, 2nd ed., J. van Benthem and A. ter Meulen, Eds. Elsevier, 181–252.

White, C. and Schwitter, R. 2009. An update on PENG light. In *Proceedings of ALTA 2009*, 80–88.

Wos, L., Overbeek, R., Lusk, E. and Boyle, J. 1984. *Automated Reasoning: Introduction and Applications*. Prentice-Hall, New Jersey.