The Jobs Puzzle
A Challenge for Logical Expressibility and Automated Reasoning

Stuart C. Shapiro
Department of Computer Science and Engineering and Center for Cognitive Science
The State University of New York at Buffalo
Buffalo, NY 14260-2000
shapiro@buffalo.edu

Abstract

The Jobs Puzzle, introduced in a book about automated reasoning, is a logic puzzle solvable by some "intelligent sixth graders," but the formalization of the puzzle by the authors was, according to them, "sometimes difficult and sometimes tedious." The puzzle thus presents a triple challenge: 1) formalize it in a non-difficult, non-tedious way; 2) formalize it in a way that adheres closely to the English statement of the puzzle; 3) have an automated general-purpose commonsense reasoner that can accept that formalization and solve the puzzle quickly. In this paper, I present and discuss three formalizations that are less difficult and less tedious than the original. However, none satisfy all three requirements as well as might be desired, and there are a significant number of automated reasoners that cannot solve the puzzle using any of the formalizations. So the Jobs Puzzle remains an interesting challenge.

1. Introduction

The Jobs Puzzle was introduced by Wos et al. (1984, pp. 44–78)[1] as [p. 44, numbering added]

> "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?"

In the next sections, Wos et al. discuss "The Solution by Person or Persons Unknown" [§3.2.1] and "The Solution by Program or Programs Known" [§3.2.2]. The "Program or Programs Known" was a resolution refutation theorem prover such as Otter (Kalman 2001; McCune and Wos 1997) was to become. In the discussion of "The Solution by Person or Persons Unknown," Wos et al. mention some immediate inferences that may be made in addition to those in the puzzle statement "based on common usage of everyday language" [p. 56], such as that Roberta is female and that the actor is male (because it's "actor," not "actress.") They suggest that the way people would solve the puzzle is by making a table whose rows are labeled with jobs and whose columns are labeled with people. As the solver reasons through the puzzle and decides which people hold which jobs and which couldn't possibly hold which jobs, she would write "yes" or "no" in the entries of the table. In the discussion of "The Solution by Program or Programs Known," Wos et al. represent the domain rules of the puzzle and the additional immediate inferences as clauses. In addition, "Clauses can be written to simulate the use of [the] table. Some simulate the table (and its labels). Others enable the program to cross off possibilities and to, in effect, fill in the squares" [p. 62]. They note that "some tedious but necessary items must be translated " [p. 60], and "Make no mistake, the representation of the problem to an automated reasoning program is sometimes difficult and sometimes tedious" [p.63].[2]

The challenge posed in this paper is to represent the Jobs Puzzle to an automated reasoning program, suitable for general-purpose commonsense reasoning, in a non-difficult, non-tedious way, by a series of logical formulae that adhere closely to the English statements of the puzzle and the allowed immediate inferences, and have that automated reasoning program solve the puz-

[1] In the remainder of this paper, every citation to sections or pages that omits mention of a work is a citation to (Wos et al. 1984).

[2] A statement of the puzzle, with "clerk" replacing "telephone operator", the correct answer, and the clauses suitable for input to OTTER are at http://www.mcs.anl.gov/~wos/mathproblems/jobs.html.

zle quickly.

In the remainder of this paper, I show and discuss three formalizations that more or less satisfy these requirements.

2. The Solution by TPTP Participants

One non-difficult and relatively non-tedious formalization of the Jobs Puzzle is given as problem PUZ019-1 in the TPTP (Thousands of Problems for Theorem Provers)[3] version 5.1.0 web site[4]. The formalization is given as a sequence of clauses, but for clarity, I will use a more standard FOL syntax. There are 64 clauses, four of which are non-Horn clauses. Rather than using "=" and paramodulation, two special-purpose equality predicates are used: $equal\_people$ and $equal\_jobs$. First are four clauses stating the reflexivity and symmetry of the equality predicates:

$$\forall(x)(equal\_people(x,x) \wedge equal\_jobs(x,x))$$

(Note that this implies that jobs are equal to themselves as people, and people are equal to themselves as jobs.)

$$\forall(x,y)(equal\_people(x,y) \Rightarrow equal\_people(y,x))$$
$$\forall(x,y)(equal\_jobs(x,y) \Rightarrow equal\_jobs(y,x))$$

Then, rather than making a unique-names assumption, 34 special-purpose nonequality axioms are given, such as

$$\neg equal\_people(roberta, thelma)$$
$$\neg equal\_jobs(chef, guard).$$

Finally, 25 clauses come from the statement of the puzzle, and one clause from the query. The formal axioms as presented below are preceded by English statements labeled "*jp*" for sentences coming directly from the statement of the Jobs Puzzle or "*inf*" for immediate inferences allowed by (Wos et al. 1984).

1. *jp: There are four people: Roberta, Thelma, Steve, and Pete.*

$$\forall x(has\_job(roberta,x) \vee has\_job(thelma,x) \vee has\_job(pete,x) \vee has\_job(steve,x))$$

inf: "if the four names did not clearly imply the sex of the people, [the puzzle] would be impossible to solve." [p. 56]

$$\forall x((male(x) \vee female(x)) \wedge \neg(male(x) \wedge female(x)))$$

(Note that this also implies that each job is male or female.)

$$male(steve) \wedge male(pete) \wedge female(roberta) \wedge female(thelma)$$

2. *jp: Among [the people], they hold eight different jobs.*

4. *jp: The jobs are: chef, guard, nurse, telephone operator, police officer (gender not implied), teacher, actor, and boxer.*

$$\forall x(has\_job(x,chef) \vee has\_job(x,guard) \vee has\_job(x,nurse) \vee has\_job(x,operator) \vee has\_job(x,police) \vee has\_job(x,teacher) \vee has\_job(x,actor) \vee has\_job(x,boxer))$$

3. *jp: Each holds exactly two jobs.*

$$\forall(x,y,z,u)(has\_job(z,y) \wedge has\_job(z,x) \wedge has\_job(z,u) \Rightarrow equal\_jobs(x,y) \vee equal\_jobs(u,y) \vee equal\_jobs(u,x))$$

inf: "No job is held by more than one person." [p. 56]

$$\forall(x,y,z)(has\_job(x,z) \wedge has\_job(y,z) \Rightarrow equal\_people(x,y))$$

5. *jp: The job of nurse is held by a male.*

$$\forall x(has\_job(x,nurse) \Rightarrow male(x))$$

inf: "everyday language distinguishes [actors and actresses] based on sex." [p. 56]

$$\forall x(has\_job(x,actor) \Rightarrow male(x))$$

6. *jp: The husband of the chef is the telephone operator.*

$$\forall x(has\_job(x,chef) \Rightarrow \forall y(husband(x,y) \Leftrightarrow has\_job(y,operator)))$$

inf: "the implicit fact that husbands are male" [p. 57]

$$\forall(x,y)(husband(x,y) \Rightarrow female(x) \wedge male(y))$$

inf: since the chef has a husband, she must be female. [p. 57]

$$\forall x(has\_job(x,chef) \Rightarrow female(x))$$

7. *jp: Roberta is not a boxer.*

$$\neg has\_job(roberta, boxer)$$

8. *jp: Pete has no education past the ninth grade.*

$$\neg educated(pete)$$

inf: "the jobs of nurse, police officer, and teacher each require more than a ninth-grade education." [p. 57]

$$\forall x(has\_job(x,nurse) \vee has\_job(x,police) \vee has\_job(x,teacher) \Rightarrow educated(x))$$

9. *jp: Roberta, the chef, and the police officer went golfing together.*
inf: "Thus, we know that Roberta is neither the chef nor the police officer." [p. 57]

$$\neg(has\_job(roberta,chef) \vee has\_job(roberta,police))$$

inf: "Since they went golfing together, the chef and the police officer are not the same person." [p. 57]

$$\forall x\neg(has\_job(x,chef) \wedge has\_job(x,police))$$

[3] http://www.tptp.org/

[4] http://tinyurl.com/jobsPuzzle

jp: Question: Who holds which jobs?

$$\exists(x1,x2,x3,x4,x5,x6,x7,x8)(has\_job(x1,chef) \wedge has\_job(x2,guard) \wedge has\_job(x3,nurse) \wedge has\_job(x4,operator) \wedge has\_job(x5,police) \wedge has\_job(x6,teacher) \wedge has\_job(x7,actor) \wedge has\_job(x8,boxer))$$

Of 29 systems that tried this formulation of the Jobs Puzzle, 20 were successful.[5] For example, SNARK (Stickel, Waldinger, and Chaudhri undated; Stickel 2010) solved this formulation of the Jobs Puzzle using unit-resulting-resolution and hyperresolution in September of 2010, after having previously failed to prove it without using unit-resulting-resolution [Mark Stickel, personal communication].

3. The Solution by Constraint Lingo

Constraint Lingo (Finkel, Marek, and Truszczyński 2002; 2004) is a high-level language for specifying a single relation via requirements and constraints. The specified relation is conceived of as a table whose i^{th} column contains entries from a specified i^{th} domain, and each of whose rows is one n-tuple in the relation. (One table entry may contain a set of elements from the appropriate domain.) The Constraint Lingo specification is translated into one of several back-end reasoners. The solution is then translated back into a table. Notice that this table does not have the same rows and columns as the table discussed in (Wos et al. 1984, §3.2.1).

A Constraint Lingo solution to the Jobs Puzzle, using lparse/smodels (Syrjänen 1998; 2000; Niemelä and Simons 2000) as the back-end, was provided to the author by Raphael Finkel [personal communication], but has been omitted from this paper due to space constraints, and because a solution directly in lparse/smodels is given below in §5. The Constraint Lingo solution is available from the author.[6]

4. The Solution by SNePS

SNePS (Shapiro and Rapaport 1992; Shapiro 2000) was designed for commonsense reasoning and natural language competence, rather than to be a high-powered theorem prover. An important design criterion was to have a formal logical language that captured the expressibility of English statements. Thus, the Jobs Puzzle is a natural example problem for SNePS, and has been distributed with SNePS[7] as a standard demonstration for a number of years. The formalization shown here uses the SNePSLOG front-end (Shapiro and The SNePS Implementation Group 2010, Chap. 6) and is for the latest version of SNePS, SNePS 2.7.1 (Shapiro and The SNePS Implementation Group 2010), which includes all the connectives discussed in (Shapiro 2010).

SNePS does not use clauses and resolution, but represents the axioms in the way they are entered and uses natural deduction. We have felt that there is heuristic information in the way that the user formalizes the information that would be lost in a canonicalization into clause form. For instance, modus ponens is implemented in SNePS, but modus tollens is not,[8] so `p => q` is treated differently from `~q => ~p`, though a user who wanted both modus ponens and modus tollens could enter `or{~p, q}` instead. Because modus tollens is not implemented, the Jobs Puzzle is formulated with `hasJob` predicates only in consequent position.

SNePS has the unique names assumption built in, which obviates the need for inequality axioms. In particular, the unique names assumption is used by the numerical quantifier (Shapiro 1979): `nexists(i,j,k)(x)(P(x): Q(x))` means that k individuals satisfy `P(x)`, and, of them, at least i and at most j also satisfy `Q(x)`. The unique names assumption is used when making these counts.

Other unique features of SNePS will be explained as they are used in the following formalization.

1. *jp: There are four people: Roberta, Thelma, Steve, and Pete.*

```
Person({Roberta, Thelma,
        Steve, Pete}).
```

If α is a set of terms and a is a term in α, then `P(`α`)` $\vdash$ `P(`a`)`. This is called "reduction inference" (Shapiro and The SNePS Implementation Group 2010, p. 65). So this axiom is a concise way to say that Roberta, Thelma, Steve, and Pete are all people.

inf: "if the four names did not clearly imply the sex of the people, [the puzzle] would be impossible to solve." [p. 56]

```
Female({Roberta, Thelma}).
Male({Steve, Pete}).
```

2. *jp: Among [the people], they hold eight different jobs.*

3. *jp: Each holds exactly two jobs.*

```
all(p)(Person(p)
     => nexists(2,2,8)(j)(Job(j):
                          hasJob(p,j))).
```

inf: "No job is held by more than one person." [p. 56]

```
all(j)(Job(j)
     => nexists(1,1,4)(p)(Person(p):
                          hasJob(p,j))).
```

4. *jp: The jobs are: chef, guard, nurse, telephone operator, police officer (gender not implied), teacher, actor, and boxer.*

[5] `http://tinyurl.com/TPTPpuzSolns`

[6] It is included in the Appendix to this version of this paper.

[7] `http://tinyurl.com/SNePSDownloads`

[8] For a full list of implemented rules of inference, see (Shapiro and The SNePS Implementation Group 2010, §6.4).

```
Job({chef, guard, nurse, operator,
     police, teacher, actor,
     boxer}).
```

5. *jp: The job of nurse is held by a male.*

```
all(w)(Female(w)
       => ~hasJob(w, nurse)).
```

inf: "everyday language distinguishes [actors and actresses] based on sex." [p. 56]

```
all(w)(Female(w)
       => ~hasJob(w, actor)).
```

6. *jp: The husband of the chef is the telephone operator.*
inf: "the implicit fact that husbands are male" [p. 57]

```
all(w)(Female(w)
       => ~hasJob(w, operator)).
```

inf: since the chef has a husband, she must be female. [p. 57]

```
all(m)(Male(m) => ~hasJob(m, chef)).
```

7. *jp: Roberta is not a boxer.*

```
~hasJob(Roberta, boxer).
```

8. *jp: Pete has no education past the ninth grade.*

```
~educated(Pete).
```

inf: "the jobs of nurse, police officer, and teacher each require more than a ninth-grade education." [p. 57]

```
all(x)(~educated(x)
       => nor{hasJob(x, nurse),
              hasJob(x, police),
              hasJob(x, teacher)}).
```

9. *jp: Roberta, the chef, and the police officer went golfing together.*
inf: "Thus, we know that Roberta is neither the chef nor the police officer." [p. 57]

```
nor{hasJob(Roberta, chef),
    hasJob(Roberta, police)}.
```

inf: "Since they went golfing together, the chef and the police officer are not the same person." [p. 57]

```
all(p)(Person(p)
       => nand{hasJob(p, chef),
               hasJob(p, police)}).
```

jp: Question: Who holds which jobs?

```
ask hasJob(?p, ?j)?
```

The SNePSLOG `ask` command triggers backward inference on its argument wff and prints all instances that are inferred. When run, what is printed is:

```
wff111!:  hasJob(Thelma,boxer)
wff101!:  hasJob(Pete,operator)
wff99!:  hasJob(Pete,actor)
wff87!:  hasJob(Steve,nurse)
wff85!:  hasJob(Roberta,guard)
wff83!:  hasJob(Roberta,teacher)
wff28!:  hasJob(Thelma,chef)
wff24!:  hasJob(Steve,police)
```

It took 0.16 seconds to infer and print these answers on a Dell Optiplex 780 minitower computer with 2 Intel(R) Core(TM)2 Duo CPU, clocked at 3.16 GHz, and with 4 GB of available system memory."

5. The Solution by Lparse/Smodels

Smodels (Niemelä and Simons 2000) is an implementation of the stable model semantics for logic programs. Essentially, it finds satisfying models of a set of ground clauses. Lparse (Syrjänen 1998; 2000) is a front-end to smodels that allows the clauses to be written in an extended logic programming syntax. The following solution is written in the language accepted by lparse. Nonobvious expressions are explained when first used.
1. *jp: There are four people: Roberta, Thelma, Steve, and Pete.*

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

`p(t1;...;tn)` is treated as the conjunction of `p(t1)`, and ..., and `p(tn)`, making this equivalent to the SNePSLOG `person({roberta,thelma,steve,pete}).`
2. *jp: Among [the people], they hold eight different jobs.*
3. *jp: Each holds exactly two jobs.*

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

This means that, for each person, there must be exactly two instances of `hasJob(X,Y)`, where `Y` is some job, making this equivalent to the SNePSLOG

```
all(x)(person(x)
       => nexists(2,2,8)(y)(job(y):
                            hasJob(x,y)))
```

except that the `8` is not specified, since negative instances are not inferred anyway.
inf: "No job is held by more than one person." [p. 56]

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

4. *jp: The jobs are: chef, guard, nurse, telephone operator, police officer (gender not implied), teacher, actor, and boxer.*

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

inf: "if the four names did not clearly imply the sex of the people, [the puzzle] would be impossible to solve." [p. 56]

```
female(roberta; thelma).
male(steve; pete).
```

No person is both male and female

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

A headless body indicates that a common instance of all body atoms is not to appear in any satisfying model.

5. *jp: The job of nurse is held by a male.*

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

inf: "everyday language distinguishes [actors and actresses] based on sex." [p. 56]

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

6. *jp: The husband of the chef is the telephone operator.*

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

inf: "the implicit fact that husbands are male" [p. 57]
inf: since the chef has a husband, she must be female. [p. 57]

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

That is, for each instance of `hasHusband(X,Y)`, where `X` and `Y` are people, that instance of both `female(X)` and `male(Y)` is to be included in each satisfying model.

7. *jp: Roberta is not a boxer.*

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

8. *jp: Pete has no education past the ninth grade.*

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

inf: "the jobs of nurse, police officer, and teacher each require more than a ninth-grade education." [p. 57]

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

The cardinality-constrained body group of atoms is a way of putting a disjunction in the body. The "2" is specified because it is known that no more than two common instances of these atoms could appear in any satisfying model.

9. *jp: Roberta, the chef, and the police officer went golfing together.*
inf: "Thus, we know that Roberta is neither the chef nor the police officer." [p. 57]

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

inf: "Since they went golfing together, the chef and the police officer are not the same person." [p. 57]

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

jp: Question: Who holds which jobs?

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

Together, these declarations indicate that only the instances of `hasJob(X,Y)` should be shown for each model.

After asking Smodels to show all the models, it reported only the correct one, and reported the computation time as "0.000".

6. Discussion

6.1 Discussion of the TPTP Solution

The remaining "tedious" aspect of the TPTP formalization of the Jobs Puzzle is the set of 38 clauses for the special-purpose equality and inequality axioms. These could be eliminated by making the unique names assumption and by using paramodulation. The remaining 25 clauses are quite straight-forward translations of the puzzle, although the formalizations of "*Each person holds at most two jobs*" and "*Each job is held by at most one person*" might be considered more clever than straight-forward.

The formulation does not include a $person$ or job predicate, and has unintended implications, such as

$equal\_people(chef, chef)$,
$equal\_jobs(roberta, roberta)$,

and

$(male(nurse) \vee female(nurse))$
$\wedge\neg(male(nurse) \wedge female(nurse))$.

There are four non-Horn clauses:

1. Everyone has at least one of the eight jobs.
2. Each job is held by one of the four people.
3. If someone seems to have three jobs, two of those jobs are the same.
4. Everyone is male or female.

Therefore, no reasoner limited to Horn clauses can solve this formulation of the puzzle. Of the 29 attempts to solve the puzzle using this formulation, 9 failed and 20 succeeded. Some of the successes were due to careful choices of strategies. For example, SNARK succeeded using unit-resulting-resolution, but before that was tried, SNARK failed [Mark Stickel, personal communication].

6.2 Discussion of the SNePS Solution

The SNePS formalization relies on several features specifically designed into SNePS to make SNePSLOG formulas closer to English statements than would otherwise be possible. Use of set arguments and reduction inference reduces the tedium of listing the four people, eight jobs, and the sexes of the people

in separate atomic formulas. The numerical quantifier, `nexists(`i,j,k`)(`x`)(`$P(x)$`: `$Q(x)$`)`, is a direct encoding of several kinds of generalized quantifiers (Barwise and Cooper 1981) and of predicate minimalization—once j Ps are found to be Qs, all other Ps are inferred to not be Qs, and once $k-i$ Ps are found not to be Qs, all other Ps are inferred to be Qs. The use of `nor` and `nand` (Shapiro 2010) makes a small reduction in the length and nesting of several axioms.

Leaving the formulas as stated, rather than translating them into some canonical form such as clauses, using natural deduction, and the omission of modus tollens (as well as several other apparently natural rules of inference), allows SNePS to focus its work on answering the given question, a focussing produced in resolution systems by careful choice of strategies. However, this requires some rewriting of some statements of the problem. For example, instead of formalizing "*The chef is female*" as

```
all(x)(hasJob(x,chef) => Female(x))
```

it is formalized as

```
all(x)(Male(x) => ~hasJob(x,chef))
```

This is the place where the SNePS formulas are least like the English statements they translate. However, this formalization also eliminates the need to say that every person is either male or female, but not both. The unique names assumption is made in the implementation of the numerical quantifier, and the two axioms that use it are the only two places where judgments of equality and inequality are required.

7. Discussion of the Lparse/Smodels Solution

Several noteworthy features of lparse/smodels are similar to features of SNePS. The reduction in tediousness achieved in SNePS by set arguments is achieved in lparse by conjunctive arguments separated by "`;`", and some of what is conveyed in SNePS by the numerical quantifier is conveyed in lparse/smodels by its cardinality constraints.

In formalizing "`The husband of the chef is the telephone operator`", not only was the obvious rule,

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

given, but also the less obvious

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

Notice that the TPTP solution also had clauses from both such rules. In fact, experimentation showed that smodels needed the second rule, but not the first.

Other than the non-obvious operator-is-husband rule, lparse/smodels satisfied the challenge well.

7.1 Some Failed Attempts

Kandefer and Shapiro (2008) attempted to represent the Jobs Puzzle in the Topbraid Ontology Editing Tool (Top Quadrant Inc. 2007) and solve it using the Pellet OWL Description Logic Reasoner (Clark & Parsia, LLC 2007), but were unsuccessful because Pellet is unable to infer positive instances from negative ones, as SNePS's numerical quantifier does (Shapiro 1979). An attempt to use SWRL (W3C 2004) was also unsuccessful because SWRL rules lack negation.

8. Conclusions

The Jobs Puzzle has been solved by "intelligent sixth graders" (Wos et al. 1984, p.55), but still presents a challenge for automatic reasoners. The challenge is three-fold:

1. Formalize the puzzle in a way that is neither difficult nor tedious.
2. Formalize the puzzle as a series of logical formulas that adhere closely to the English statement of the puzzle. (This would entail part (1).)
3. Have a general-purpose commonsense reasoning program that can accept that formalization, and solve the puzzle without further human assistance.

The original formalization, by the original posers of the puzzle, was, as admitted by them, "sometimes difficult and sometimes tedious." The TPTP formalization of the puzzle is less so, but some tedium remains, and some of the formalizations of some of the statements of the puzzle are more clever than they are direct translations. Nine of 29 recorded attempts to have automatic reasoners use this formalization to solve the puzzle failed, and no Horn-clause reasoner could possibly succeed. A formalization in SNePSLOG, using its generalized quantifier and set arguments, came quite close to a direct translation of the statements of the puzzle, but some statements needed to be translated into their contrapositives in order for SNePS to solve the puzzle. A formalization in lparse/smodels, using its conjunctive arguments and cardinality constraints came extremely close to meeting the challenge, needing only one "clever" rule. However, since smodels is a model-finder using what is essentially propositional logic, it might be argued that it is not a general-purpose commonsense reasoner. Attempts to solve the puzzle using a Description Logic reasoner failed, as did an attempt to formalize it using SWRL rules. Other attempts to meet the challenge are welcomed.

Acknowledgments

I am grateful to Mark Stickel for pointing me to TPTP, explaining the information contained there, and for discussions about SNARK. Inclusion of a Constraint Lingo solution was recommended by an anonymous reviewer

of this paper. I thank Raphael Finkel for supplying the solution and for discussions about Constraint Lingo, and for motivating me to investigate lparse/smodels. I apologize for having to omit that solution from the final version of this paper. I thank William J. Rapaport and Jonathan P. Bona for comments on earlier drafts of this paper, and to Christian Miller for telling me how to describe the computer on which SNePS solved the puzzle. I am grateful to present and past members of the University at Buffalo's SNePS Research Group for aiding in the implementation of SNePS, and for many years of fruitful and enjoyable collaboration. This work has been supported in part by a Multidisciplinary University Research Initiative (MURI) grant (Number W911NF-09-1-0392) for "Unified Research on Network-based Hard/Soft Information Fusion", issued by the US Army Research Office (ARO) under the program management of Dr. John Lavery.

References

Barwise, J., and Cooper, R. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy* 4(2):159–219. Reprinted in (Kulas, Fetzer, and Rankin 1988, 241–301).

Clark & Parsia, LLC. 2007. Pellet: The Open Source OWL DL Reasoner. `http://pellet.owldl.com/`.

Finkel, R.; Marek, V.; and Truszczyński, M. 2002. Constraint lingo: A program for solving logic puzzles and other tabular constraint problems. In Flesca, S.; Greco, S.; Ianni, G.; and Leone, N., eds., *Logics in Artificial Intelligence*, volume 2424 of *Lecture Notes in Computer Science*. Berlin / Heidelberg: Springer. 513–516.

Finkel, R.; Marek, V.; and Truszczyński, M. 2004. Constraint lingo: Towards high-level constraint programming. *Softward Practice and Experience* 34(15):1481–1504.

Kalman, J. A. 2001. *Automated Reasoning with Otter*. Princeton, NJ: Rinton Press.

Kandefer, M., and Shapiro, S. C. 2008. Comparing SNePS with Topbraid/Pellet. SNeRG Technical Note 42, Department of Computer Science and Engineering, The State University of New York at Buffalo, Buffalo, NY.

Kulas, J.; Fetzer, J. H.; and Rankin, T. L., eds. 1988. *Philosophy, Language, and Artificial Intelligence*. Studies in Cognitive Systems. Dordrecht: Kluwer.

Lehmann, F., ed. 1992. *Semantic Networks in Artificial Intelligence*. Oxford: Pergamon Press.

McCune, W., and Wos, L. 1997. Otter: The cade-13 competition incarnations. *Journal of Automated Reasoning* 18(211-220).

Niemelä, I., and Simons, P. 2000. Extending the smodels system with cardinality and weight constraints. In Minker, J., ed., *Logic-Based Artificial Intelligence*. Boston: Kluwer. 491–521.

Shapiro, S. C., and Rapaport, W. J. 1992. The SNePS family. *Computers & Mathematics with Applications* 23(2–5):243–275. Reprinted in (Lehmann 1992, pp. 243–275).

Shapiro, S. C., and The SNePS Implementation Group. 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. Available as `http://www.cse.buffalo.edu/sneps/Manuals/manual271.pdf`.

Shapiro, S. C. 1979. Numerical quantifiers and their use in reasoning with negative information. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann. 791–796.

Shapiro, S. C. 2000. SNePS: A logic for natural language understanding and commonsense reasoning. In Iwańska, Ł. M., and Shapiro, S. C., eds., *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. Menlo Park, CA: AAAI Press/The MIT Press. 175–195.

Shapiro, S. C. 2010. Set-oriented logical connectives: Syntax and semantics. In Lin, F.; Sattler, U.; and Truszczynski, M., eds., *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR2010)*, 593–595. Menlo Park, CA: AAAI Press.

Stickel, M. E.; Waldinger, R. J.; and Chaudhri, V. K. undated. A guide to SNARK. `http://www.ai.sri.com/snark/tutorial/tutorial.html`.

Stickel, M. E. 2010. SNARK - SRI's new automated reasoning kit. `http://www.ai.sri.com/~stickel/snark.html`.

Syrjänen, T. 1998. Implementation of local grounding for logic programs with stable model semantics. Technical Report B18, Digital Systems Laboratory, Helsinki University of Technology.

Syrjänen, T. 2000. *Lparse 1.0 User's Manual*. `http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz`.

Top Quadrant Inc. 2007. Topbraid Composer. `http://www.topbraidcomposer.com/`.

W3C. 2004. SWRL: A semantic web rule language. `http://www.w3.org/Submission/SWRL/`.

Wos, L.; Overbeek, R.; Lusk, E.; and Boyle, J. 1984. *Automated Reasoning: Introduction and Applications*. Englewood Cliffs, NJ: Prentice-Hall.

Appendix[9]

The Solution by Constraint Lingo

Constraint Lingo (Finkel, Marek, and Truszczyński 2002; 2004) is a high-level language for specifying a relation via requirements and constraints. The specified relation is conceived of as a table whose i^{th} column contains entries from a specified i^{th} domain, and each of whose rows is one n-tuple in the relation. (However, the specifications may allow one table entry to contain a set of elements from the appropriate domain.) The Constraint Lingo specification is translated into one of several back-end reasoners. The solution is then translated back into a table. Notice that this table does not have the same rows and columns as the table discussed in (Wos et al. 1984, §3.2.1).

The Constraint Lingo solution[10] to the Jobs Puzzle uses `person`, `gender`, and `jobs` as the three domains. The Constraint Lingo constructs used in this solution and their meanings are:

`CLASS` *`name`*`:` *`member...`*
declares the name of a domain and its elements, each of which must appear in one and only one row of the table.

`PARTITION` *`name`*`:` *`pmember...`*
declares the name of a domain and its elements. Some elements may occur multiple times in the column, and not all need appear.

`POWERCLASS (`*`i`*`,`*`j`*`)` *`name`*`:` *`member...`*
declares the name of a domain and its elements. Each entry in the column must contain a set of at least i and at most j of the elements, and no two such sets may be the same.

`AGREE` *`pmember`*`:` *`domainmember...`*
Every row in which one of the listed *`domainmember`*s appears must contain the given partition *`pmember`* in its appropriate column.

`USED` *`member`* `<=` *`j`*
The given *`member`* must appear in at least 1 and at most *`j`* rows of its column.

`CONFLICT` *`member...`*
No row may contain more than one of the listed *`member`*s.

The Constraint Lingo solution follows:

1. *jp: There are four people: Roberta, Thelma, Steve, and Pete.*

```
CLASS person: roberta thelma steve
              pete
```

inf: "if the four names did not clearly imply the sex of the people, [the puzzle] would be impossible to solve." [p. 56]

```
PARTITION gender: male female
AGREE male: steve pete
AGREE female: roberta thelma
```

2. *jp: Among [the people], they hold eight different jobs.*
3. *jp: Each holds exactly two jobs.*
4. *jp: The jobs are: chef, guard, nurse, telephone operator, police officer (gender not implied), teacher, actor, and boxer.*

```
POWERCLASS jobs(2,2): chef guard
   nurse operator police teacher
   actor boxer
```

inf: "No job is held by more than one person." [p. 56]

```
USED chef <= 1
USED guard <= 1
USED nurse <= 1
USED operator <= 1
USED police <= 1
USED teacher <= 1
USED actor <= 1
USED boxer <= 1
```

5. *jp: The job of nurse is held by a male.*

```
AGREE male: nurse
```

inf: "everyday language distinguishes [actors and actresses] based on sex." [p. 56]

```
AGREE male: actor
```

6. *jp: The husband of the chef is the telephone operator.*
inf: "the implicit fact that husbands are male" [p. 57]
inf: since the chef has a husband, she must be female. [p. 57]

```
AGREE female: chef
AGREE male: operator
```

7. *jp: Roberta is not a boxer.*

```
CONFLICT roberta boxer
```

8. *jp: Pete has no education past the ninth grade.*
inf: "the jobs of nurse, police officer, and teacher each require more than a ninth-grade education." [p. 57]

```
CONFLICT pete teacher
CONFLICT pete police
CONFLICT pete nurse
```

9. *jp: Roberta, the chef, and the police officer went golfing together.*
inf: "Thus, we know that Roberta is neither the chef nor the police officer." [p. 57]
inf: "Since they went golfing together, the chef and the police officer are not the same person." [p. 57]

```
CONFLICT roberta chef police
```

jp: Question: Who holds which jobs?
The solution, derived in this case by Smodels (Niemelä and Simons 2000), is

[9] Not included in the published version due to space constraints.

[10] provided by Raphael Finkel [personal communication], with some modifications suggested by me.

```
# gender jobs           person
# ====== ============== =======
# male   actor;operator pete
# female guard;teacher  roberta
# male   nurse;police   steve
# female chef;boxer     thelma
```

Discussion of the Constraint Lingo Solution Constraint Lingo was designed for problems like the Jobs Puzzle, so its solution is rather simple and straightforward. The most tedious aspect is the eight `USED` declarations. The cleverness required to specify that "*Each person holds at most two jobs*" and "*Each job is held by at most one person*" in the TPTP solution is avoided by use of the built-in cardinality constraints of `POWERCLASS` and `USED`. The biggest problem with the Constraint Lingo solution is that it does not adhere very well to the English statements of the puzzle. Constraint Lingo was not designed to be a language for formalizing commonsense reasoning in general, and so the predicates used in the English statements are absent from the Constraint Lingo declarations, instead, they all specify the requirements and constraints of a single, unnamed, relation.