TLP **16** (5–6): 703–720, 2016. © Cambridge University Press 2016
doi:10.1017/S1471068416000326

Paraconsistency and word puzzles

TIANTIAN GAO, PAUL FODOR and MICHAEL KIFER
Stony Brook University, New York, USA
(*e-mail:* {tiagao, pfodor, kifer}@cs.stonybrook.edu)

submitted 6 May 2016; revised 8 July 2016; accepted 22 August 2016

Abstract

Word puzzles and the problem of their representations in logic languages have received considerable attention in the last decade (Ponnuru *et al.* 2004; Shapiro 2011; Baral and Dzifcak 2012; Schwitter 2013). Of special interest is the problem of generating such representations directly from natural language (NL) or controlled natural language (CNL). An interesting variation of this problem, and to the best of our knowledge, scarcely explored variation in this context, is when the input information is inconsistent. In such situations, the existing encodings of word puzzles produce inconsistent representations and break down. In this paper, we bring the well-known type of paraconsistent logics, called *Annotated Predicate Calculus* (APC) (Kifer and Lozinskii 1992), to bear on the problem. We introduce a new kind of non-monotonic semantics for APC, called *consistency preferred stable models* and argue that it makes APC into a suitable platform for dealing with inconsistency in word puzzles and, more generally, in NL sentences. We also devise a number of general principles to help the user choose among the different representations of NL sentences, which might seem equivalent but, in fact, behave differently when inconsistent information is taken into account. These principles can be incorporated into existing CNL translators, such as Attempto Controlled English (ACE) (Fuchs *et al.* 2008) and PENG Light (White and Schwitter 2009). Finally, we show that APC with the consistency preferred stable model semantics can be equivalently embedded in ASP with preferences over stable models, and we use this embedding to implement this version of APC in Clingo (Gebser *et al.* 2011) and its Asprin add-on (Brewka *et al.* 2015).

KEYWORDS: Paraconsistency, Word Puzzles, Annotated Predicate Calculus, Controlled Natural Language

1 Introduction

The problem of logical representation for word puzzles has recently received considerable attention (Ponnuru *et al.* 2004; Shapiro 2011; Baral and Dzifcak 2012; Schwitter 2013). In all of these studies, however, the input information is assumed to be consistent and the proposed logical representations break on inconsistent input. The present paper proposes an approach that works in the presence of inconsistency and not just for word puzzles.

At first sight, one might think that the mere use of a paraconsistent logic such as Belanp's four valued logic (Belnap Jr 1977) or Annotated Logic Programming (Blair and Subrahmanian 1989; Kifer and Subrahmanian 1992) is all what is needed to

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

address the problem, but it is not so. We do start with a well-known paraconsistent logic, called *Annotated Predicate Calculus* (APC) (Kifer and Lozinskii 1992), which is related to the aforementioned Annotated Logic Programs, but this is not enough: a number of issues arise in the presence of paraconsistency and different translations might seem equivalent but behave differently when inconsistent information is taken into account. As it turns out, several factors can affect the choice of the "right" logical representation for many NL sentences, especially for implications. We formalize several principles to guide the translation of NL sentences into APC, principles that can be incorporated into existing controlled language translators, such as Attempto Controlled English (ACE) (Fuchs *et al.* 2008) and PENG Light (White and Schwitter 2009). We illustrate these issues with the classical Jobs Puzzle (Wos *et al.* 1984) and show how inconsistent information affects the conclusions.

To address the above problems formally, we introduce a new kind of non-monotonic semantics for APC, which is based on *consistency-preferred stable models* and is inspired by the concept of the most epistemically-consistent models of (Kifer and Lozinskii 1992). We argue that this new semantics makes APC into a good platform for dealing with inconsistency in word puzzles and, more generally, for translating natural language sentences into logic.

Finally, we show that the consistency-preferred stable models of APC can be computed using answer-set programming (ASP) systems that support preferences over stable models, such as Clingo (Gebser *et al.* 2011) with the Asprin add-on (Brewka *et al.* 2015).

This paper is organized as follows. Section 2 provides background material on APC. In Section 3 we consider the logic programming subset of APC and define preferential stable models for it. In Section 4, we show that the logic programming subset of APC (under the consistency-preferred stable model semantics) can be encoded in ASP in semantically-preserving way. In Section 5, we discuss variations of Jobs Puzzle (Wos *et al.* 1984) when various kinds of inconsistency are injected into the formulation of the puzzle. Section 6 explains that logical encoding of common knowledge in the presence of inconsistency needs to take into account a number of considerations that are not present when inconsistency is not an issue. We organize those considerations into several different principles and illustrate their impact. Section 8 concludes the paper. Finally, (Gao *et al.* 2016, Appendix A) contains the full encoding of Jobs Puzzle in APC under the consistency-preferred semantics. This appendix also includes variations that inject various kinds of inconsistency into the puzzle, and the derived conclusions are discussed. (Gao *et al.* 2016, Appendices B and C) contain similar analyses of other well-known puzzles: Zebra Puzzle[1] and Marathon Puzzle (C. Guéret and Sevaux 2000). Ready-to-run encodings of these programs in Clingo/Asprin can be found at `https://bitbucket.org/tiantiangao/apc_lp`.

[1] `https://en.wikipedia.org/wiki/Zebra_Puzzle`

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

2 Annotated predicate calculus: Background and extensions

To make this paper self-contained, this section provides the necessary background on APC. At the end of the section, we define new semantic concepts for APC, which will be employed in later sections.

The *alphabet* of APC consists of countably-infinite sets of: *variables* $\mathscr{V}$, *function symbols* $\mathscr{F}$ (each symbol having an arity; *constants* are viewed as 0-ary function symbols), *predicate symbols* $\mathscr{P}$, *truth annotations*, quantifiers, and logical connectives. In (Kifer and Lozinskii 1992), truth annotations could come from an arbitrary upper semilattice (called "belief semilattice" there), but here we will use only $\bot$ (unknown), **f** (false), **t** (true) and $\top$ (contradiction or inconsistency), which are partially ordered as follows: $\bot \leqslant \mathbf{f} \leqslant \top$ and $\bot \leqslant \mathbf{t} \leqslant \top$. *Terms* in APC are constructed exactly as in predicate calculus: from constants, variables and function symbols. A *ground term* is one that has no variables.

Definition 1 (*Atomic formulas (Kifer and Lozinskii 1992)*)
A *predicate term* has the form $p(t_1, t_2, \ldots, t_n)$, where p is an n-ary predicate symbol and $t_1, t_2, \ldots, t_n$ are terms. An APC *atomic formula* (or an APC predicate) has the form $p(t_1, t_2, \ldots, t_n) : \mathbf{s}$, where $p(t_1, t_2, \ldots, t_n)$ is a predicate term and **s** is annotation indicating the degree of belief (or truth) in the predicate term. A *ground atomic formula* is an atomic formula that has no variables.

We call an atomic formula of the form $p : \mathbf{s}$ a **t**-*predicate* (resp., an **f**-, $\top$-, or $\bot$-predicate) if **s** is **t** (resp., **f**-, $\top$-, or $\bot$).

APC includes the usual universal and existential quantifiers, the connectives, $\wedge$ and $\vee$, and there are two negation and two implication connectives: the *ontological negation* $\neg$ and *ontological implication* $\leftarrow$, plus the *epistemic negation* $\sim$ and *epistemic implication* $\lessdot$. As will be seen later, the distinction between the ontological and the epistemic connectives is useful because they behave differently in the presence of inconsistency.

Definition 2 (*APC well-formed formulas (Kifer and Lozinskii 1992)*)
An APC *well-formed formula* is defined inductively as follows:

- an atomic formula $p(t_1, t_2, \ldots, t_n) : \mathbf{s}$
- if ϕ and ψ are well-formed formulas, then so are $\sim \phi$, $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \leftarrow \psi$, and $\phi \lessdot \psi$.
- if ϕ is a formula and X is a variable, then $(\forall X \phi)$ and $(\exists X \phi)$ are formulas.

An APC *literal* is either a predicate $p : \mathbf{s}$ or an ontologically negated predicate $\neg p : \mathbf{s}$. An *epistemic literal* is either a predicate $p : \mathbf{s}$ or an epistemically negated predicate $\sim p : \mathbf{s}$.

In (Kifer and Lozinskii 1992), the semantics was defined with respect to general models, but here we will be dealing with logic programs and the Herbrand semantics will be more handy.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

Definition 3 (*APC Herbrand universe, base, and interpretations*)
The *Herbrand universe* $\mathscr{U}$ for APC is the set of all ground terms. The *Herbrand base* $\mathscr{B}$ for APC is the set of all ground APC atomic formulas. A *Herbrand interpretation* I for APC is a non-empty subset of the Herbrand base that is *closed* with respect to the following operations:

- if $p : \mathbf{s} \in I$, then also $p : \mathbf{s}' \in I$ for all $\mathbf{s}' \leqslant \mathbf{s}$; and
- if $p : \mathbf{s}_1$, $p : \mathbf{s}_2 \in I$, and $\mathbf{s} = lub_{\leqslant}(\mathbf{s}_1, \mathbf{s}_2)$ then $p : \mathbf{s} \in I$.

The annotations used in APC form a lattice (in our case a 4-element lattice) with the order $\leqslant$ and with $lub_{\leqslant}$ used as the least upper bound operator of that lattice. We will also use $\mathscr{B}_{\top}$ to denote the subset of all $\top$-*predicates* in $\mathscr{B}$.

As usual, a *variable assignment* is a mapping $v : \mathscr{V} \rightarrow \mathscr{U}$ that takes a variable and returns a ground term. This mapping is extended to terms as follows: $v(f(t_1, \ldots, t_n)) = f(v(t_1), \ldots, v(t_n))$. We will disregard variable assignments for formulas with no free variables (called *sentences*) since they do not affect ground formulas.

Definition 4 (*APC Herbrand Models*)
Let I be an APC Herbrand interpretation and v be a variable assignment. For an atomic formula $p(t_1, t_2, \ldots, t_n) : \mathbf{s}$, we write $I \models_v p(t_1, t_2, \ldots, t_n) : \mathbf{s}$ if and only if $p(v(t_1), v(t_2), \ldots, v(t_n)) : \mathbf{s} \in I$. For well-formed formulas ϕ and ψ, we write:

- $I \models_v \phi \wedge \psi$ if and only if $I \models_v \phi$ and $I \models_v \psi$;
- $I \models_v \phi \vee \psi$ if and only if $I \models_v \phi$ or $I \models_v \psi$;
- $I \models_v \neg\phi$ if and only if not $I \models_v \phi$;
- $I \models_v (\forall X)\phi$ if and only if $I \models_{v'} \phi$, for every assignment v' that differs from v only in its X-value;
- $I \models_v (\exists X)\phi$ if and only if $I \models_{v'} \phi$, for some v' that differs from v only in its X-value;
- $I \models_v \psi \leftarrow \phi$ if and only if $I \models_v \neg\phi \vee \psi$;
- $I \models_v \sim p : \mathbf{s}$ if and only if $I \models_v p :\sim \mathbf{s}$, where $\sim \mathbf{t} = \mathbf{f}$, $\sim \mathbf{f} = \mathbf{t}$, $\sim \top = \top$ and $\sim \bot = \bot$;

We also define: $\sim \neg\phi \equiv \neg \sim \phi$, $\sim (\phi \wedge \psi) \equiv \sim \phi \vee \sim \psi$, $\sim (\phi \vee \psi) \equiv \sim \phi \wedge \sim \psi$, $\sim \forall X\phi \equiv \exists X \sim \phi$, $\sim \exists X\phi \equiv \forall X \sim \phi$, and $\psi \lessdot\!\!\sim \phi \equiv \sim \phi \vee \psi$.

A formula ϕ is *satisfied* by I if and only if $I \models_v \phi$ for every valuation v. In this case we write simply $I \models \phi$. I is a *model* of a set of formulas P if and only if every formula $\phi \in P$ is satisfied in I. A set of formulas P logically entails a formula ψ, denoted $P \models \psi$, if and only if every model of P is also a model of ψ.

APC has two types of logical entailment: *ontological* and *epistemic*. Ontological entailment is the entailment $\models$, which we have just defined. Before defining the *epistemic* entailment, we motivate it with a number of examples. To avoid clutter, in all examples we will only show the highest annotation for each APC predicate. For instance, if a model contains $p : \top$, then we will not show $p : \mathbf{t}$, $p : \mathbf{f}$, or $p :\bot$.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

Example 1
Consider the following set of APC formulas $P = \{q : \mathbf{t} \leftarrow p : \mathbf{t},\ \ p : \mathbf{t}\}$. It has four models: $m_1 = \{p : \mathbf{t},\ \ q : \mathbf{t}\}$, $m_2 = \{p : \top,\ \ q : \mathbf{t}\}$, $m_3 = \{p : \mathbf{t},\ \ q : \top\}$ and $m_4 = \{p : \top,\ \ q : \top\}$. Thus, $P \models q : \mathbf{t}$ holds (since $q : \mathbf{t}$ occurs in every model of P).

Example 2
The APC set of formulas $P = \{q : \mathbf{t} \leftarrow p : \mathbf{t},\ \ p : \top\}$ has two models: $m_1 = \{p : \top,\ \ q : \mathbf{t}\}$ and $m_2 = \{p : \top,\ \ q : \top\}$. Therefore, $P \models q : \mathbf{t}$ holds.

Example 3
This set of formulas $P = \{q : \mathbf{t} \nleftarrow p : \mathbf{t},\ \ p : \mathbf{t}\}$ is similar to that in Example 1 except that it uses epistemic implication instead of the ontological one. One of the models of that set is $m = \{p : \top,\ \ q : \bot\}$ and therefore $P \not\models q : \mathbf{t}$.

Examples 1 and 2 show that *ontological implication* has the modus ponens property, but it may be too strong, as it allows one to draw conclusions from inconsistent information. *Epistemic implication* of Example 3, on the other hand, is too cautious and does *not* have the modus ponens property. However, epistemic implication *does* have the modus ponens property *and* it blocks drawing conclusions from inconsistency under the *epistemic entailment*, defined next.

Definition 5 (*Most e-consistent models (Kifer and Lozinskii 1992)*)
A Herbrand interpretation I_1 is *more* (or equally) *e-consistent* than another interpretation I_2 (denoted $I_1 \leqslant_\top I_2$) if and only if $I_1 \models p : \top$ implies $I_2 \models p : \top$ for every ground predicate term p.

A model I of a set of formulas P is a *most e-consistent model*, if there is no other model of P that is strictly more e-consistent than I.

A program P *epistemically* entails a formula ψ, denoted $P \mid\!\approx \psi$, if and only if every most e-consistent model of P is also a model of ψ.

Going back to Example 3, it has only one most e-consistent model $m = \{p : \mathbf{t}, q : \mathbf{t}\}$, so $P \mid\!\approx q : \mathbf{t}$ holds. The next example shows that $\nleftarrow$ does *not* propagate inconsistency to conclusions.

Example 4
Let $P = \{q : \mathbf{t} \nleftarrow p : \mathbf{t},\ \ p : \top\}$. Observe that P has a most e-consistent model $m = \{p : \top,\ \ q : \bot\}$, in which $q : \mathbf{t}$ does not hold. Therefore, $P \not\mid\!\approx q : \mathbf{t}$ holds.

Next we observe that not all inconsistent information is created equal, as people have different degrees of confidence in different pieces of information. For instance, one normally would have higher confidence in the fact that someone named Robin is a person than in the fact that Robin is a male. Therefore, given a choice, we would hold it less likely that *person*(*robin*) is inconsistent than that *male*(*robin*) is. Likewise, in the following example, given a choice, we are more likely to hold to a belief that Pete is a person than to a belief that he is rich.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

Example 5
Consider the following formulas

$$person(pete) : \mathbf{t}.$$
$$businessman(pete) : \mathbf{t}.$$
$$rich(pete) : \mathbf{f}.$$
$$rich(X) : \mathbf{t} \leftsquigarrow person(X) : \mathbf{t} \ \wedge \ businessman(X) : \mathbf{t}.$$

There are three *most e-consistent models*:

$$m_1 = \{person(pete) : \top, \ businessman(pete) : \mathbf{t}, \ rich(pete) : \mathbf{f}\}$$
$$m_2 = \{person(pete) : \mathbf{t}, \ businessman(pete) : \top, \ rich(pete) : \mathbf{f}\}$$
$$m_3 = \{person(pete) : \mathbf{t}, \ businessman(pete) : \mathbf{t}, \ rich(pete) : \top\}$$

Based on the aforesaid confidence considerations, we are more likely to believe that Pete is a person than that he is a businessman or rich. Therefore, we are likely to think that the models m_2 and m_3 are better descriptions of the real world than m_1.

In this paper, we capture the above intuition by extending the notion of most e-consistent models with additional preferences over models.

Definition 6 (*Consistency-preference relation and consistency-preferred models*)
A *consistency preference* S over interpretations, where S is a set of ground $\top$-predicates in APC, is defined as follows:

- An interpretation I_1 is *consistency-preferred* over I_2 with respect to S, denoted $I_1 <_S I_2$, if and only if $S \cap I_1 \subset S \cap I_2$.
- Interpretation I_1 and I_2 are *consistency-equal* with respect to S, denoted $I_1 =_S I_2$, if and only if $S \cap I_1 = S \cap I_2$.

A *consistency-preference relation* $<_{\mathscr{S}}$, where $\mathscr{S} = (S_1, S_2, \ldots, S_n)$ is a sequence of sets of ground $\top$-predicates, is defined as a lexicographic order composed out of the sequence of consistency preferences $S_1, S_2, \ldots, S_n$. Namely, $I_1 <_{\mathscr{S}} I_2$ if and only iff there is $1 \leqslant i \leqslant n$ such that $(\bigwedge_{1 \leqslant j < i}(I_1 =_{S_j} I_2))$ and $I_1 <_{S_i} I_2$.

A model I of a set of formulas P is called (most) *consistency-preferred* with respect to $<_{\mathscr{S}}$ if P has no other model J such that $J <_{\mathscr{S}} I$.

We will always assume that $S_n = \mathscr{B}_\top$ — the set of all ground $\top$-predicates and, therefore, any most consistency-preferred model is also a most e-consistent one.

We use the notation $|\!\approx_{\mathscr{S}}$ to denote *epistemic entailment* with respect to most consistency-preferred models. A program P *epistemically entails* a formula ψ with respect to a consistency-preference relation $<_{\mathscr{S}}$, denoted $P |\!\approx_{\mathscr{S}} \psi$, if and only if every most consistency-preferred model of P is also a model of ψ.

3 Logic programming subset of APC and its stable models semantics

In this section, we define the logic programming subset of APC, denoted APC_{LP}, and give it a new kind of semantics based on *consistency-preferred stable models*.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

Definition 7
An APC_{LP} *program* consists of rules of the form:

$$l_0 \vee \cdots \vee l_m \leftarrow l_{m+1} \wedge \cdots \wedge l_n \wedge \neg\, l_{n+1} \wedge \cdots \wedge \neg\, l_k.$$

where each l_i is an epistemic literal. Variables are assumed to be implicitly universally quantified. An APC_{LP} *formula* is either a singleton epistemic literal, or a conjunction of epistemic literals, or a disjunction of them.

The formula $l_0 \vee \cdots \vee l_m$ is called the *head* of the rule, and $l_{m+1} \wedge \cdots \wedge l_n \wedge \neg\, l_{n+1} \wedge \cdots \wedge \neg\, l_k$ is the *body* of that rule.

Recall from Section 2 that epistemic negation can be pushed inside and eliminated via this law: $\sim p : \alpha \equiv p :\sim \alpha$, where $\sim \mathbf{t} = \mathbf{f}$, $\sim \mathbf{f} = \mathbf{t}$, $\sim \top = \top$, and $\sim\bot = \bot$ so, for brevity, we assume that all APC_{LP} programs are transformed in this way and the epistemic negation is eliminated.

When the rule body is empty, the ontological implication symbol $\leftarrow$ is usually omitted and the rule becomes a disjunction. Such a disjunction can also be represented as an epistemic implication and sometimes this representation may be closer to a normal English sentence. For instance, the sentence, "If a person is a businessman then that person is rich," can be represented as an epistemic implication: $rich(X) : \mathbf{t} \;\lessapprox\; person(X) : \mathbf{t} \;\wedge\; businessman(X) : \mathbf{t}$, which is easier to read than the equivalent disjunction $rich(X) : \mathbf{t} \vee person(X) : \mathbf{f} \vee businessman(X) : \mathbf{f}$.

The notion of stable models for APC_{LP} carries over from standard answer set programming (ASP) with very few changes.

Definition 8 (*The Gelfond-Lifschitz reduct for* APC_{LP})
Let P be an APC_{LP} program and M be a Herbrand interpretation. The reduct of P w.r.t. M, denoted $\frac{P}{M}$, is a program free from ontological negation obtained by

1. removing rules with $\neg p : \mathbf{s}$ in the body, where $M \models p : \mathbf{s}$; and
2. removing literals $\neg p : \mathbf{s}$ from all remaining rules.

Definition 9 (*Stable models for* APC_{LP})
A Herbrand interpretation M is a *stable model* of an APC_{LP} program P if M is a minimal model of $\frac{P}{M}$. Here, minimality is with respect to set inclusion.

Definition 10 (*Consistency-preferred stable models for* APC_{LP})
Let $<_{\mathscr{S}}$ be a consistency-preference relation of Definition 6, where $\mathscr{S} = (S_1, S_2, \ldots, S_n)$ is a sequence of sets of ground $\top$-predicates. An APC_{LP} interpretation M is a (most) *consistency-preferred stable model* of an APC_{LP} program P if and only if:

– M is a stable model of P, and
– M is a most consistency-preferred model with respect to $<_{\mathscr{S}}$.

4 Embedding APC_{LP} into ASP

We now show that APC_{LP} can be isomorphically embedded in ASP extended with a model preference framework, such as the Clingo system (Gebser *et al.* 2011) with its Asprin extension (Brewka *et al.* 2015). We then prove the correctness of this

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

embedding, i.e., that it is one-to-one and preserves the semantics. Next, we define the subset of ASP onto which APC_{LP} maps.

Definition 11
ASP_{truth} is a subset of ASP programs where the only predicate is `truth/2`, which is used to reify the APC predicate terms and associate them with truth values. That is, these atoms have the form $truth(p, \mathbf{s})$, where the first argument is the reification of an APC predicate term and the second argument is one of these truth annotations: `t`, `f`, `top`, or `bottom`.

An ASP_{truth} *program* consists of a set of rules of the form:

$$t_0 \vee \cdots \vee t_m \leftarrow t_{m+1} \wedge \cdots \wedge t_n \wedge not\ t_{n+1} \wedge \cdots \wedge not\ t_k.$$

where the t_i's are `truth/2`-predicates.

An ASP_{truth} *formula* is either a singleton `truth/2`-predicate, a conjunction of such predicates, or a disjunction of them.

Definition 12
The embedding of an APC_{LP} program in ASP_{truth}, denoted Ξ_{asp}, is defined recursively as follows (where tv_{asp} is the truth value mapping):

- $tv_{asp}(\mathbf{t}) = \texttt{t}$
- $tv_{asp}(\mathbf{f}) = \texttt{f}$
- $tv_{asp}(\top) = \texttt{top}$
- $tv_{asp}(\bot) = \texttt{bottom}$
- $\Xi_{asp}(p : \mathbf{s}) = \texttt{truth(p,}tv_{asp}(\mathbf{s})\texttt{)}$
- $\Xi_{asp}(\neg L) = \texttt{not}\ \Xi_{asp}(L)$, where L is an APC predicate
- $\Xi_{asp}(L \vee \phi) = \Xi_{asp}(L) \vee \Xi_{asp}(\phi)$, where L is an APC predicate and ϕ is a disjunction of APC predicates
- $\Xi_{asp}(L \wedge \phi) = \Xi_{asp}(L) \wedge \Xi_{asp}(\phi)$, where L is an APC literal and ϕ is a conjunction of APC literals
- $\Xi_{asp}(head \leftarrow body) = \Xi_{asp}(head) \leftarrow \Xi_{asp}(body)$, where *head* (resp., *body*) denotes the head (resp., the body) of a rule.

The embedding Ξ_{asp} also applies to APC Herbrand interpretations: each APC Herbrand interpretation (which is a set of APC atoms of the form $p : \mathbf{s}$) is mapped to a set of ASP_{truth} atoms (of the form $\texttt{truth(p,}tv_{asp}(\mathbf{s})\texttt{)}$).

We require that each ASP_{truth} program includes the following background axioms to match the semantics of APC:

```
truth(X,top) :- truth(X,t),truth(X,f).
truth(X,t) :- truth(X,top).
truth(X,f) :- truth(X,top).
truth(X,bottom).
```

Lemma 1
The embedding $\Xi asp : APC_{LP} \rightarrow ASP_{truth}$ is a one-to-one correspondence.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

Proof
As mentioned, we can limit our attention to $\sim$-free programs. First, it is obvious that Ξ_{asp} is injective on APC literals. Injectivity on APC conjunctions and disjunctions can be shown by a straightforward induction on the number of conjuncts and disjuncts. Surjectivity follows similarly because it is straightforward to define the inverse of Ξ_{asp} by reversing the equations of Definition 12. □

Next, we show the above APC-to-ASP embedding preserves models, Gelfond-Lifshitz reduct, stable models, and also consistency preference relations.

Lemma 2
The models of any ASP_{truth} program are closed with respect to $lub_{\leqslant}$ and downward-closed with respect to the $\leqslant$-ordering. Also, I is a model of an APC_{LP} program P if and only if $\Xi_{asp}(I)$ is a model of $\Xi_{asp}(P)$.

Proof
Recall that every APC_{truth} is required to have the four rules listed right after Definition 12. These rules obviously enforce the requisite closures. The second part of the lemma follows directly from the definitions. □

Lemma 3
Ξ_{asp} preserves the Gelfond-Lifshitz reduct: $\Xi_{asp}(\frac{P}{I}) = \frac{\Xi asp(P)}{\Xi_{asp}(I)}$.

Proof
For every predicate $p : \mathbf{s} \in P$, we have $I \models p : \mathbf{s}$ if and only if $\Xi_{asp}(I) \models truth(p, \mathbf{s})$, by Lemma 2. By the same lemma, if $r \in P$ then $I \models p : \mathbf{s}$ where $\neg p : \mathbf{s} \in body(r)$ if and only if $\Xi_{asp}(I) \models truth(p, \mathbf{s})$, where $not\ truth(p, \mathbf{s}) \in body(\Xi_{asp}(r))$. As a result, rule r gets eliminated by Gelfond-Lifschitz reduction if and only if $\Xi_{asp}(r)$ is eliminated and a negative literal in the body of r gets dropped if and only if its image in $\Xi_{asp}(r)$ gets dropped. □

Lemma 4
Let I be an APC Herbrand interpretation. J is an APC Herbrand model of $\frac{P}{I}$ if and only if $\Xi_{asp}(J)$ is a model of $\Xi_{asp}(\frac{P}{I})$.

Proof
If $r \in P$ is a rule then $J \models head(r)$ if and only if $\Xi_{asp}(J) \models head(\Xi_{asp}(r))$ and $J \not\models body(r)$ if and only if $\Xi_{asp}(J) \not\models body(\Xi_{asp}(r))$. Thus, $J \models r$ if and only if $\Xi_{asp}(J) \models \Xi_{asp}(r)$. □

Lemma 5
Let I_1 and I_2 be APC *Herbrand interpretations*. $I_1 \subseteq I_2$ if and only if $\Xi_{asp}(I_1) \subseteq \Xi_{asp}(I_2)$.

Proof
Follows directly from the definition of Ξ_{apc} and its inverse. □

Theorem 6
M is a stable model of an APC_{LP} program P if and only if $\Xi_{asp}(M)$ is a stable model of $\Xi_{asp}(P)$.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

Proof
By Lemma 4, J is a model of $\frac{P}{M}$ if and only if $\Xi_{asp}(J)$ is a model of $\frac{\Xi_{asp}(P)}{\Xi_{asp}(M)}$. Thus, the set of models for $\frac{P}{M}$ is in a one-one correspondence with the set of models for $\frac{\Xi_{asp}(P)}{\Xi_{asp}(M)}$. By Lemma 5, this correspondence preserves set-inclusion, so the set of *minimal* models of $\frac{P}{M}$ stands in one-one correspondence with respect to Ξ_{asp} with the set of minimal models of $\frac{\Xi_{asp}(P)}{\Xi_{asp}(M)}$. □

A consistency preference relation $<_{\mathscr{S}}$, where $\mathscr{S} = (S_1, S_2, \ldots, S_n)$, is translated into the following Asprin (Brewka *et al.* 2015) *lexico* preference relation $\mathscr{A}$ along with several `subset` preferences relations, each corresponding to one of the $<_{S_i}$ that are part of $<_{\mathscr{S}}$ (see Definition 6).

`#preference(` $\mathscr{A}$ `, lexico){ 1::name(` s_1 `); 2::name(` s_2 `); ...; n::name(` s_n `)}.`

`#preference(` s_1 `, subset){` the list of elements in $\Xi_{asp}(S_1)$ `}.`

...

`#preference(` s_n `, subset){` the list of elements in $\Xi_{asp}(S_n)$ `}.`

Lemma 7
Let I_1 and I_2 be APC *Herbrand interpretations*, $\mathscr{S} = (S_1, S_2, \ldots, S_n)$ be a consistency preference relation and $\mathscr{A}$ be its corresponding Asprin preference relation. $I_1 <_{\mathscr{S}} I_2$ if and only if $\Xi_{asp}(I_1)$ is preferred over $\Xi_{asp}(I_2)$ with respect to $\mathscr{A}$.

Proof
The definition in the Asprin manual of the Asprin *lexico* and *subset* preference relations, as applied to our preference statements $\mathscr{A}$ given just prior to Lemma 7, is just a paraphrase of the lexicographical consistency-preference relation $\mathscr{S}$ in Definition 6. The lemma now follows from the obvious fact that Ξ_{asp} maps $\top$-literals of ASP onto the `top`-literals of ASP_{truth}, which have the form $truth(p, \texttt{top})$. □

Theorem 8
M is a *most consistency preferred stable model* of an APC_{LP} program P with respect to a consistency preference relation $<_{\mathscr{S}}$ (where $\mathscr{S} = (S_1, S_2, \ldots, S_n)$) if and only if $\Xi_{asp}(M)$ is a preferred model of $\Xi_{asp}(P)$ with respect to the corresponding Asprin preference relation $\mathscr{A}$.

Proof
By Lemma 4, J is a model of $\frac{P}{M}$ if and only if $\Xi_{asp}(J)$ is a model of $\frac{\Xi_{asp}(P)}{\Xi_{asp}(M)}$. Since, by Lemma 7, Ξ_{asp} maps the preference relation $\mathscr{S}$ over the APC models into the preference relation $\mathscr{A}$ over the ASP models, the result follows. □

5 Jobs puzzle and inconsistency

Jobs Puzzle (Wos *et al.* 1984) is a classical logical puzzle that became a benchmark of sorts for many automatic theorem provers (Shapiro 2011; Schwitter 2013); it is also included in TPTP.[2] The usual description of Jobs Puzzle does not include

[2] Thousands of Problems for Theorem Provers (`http://www.cs.miami.edu/ tptp/`).

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

implicit knowledge, like the facts that a person is either a male or a female (but not both), the husband of a person must be unique, etc., so we add this knowledge explicitly, like (Schwitter 2013). We also changed the name Steve to Robin in order to better illustrate one form of inconsistency.

1. There are four people: Roberta, Thelma, Robin 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.

In sum there are four people and eight jobs and to solve the puzzle one must figure out who holds which jobs. The solution is that Thelma is a chef and a boxer (and is married to Pete). Pete is a telephone operator and an actor. Roberta is a teacher and a guard. Finally, Robin is a police officer and a nurse.

However, if we inject inconsistency into the puzzle, current logical approaches fail because they are based on logics that do not tolerate inconsistency. Consider the following examples.

Example 6
Let us add to the puzzle that "Thelma is an actor." Given that the original puzzle implies that Thelma is *not* an actor (she was a chef and a boxer), this addition causes inconsistencies. A first-order encoding of the puzzle (as, say, in TPTP) or an ASP-based one in (Schwitter 2013) will not find any models. In contrast, an encoding in APC_{LP} can isolate inconsistent information. There are two possibilities: one where Thelma is an actor and the other where Thelma is a female. If we add background knowledge that Thelma is a female's name, it is less likely that Thelma's gender will be inconsistent, so the only consistency-preferred model will have one inconsistent conclusion that Thelma is an actor, but all other true facts will remain consistent.

Example 7
Consider adding the sentences "Robin is a male name" and "Robin is a female name," which will imply that Robin is both a male and female. The first-order and ASP-based encodings will, again, find no models, while an APC_{LP}-based encoding will localize inconsistency to just $male(robin) : \top$ and $female(robin) : \top$.

Example 8
Consider adding the sentence "Robin is Thelma's husband." Since the original job puzzle implies that Pete is Thelma's husband, this will cause inconsistency. If we add the background knowledge that husband is unique, again, the encoding of this modified puzzle in ACP_{LP} will localize inconsistency to just the aforesaid husband-facts.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

6 Knowledge representation principles for inconsistency

Mere encoding of Jobs Puzzle in APC_{LP} is not enough because it is not unique: when inconsistency is taken into account, more information needs to be provided to obtain the encodings that match user intent. The main problem is that, if inconsistency is allowed, the number of possible worlds can grow to many hundreds even in relatively simple scenarios like Jobs Puzzle, and this practically annuls the benefits of the switch to a paraconsistent logic. We have already seen small examples of such scenarios at the end of Section 2, which motivated our notion of consistency preference, but there are more. We organize these scenarios around six main principles.

Principle 1: Contrapositive inference

Like in classical logic, contrapositive inference may be useful for knowledge representation. Consider the following sentences:

> If someone is a nurse, then that someone is educated.
> Pete is not educated.

We could encode the first sentence as $educated(X) : \mathbf{t} \leftarrow nurse(X) : \mathbf{t}$ or as $educated(X) : \mathbf{t} \lessapprox nurse(X) : \mathbf{t}$. Classically, the above sentences imply that Pete is not a nurse, but the encoding of the first sentence using the ontological implication $\leftarrow$ would not allow for that. If contrapositive inference is required, epistemic implication should be used.

Example 9

Consider $P = \{\text{educated}(X) : \mathbf{t} \lessapprox nurse(X) : \mathbf{t}, \quad \sim \text{educated}(pete) : \mathbf{t}\}$. It has only one most consistency preferred model with respect to $<_{\mathscr{S}}$ (with $\mathscr{S} = (\mathscr{B}_{\top})$), namely $m = \{\text{educated}(pete) : \mathbf{f}, \quad nurse(pete) : \mathbf{f}\}$. Therefore, $P \mid\!\approx_{\mathscr{S}} nurse(pete) : \mathbf{f}$ holds.

The above example uses contrapositive inference, but this is not always desirable. For instance, suppose $P' = \{male(X) : \mathbf{t} \leftarrow nurse(X) : \mathbf{t}, \quad \sim male(robin) : \mathbf{t}\}$. Here we use ontological implication to block contrapositive inference. Observe that P' has a most consistency preferred model with respect to $<_{\mathscr{S}}$, namely $m = \{male(robin) : \mathbf{f}, \quad nurse(robin) : \bot\}$. Therefore, $nurse(robin) : \mathbf{f}$ does not hold, and this is exactly what we want, even if Robin happens to be not a male.[3]

Principle 2: Propagation of inconsistency

As discussed in Example 2, APC gives us a choice of whether to draw conclusions from inconsistent information or not, and it is a useful choice. One way to block such inferences, illustrated in that example, is to use epistemic implication. Another way is to use the ontological implication with the $\mathbf{t} + \neg\top$ pattern in the rule body, e.g.,

$$educated(X) : \mathbf{t} \leftarrow nurse(X) : \mathbf{t} \wedge \neg nurse(X) : \top.$$

Both techniques block inferences from inconsistent information, but the second also blocks inference by contraposition, as discussed in Principle 1. The following examples illustrate the use of both of these methods.

[3] In the USA as opposed to the U.K.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

Example 10
Let $P = \{\text{educated}(X) : \mathbf{t} \lessapprox nurse(X) : \mathbf{t},\ \ nurse(pete) : \top\}$. Observe that there is one most consistency preferred model with respect to $<_{\mathscr{S}}$ (as before, $\mathscr{S} = (\mathscr{B}_{\top})$): $m = \{\text{educated}(pete) : \bot,\ \ nurse(pete) : \top\}$. Therefore, $P \not\models_{\mathscr{S}} \text{educated}(pete) : \mathbf{t}$.

Example 11
Let $P = \{\text{educated}(X) : \mathbf{t} \leftarrow nurse(X) : \mathbf{t} \wedge \neg nurse(X) : \top,\ \ nurse(pete) : \top\}$. As in the previous example, P has a most consistency preferred model $m = \{nurse(pete) : \top,\ \ \text{educated}(pete) : \bot\}$ and so $P \not\models_{\mathscr{S}} \text{educated}(pete) : \mathbf{t}$.

In both of these examples, inconsistency is not propagated through the rules, but Example 10 allows for contrapositive inference, while Example 11 does not. Indeed, suppose that instead of $nurse(pete) : \top$ we had $educated(pete) : \mathbf{f}$. Then, in the first case, $nurse(pete) : \mathbf{f}$ would be derived, while in the second it would not.

Blocking contrapositive inference and non-propagation of inconsistency can be applied selectively to some literals but not the others.

Example 12
Consider the following sentence, "if a person holds a job of nurse then that person is educated". It can be encoded as

$$(educated(X) : \mathbf{t} \ \lessapprox\ nurse(X) : \mathbf{t}) \ \leftarrow\ person(X) : \mathbf{t}.$$

The rule allows propagation of inconsistency through the *person*-predicate but blocks such propagation for the *nurse*-predicate. It also inhibits contrapositive inference of $person(pete) : \mathbf{f}$ if the head of the rule is falsified by the additional facts $nurse(pete) : \mathbf{t}$ and $\text{educated}(pete) : \mathbf{f}$. However, due to the head of the rule, contrapositive inference would be allowed for $nurse(pete) : \mathbf{f}$ if $\text{educated}(pete) : \mathbf{f}$ was given.

Principle 3: Polarity

This principle addresses situations such as the sentence "A person must be either a male or a female, but not both". When inconsistency is possible, we want to say three things: that any person must be either a male and or a female, that these facts cannot be unknown, and that if one of these is inconsistent then the other is too.

Example 13
Let P be:

$$\begin{aligned}
&male(X) : \mathbf{t} \ \vee\ \text{female}(X) : \mathbf{t} \ \leftarrow\ person(X) : \mathbf{t}\\
&male(X) : \mathbf{f} \ \vee\ \text{female}(X) : \mathbf{f} \ \leftarrow\ person(X) : \mathbf{t}\\
&male(X) : \top \ \leftarrow\ person(X) : \mathbf{t} \ \wedge\ \text{female}(X) : \top\\
&\text{female}(X) : \top \ \leftarrow\ person(X) : \mathbf{t} \ \wedge\ male(X) : \top\\
&person(robin) : \mathbf{t}
\end{aligned}$$

Two most consistency preferred models exist, which minimize the inconsistency of $person(robin)$:

$$\begin{aligned}
m_1 &= \{person(robin) : \mathbf{t},\ \ male(robin) : \mathbf{t},\ \ \text{female}(robin) : \mathbf{f}\}, \text{ and}\\
m_2 &= \{person(robin) : \mathbf{t},\ \ male(robin) : \mathbf{f},\ \ \text{female}(robin) : \mathbf{t}\}.
\end{aligned}$$

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

If we add $male(robin) : \top$ (or $female(robin) : \top$) to P, then only one most consistency preferred model remains: $m = \{male(robin) : \top, female(robin) : \top, person(robin) : \mathbf{t}\}$.

Conditional *polarity* (or *polar dependency*) is generally represented as follows

$$p : \mathbf{t} \ ; \ q : \mathbf{t} \leftarrow condition.$$
$$p : \mathbf{f} \ ; \ q : \mathbf{f} \leftarrow condition.$$
$$q : \top \leftarrow condition \wedge p : \top.$$
$$p : \top \leftarrow condition \wedge q : \top.$$

where *condition* is a conjunction of atomic formulas and p, q are polar facts with respect to that condition.

Principle 4: Consistency preference relations

Recall from Example 5 that inconsistent information is not created equal, as people have different degrees of confidence in different pieces of information. For example, we have more confidence that someone whom we barely know is a person compared to the information about this person's marital situation (e.g., whether a husband exists). Therefore, person-facts are more likely to be consistent than marriage-facts and so we need to define consistency preference relations to specify the degrees of confidence. Consistency preference relations were introduced in Definition 6, and we already had numerous examples of its use. In Jobs Puzzle encoding in (Gao *et al.* 2016, Appendix A), we use one, fairly elaborate, consistency preference relation. It first sets person and job information to be of the highest degree of confidence. Then, it prefers consistency of gender information of everybody but Robin. Third, it prefers consistency of the job assignment information. And finally, it minimizes inconsistency in general, for all facts.

Principle 5: Complete knowledge

This principle stipulates that certain information is defined completely, and cannot be unknown ($\bot$). But it can be inconsistent. Moreover, similarly to closed world assumption, negative information is preferred. For instance, if we do not know that someone is someone's husband, we may assume that that person is not. Such conclusions can be specified via a rule like this:

$$husband(X, Y) : \mathbf{f} \ \leftarrow \ person(X) : \mathbf{t} \ \wedge \ person(Y) : \mathbf{t} \ \wedge \ \neg husband(X, Y) : \mathbf{t}$$

Note that, unlike, say ASP, jumping to negative conclusions is not ensured by the stable model semantics of APC and must be given explicitly. But the advantage is that it can be done selectively. More generally, this type of reasoning can be specified as

$$p : \mathbf{f} \ \leftarrow \ condition \ \wedge \ \neg p : \mathbf{t}.$$

if p is known to be a predicate that is defined completely under the *condition*.

Principle 6: Exactly N

This principle captures the encoding of cardinality constraints in the presence of inconsistency. For instance, in Jobs Puzzle, the sentences "Every person holds exactly

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

two jobs" and "Every job is held by exactly one person" are encoded as cardinality constraints:

$$2\ \{hold(X,Y):\mathbf{t}\quad if\quad job(Y):\mathbf{t}\}\ 2\ \leftarrow\ person(X):\mathbf{t}.$$
$$1\ \{hold(X,Y):\mathbf{t}\quad if\quad person(X):\mathbf{t}\}\ 1\ \leftarrow\ job(Y):\mathbf{t}.$$
$$hold(X,Y):\mathbf{f}\ \leftarrow\ person(X):\mathbf{t}\ \wedge\ job(Y):\mathbf{t}\ \wedge\ \neg hold(X,Y):\mathbf{t}.$$

These constraints count both true and inconsistent *hold*-facts, but can be easily modified to count only consistent true facts. Note the role of the last rule, which closes off the information being counted by the constraint. This is necessary because if, say, Pete is concluded to hold *exactly* two jobs (of an actor and a phone operator) then there should be nothing unknown about him holding any other job. Instead, $hold(pete,X):\mathbf{f}\wedge hold(pete,X):\top$ should be true for any other job X.

The general form of the exactly N constraint is:

$$N\ \{L:\mathbf{t}\quad if\quad condition\}\ N\ \leftarrow\ guard.$$
$$L:\mathbf{f}\ \leftarrow\ guard\ \wedge\ condition\ \wedge\ \neg L:\mathbf{t}.$$

As in ASP, such statements can be represented as a number of ground disjunctive rules. The "exactly N" constraints can be generalized to "at least N and at most M" constraints, if we extend the semantics in the direction of (Soininen *et al.* 2001).

7 Comparison with other work

Although a great deal of work is dedicated to paraconsistent logics and logical formalizations for word puzzles separately, we are unaware of any work that applies paraconsistent logics to solving word puzzles that might contain inconsistencies. As we demonstrated, mere encoding of such puzzles in a paraconsistent logic leads to an explosion of possible worlds, which is not helpful.[4] Most paraconsistent logics (Priest *et al.* 2015; J. Y. Beziau 2007; Belnap Jr 1977; da Costa 1974) deal with inconsistency from the philosophical or mathematical point of view and do not discuss knowledge representation. Other paraconsistent logics (Blair and Subrahmanian 1989; Kifer and Subrahmanian 1992) were developed for definite logic programs and cannot be easily applied to solving more complex knowledge representation problems that arise in word puzzles. An interesting question is whether our use of APC is essential, i.e., whether the notions of consistency-preferred models can be adapted to other paraconsistent logics and the relationship with ASP can be established. First, it is clear that such an adaptation is unlikely for proof-theoretic approaches to inconsistency, such as (da Costa 1974). We do not know if such an adaptation is possible for model-theoretic approaches, such as (Belnap Jr 1977).

On the word puzzles front, (Wos *et al.* 1984) used the first-order logic theorem prover OTTER to solve Jobs Puzzle[5] and (Shapiro 2011) represented Jobs Puzzle

[4] Also see (Gao *et al.* 2016, Appendix A) and the ready-to-run examples at `https://bitbucket.org/tiantiangao/apc_lp`.

[5] `http://www.mcs.anl.gov/ wos/mathproblems/jobs.txt`

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

in multiple logical languages: TPTP,[6] Constraint Lingo (Finkel *et al.* 2004) layered on top of the ASP system Smodels (Syrjänen and Niemelä 2001) as the backend, and the SNePS commonsense reasoning system (Shapiro 2000). More recently, (Baral and Dzifcak 2012; Schwitter 2013) represented word puzzles using NL/CNL sentences, and then automatically translate them into ASP. None of these underlying formalisms, FOL, ASP, and SNePS, are equipped to reason in the presence of inconsistency. In contrast, APC_{LP}, combined with the knowledge representation principles developed in Section 6, localizes inconsistency and computes useful possible worlds. In addition, APC_{LP} has mechanisms to control how inconsistency is propagated through inference, it allows one to prioritize inconsistent information, and it provides several other ways to express user's intent (through contraposition, completion of knowledge, etc.).

8 Conclusion

In this paper we discussed the problem of knowledge representation in the presence of inconsistent information with particular focus on representing English sentences using logic, as in word puzzles (Wos *et al.* 1984; Shapiro 2011; Ponnuru *et al.* 2004; Schwitter 2013; Baral and Dzifcak 2012). We have shown that a number of considerations play a role in deciding on a particular encoding, which includes whether or not inconsistency should be propagated through implications, relative degrees of confidence in different pieces of information, and others. We used the well-known Jobs, Zebra and Marathon puzzles (see the appendices in the supplemental material) to illustrate many of the above issues and show how the conclusions change with the introduction of different kinds of inconsistency into the puzzle.

As a technical tool, we started with a paraconsistent logic called *Annotated Predicate Calculus* (Kifer and Lozinskii 1992) and then gave it a special kind of non-monotonic semantics that is based on *consistency-preferred stable models*. We also showed that these models can be computed using ASP systems that support preference relations over stable models, such as Clingo (Gebser *et al.* 2011) with the Asprin extension (Brewka *et al.* 2015).

For future work, we will consider additional puzzles which may suggest new knowledge representation principles. In addition, we will investigate ways to incorporate inconsistency into CNL systems. This will require introduction of background knowledge into these systems and linguistic cues into the grammar.

Supplementary material

For supplementary material for this article, please visit http://dx.doi.org/10.1017/S1471068416000326

[6] `http://www.cs.miami.edu/ tptp/cgi-bin/SeeTPTP?Category=Problems&Domain=PUZ&File=PUZ019-1.p`

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

References

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 *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press, Rome, Italy.

Belnap Jr, N. D. 1977. A useful four-valued logic. In *Modern uses of multiple-valued logic*. Springer, Volume 2, 5–37.

Blair, H. and Subrahmanian, V. 1989. Paraconsistent logic programming. *Theoretical Computer Science 68*, 135–154.

Brewka, G., Delgrande, J. P., Romero, J. and Schaub, T. 2015. asprin: Customizing answer set preferences without a headache. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. AAAI Press, Austin, Texas, 1467–1474.

C. Guéret, C. P. and Sevaux, M. 2000. *Programmation linéaire - 65 problèmes d'optimisation modélisés et résolus avec Visual Xpress*. Eyrolles, France. ISBN : 2-212-09202-4.

da Costa, N. 1974. On the theory of inconsistent formal systems. *Notre Dame J. of Formal Logic 15,* 4 (October), 497–510.

Finkel, R. A., Marek, V. W. and Truszczynski, M. 2004. Constraint lingo: towards high-level constraint programming. *Softw., Pract. Exper. 34,* 15, 1481–1504.

Fuchs, N. E., Kaljurand, K. and Kuhn, T. 2008. Attempto controlled English for knowledge representation. In *Reasoning Web, 4th International Summer School 2008, Venice, Italy, September 7-11, 2008, Tutorial Lectures*, Lecture Notes in Computer Science, vol. 5224. Springer, Venice, Italy, 104–124.

Gao, T., Fodor, P. and Kifer, M. 2016. Paraconsistency and Word Puzzles. *ArXiv e-prints* `http://arxiv.org/abs/1608.01338`, 1–30.

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, 107–124.

J. Y. Beziau, W. Carnielli, D. M. G. 2007. *Handbook of Paraconsistency (Studies in Logic)*. College Publications, United States.

Kifer, M. and Lozinskii, E. L. 1992. A logic for reasoning with inconsistency. *J. Autom. Reasoning 9,* 2, 179–215.

Kifer, M. and Subrahmanian, V. S. 1992. Theory of generalized annotated logic programming and its applications. *J. Log. Program. 12,* 3&4, 335–367.

Ponnuru, H., Finkel, R. A., Marek, V. W. and Truszczynski, M. 2004. Automatic generation of English-language steps in puzzle solving. In *Proceedings of the International Conference on Artificial Intelligence, IC-AI '04, June 21-24, 2004, Las Vegas, Nevada, USA, Volume 1*. CSREA Press, Las Vegas, Nevada, USA, 437–442.

Priest, G., Tanaka, K. and Weber, Z. 2015. Paraconsistent logic. In *The Stanford Encyclopedia of Philosophy*, Spring 2015 ed., E. N. Zalta, Ed. Stanford, USA.

Schwitter, R. 2013. The Jobs Puzzle: Taking on the challenge via controlled natural language processing. *Theory and Practice of Logic Programming 13,* 4-5, 487–501.

Shapiro, S. C. 2000. An introduction to sneps 3. In *Conceptual Structures: Logical, Linguistic, and Computational Issues, 8th International Conference on Conceptual Structures, ICCS 2000, Darmstadt, Germany, August 14-18, 2000, Proceedings*. Springer, Darmstadt, Germany, 510–524.

Shapiro, S. C. 2011. The jobs puzzle: A challenge for logical expressibility and automated reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, California, USA, March 21-23, 2011*. AAAI, Stanford, California, USA.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press

Soininen, T., Niemelä, I., Tiihonen, J. and Sulonen, R. 2001. Representing configuration knowledge with weight constraint rules. In *Answer Set Programming, Towards Efficient and Scalable Knowledge Representation and Reasoning, Proceedings of the 1st Intl. ASP'01 Workshop, Stanford, March 26-28, 2001*. Springer, Stanford, California, USA.

Syrjänen, T. and Niemelä, I. 2001. The Smodels system. In *Logic Programming and Nonmonotonic Reasoning, 6th International Conference, LPNMR 2001, Vienna, Austria, September 17-19, 2001, Proceedings*. Springer, Vienna, Austria, 434–438.

White, C. and Schwitter, R. 2009. An update on PENG Light. In *Proceedings of ALTA*. Vol. 7. Springer, Sydney, Australia, 80–88.

Wos, L., Overbeck, R., Lusk, E. and Boyle, J. 1984. *Automated reasoning: Introduction and applications*. Prentice Hall Inc.,Old Tappan, NJ, United States.

https://doi.org/10.1017/S1471068416000326 Published online by Cambridge University Press