

Assumable Answer Set Programming

Zhizheng Zhang

School of Computer Science and Engineering, Southeast University, No.2 Dongnandaxue Rd, Nanjing, 211198, China

Abstract

For modeling the assumption-based intelligent agents who make assumptions and use them to construct their belief sets, this paper proposes a logic programming language AASP (Assumable Answer Set Programming) by extending ASP (Answer Set Programming). Rational principles of assumption-based reasoning are given to define the semantics of the AASP program. The relation of AASP to some existing default formalism and extensions of ASP implies that AASP can be used to model and solve some interesting problems with incomplete information in the framework of assumption-based reasoning.

Keywords

assumption based reasoning, answer set, logic programming

1. Introduction

In the case of incomplete information, one framework of the mental behavior for an intelligent agent is to make assumptions, and use them to construct belief set through deductive reasoning [1]. Various approaches for assumption-based or hypothetical reasoning have been proposed in the past. Examples include Assumption-Based Truth Maintenance System introduced in [2], [3], [4], and [5], Probabilistic Assumption-Based Model and language proposed in [6] and [7], Poole's Default Theory in [8] and [9] etc.. Some efforts are made to explore the way of identifying assumptions in reasoning, in which assumptions are not given explicitly. For example, [10] explores the derivation of assumptions to explain observed events, [11] and [12] present an approach to hypothetical planning involves generating assumptions about actions that can not be derived from the knowledge-base etc.. Besides, many studies show that assumption-based reasoning is closely related to the topics like argumentation [13], action reasoning [14], planning [15], contextual reasoning [1], defeasible reasoning [16], logic programming [17], default reasoning [18] [19] etc..

Presently, Answer Set Programming (ASP) has become an increasingly popular tool for declarative programming, knowledge representation, and nonmonotonic reasoning [20] [21] [22] [23]. Answer sets are widely accepted as a rational reasoner's views in an environment described by a logic program that may contain incomplete information. The increasing success of the answer-set based reasoning inspires us to explore how to use it in both assumption making and belief building of the framework of the assumption-based reasoning.

This paper presents a new logic programming formalism for assumption-based reasoning by combining the idea of ASP and the framework of assumption-based reasoning. Specifically, we propose Assumable Answer Set Programming (AASP) language, an extension of ASP, that can be used to design the assumption-based intelligent agent whose mental behaviors of assumption making and belief building are defined on the answer-set based reasoning.


ICLP Workshops 2022, July 31, 2022, Haifa, Israel

✉ seu_zzz@seu.edu.cn (Z. Zhang)

🆔 0000-0001-9851-6184 (Z. Zhang)

© 2022 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

The rest of the paper will introduce AASP formally and is organized as follows. In the next section, ASP, two extensions of ASP, and constrained default logic are briefly introduced as background knowledge for the self-contained requirement and as objects compared with AASP. In section 3, we introduce syntax, semantics, and some properties of the AASP program. In section 4, the relation between AASP and some other formalism is given. We conclude in section 5 with some further discussion.

We will restrict our discussion in this paper to propositional programs. However, as usual in answer set programming, we admit rule schemata containing variables bearing in mind that these schemata are just convenient representations for the set of their ground instances.

2. Preliminaries

2.1. Answer Set Program

Follow the description of ASP from [22]. A regular ASP program is a collection of rules of the form

$$l_1 \text{ or } \dots \text{ or } l_k \leftarrow l_{k+1}, \dots, l_m, \text{ not } l_{m+1}, \dots, \text{ not } l_n$$

where the l s are literals, *not* denotes negation as failure, *or* is epistemic disjunction. The left-hand side of a rule is called the *head* and the right-hand side is called the *body*. A rule is called a fact if its body is empty (equivalent to containing only a literal \top) and its head contains only one literal, and a rule is called a denial if its head is empty (equivalent to containing only a literal \perp). An answer set X of a program Π if it is the minimal set (in the sense of set inclusion) that satisfies Π^X , where Π^X is G-L reduct of Π with respect to X achieved by two rules:

- delete all rules whose bodies are not satisfied by X .
- delete *not* l in the bodies of the remaining rules.

$AS(\Pi)$ is used to denote the set of all answer sets of an ASP program Π .

2.1.1. CR-Prolog

CR-Prolog extends the regular ASP with a purpose of representing indirect exceptions to defaults ([22]). Follow the description of CR-Prolog from [24] and [22], a CR-Prolog program is a collection of regular ASP rules or consistency-restoring rules (CR-rule) of the form

$$l_1 \text{ or } \dots \text{ or } l_k \overset{+}{\leftarrow} l_{k+1}, \dots, l_m, \text{ not } l_{m+1}, \dots, \text{ not } l_n$$

where the l s are literals, *not* denotes negation as failure, *or* is epistemic disjunction. And, \leq is a partial order defined on sets of CR-rules in the program. This partial order is often referred to as a preference relation based on the set-theoretic inclusion ($R_1 \leq R_2$ iff $R_1 \subset R_2$) or defined by the cardinality of the corresponding sets ($R_1 \leq R_2$ iff $|R_1| \subset |R_2|$).

The set of regular ASP rules of a CR-Prolog program Ω is denoted by Ω^r ; By $\alpha(r)$ we denote a regular rule obtained from a consistency-restoring rule r by replacing $\overset{+}{\leftarrow}$ by \leftarrow , and α can be expanded in a standard way to a set R of CR-rules, i.e., $\alpha(R) = \{\alpha(r) | r \in R\}$.

A minimal (with respect to the preference relation of the program) collection R of CR-rules of Ω such that $\Omega^r \cup R$ is consistent (i.e., has an answer set) is called an abductive support of Ω . Then, a set M is called an answer set of Ω if it is an answer set of a regular program $\Omega \cup \alpha(R)$ for some abductive support R of Ω . We use $AS_c(\Omega)$ and $AS_\#(\Omega)$ to denote the collection of answer sets of Ω w.r.t. the preference relation on set-theoretic inclusion and the cardinality respectively.

2.1.2. Abductive ASP

We consider two versions of the abductive answer set programs. In [25], an abductive logic program (ALP93) Γ is defined as a pair $\langle P, \mathcal{A} \rangle$ where P is a regular ASP program and \mathcal{A} is a set of literals from the language of P called abducibles. G a ground literal represents a positive observation. A set S is a belief set of Γ with respect to E if S is an answer set of $P \cup E$ where $E \subseteq \mathcal{A}$. S is called \mathcal{A} minimal if there is no belief set T of Γ such that $T \cap \mathcal{A} \subset S \cap \mathcal{A}$. A set E is an explanation of G with respect to Γ if G is true in a belief set S of Γ such that $E = S \cap \mathcal{A}$. An explanation E of G is minimal if no $E' \subset E$ is an explanation of G . E is a minimal explanation of G iff S is an \mathcal{A} minimal belief set of $\langle P \cup \{\leftarrow \text{not } G\}, \mathcal{A} \rangle$.

In [26], an abductive logic program (ALP95) Γ is defined as a pair $\langle P, \mathcal{A} \rangle$ where both P and \mathcal{A} are regular ASP programs. G a ground literal represents a positive observation. A pair (E, F) is an explanation of G with respect to Γ if

1. $G \subseteq M$ for $\forall M \in AS((P - F) \cup E)$
2. $(P - F) \cup E$ is consistent
3. $E \subseteq (\mathcal{A} - P)$ and $F \subseteq \mathcal{A} \cap P$

On the other hand, a pair (E, F) is an anti-explanation of G with respect to Γ if

1. $G \not\subseteq M$ for $\forall M \in AS((P - F) \cup E)$
2. $(P - F) \cup E$ is consistent
3. $E \subseteq (\mathcal{A} - P)$ and $F \subseteq \mathcal{A} \cap P$

An (anti-)explanation (E, F) of G is called minimal if for any (anti-)explanation (E', F') of G , $E' \subseteq E$ and $F' \subseteq F$ imply $E' = E$ and $F' = F$.

2.2. Constrained Default Logic

As defined by Reiter in [27], a default theory Δ is a pair (D, W) where W is a set of first-order formulas and D is a set of defaults of the form

$$\frac{\alpha : M\beta}{\gamma}$$

where α , β , and γ are quantifier-free first order formulas. α is called the *prerequisite*, β the *justifications*, and γ the *consequent*. Formally, For any set S of first order logic formulas, Let $\Gamma(S)$ be the smallest set of formulas such that

- $W \subseteq \Gamma(S)$
- $\Gamma(S) = Th(\Gamma(S))$
- if $\frac{\alpha : M\beta}{\gamma} \in D$ and $\alpha \in \Gamma(S)$ and $\neg\beta \notin S$ then $\gamma \in \Gamma(S)$

A set of formulas E is an extension of Δ iff $E = \Gamma(E)$

[28] proposes a variant of the extension of Reiter's default logic, for any set T of first order logic formulas, let $Y(T)$ be the pair of smallest sets (S', T') of formulas such that

- $W \subseteq S' \subseteq T'$
- $S' = Th(S')$ and $T' = Th(T')$
- For any $\frac{\alpha : M\beta}{\gamma} \in D$, if $\alpha \in S'$ and $T' \cup \{\beta, \gamma\} \not\vdash \perp$ then $\gamma \in S'$ and $\beta \wedge \gamma \in T'$.

A pair of sets of formulas (E, C) is a constrained extension of Δ iff $Y(C) = (E, C)$.

3. Assumable Answer Set Program

3.1. Syntax

An AASP rule r is written as

$$l_1 \text{ or } \dots \text{ or } l_k \leftarrow e_1, \dots, e_m : l_{k+1}, \dots, l_n.$$

where the l s are literals in propositional logic language and e s are literals possibly preceded by negation as failure *not*, $:$ is called assumption operator. $head(r)$ is used to denote the left-hand side of r where *or* is an epistemic disjunction, and $headlit(r)$ is used to denote the set $\{l_1, \dots, l_k\}$ of literals in the head of r . $body(r)$ is used to denote the right-hand side of r , and $bodylit(r)$ is used to denote the set $\{e_1, \dots, e_m, l_{k+1}, \dots, l_n\}$ of literals in the body of r . e_1, \dots, e_m is called the precondition of r and denoted by $pbody(r)$. Let $pbodylit(r)$ denote the set $\{e_1, \dots, e_m\}$ of literals in the precondition of r . l_{k+1}, \dots, l_n is called the assumption of r and denoted by $abody(r)$. Let $abodylit(r)$ denote the set $\{l_{k+1}, \dots, l_n\}$ of literals in the assumption of r . As in usual logic programming, a rule is called a fact if its body is empty (equival to containing only a literal \top) and its head contains only one literal, and a rule is called a constraint if its head is empty (equival to containing only a literal \perp). An AASP rule is called assumption-free if its assumption is empty, otherwise it is called an assumption rule. Sometimes, we use $head(r) \leftarrow body(r)$ or $head(r) \leftarrow pbody(r) : abody(r)$ to denote r . $lit(r)$ is used to denote the set of propositional logic literals appearing in r . r can be read as *On the assumption of $abody(r)$, $head(r)$ is believed if $pbody(r)$ is believed or $head(r)$ is believed if $pbody(r)$ is believed and it is consistent to assume $abody(r)$.*

An AASP program is a collection of AASP rules. $lit(\Pi)$ is used to denote the set of propositional logic literals appearing in Π . For convenient description, sometimes an AASP program Π is written as a pair (Π^D, Π^W) in which Π^W is the set of assumption-free AASP rules in Π and Π^D is the set of assumption rules in Π .

It is clear that an assumption-free AASP rule is a regular ASP rule, and an assumption-free AASP program is an ASP program that can be dealt with by ASP solvers like DLV ([29]), CLASP ([30]).

3.2. Semantics

3.2.1. Satisfiability

Let M be a collection of literals, r be an AASP rule, the notion of satisfiability denoted by \vDash_{AASP} is defined below.

- $M \vDash_{\text{AASP}} l$ if $l \in M$
- $M \vDash_{\text{AASP}} \text{not } l$ if $l \notin M$
- $M \vDash_{\text{AASP}} head(r)$ if $\exists l \in headlit(r), M \vDash_{\text{AASP}} l$
- $M \vDash_{\text{AASP}} pbody(r)$ if $\forall e \in pbodylit(r), M \vDash_{\text{AASP}} e$
- $M \vDash_{\text{AASP}} abody(r)$ if $\forall e \in abodylit(r), M \vDash_{\text{AASP}} e$
- $M \vDash_{\text{AASP}} body(r)$ if $M \vDash_{\text{AASP}} abody(r)$ and $M \vDash_{\text{AASP}} pbody(r)$.
- $M \vDash_{\text{AASP}} r$ if whenever $M \vDash_{\text{AASP}} body(r), M \vDash_{\text{AASP}} head(r)$.

We say M is a model of an AASP program Π , denoted by $M \vDash_{\text{AASP}} \Pi$, if we have $M \vDash_{\text{AASP}} r$ for $\forall r \in \Pi$. A set M of literals is inconsistent if it contains a literal l and its contrary \bar{l} .

3.2.2. Assumable Answer Set

We first introduce the notion of *Assumption Set* of an AASP program that is viewed as the result of assumption making in the framework of assumption-based reasoning.

Definition 1. Given an AASP program Π , an arbitrary set $A \subseteq \text{lit}(\Pi)$, A is an assumption set of Π if and only if

$$A \in AS(\Pi^{(A)} \cup \overleftarrow{A})$$

where $\Pi^{(A)}$ is a regular ASP program obtained by

$$\Pi^{(A)} = \{\text{head}(r) \leftarrow \text{pbody}(r) \mid r \in \Pi \text{ and } A \models_{\text{AASP}} \text{abody}(r)\}$$

and \overleftarrow{A} is used to denote the rules set $\{l \leftarrow \mid l \in A\}$.

$ASS(\Pi)$ is used to denote the collection of all assumption sets of an AASP program Π .

Example 1. Consider Π_1 :

$$p \leftarrow : q$$

$A_1 = \emptyset$, $A_2 = \{p\}$, $A_3 = \{q\}$, and $A_4 = \{p, q\}$. We have

$$\begin{aligned} \Pi_1^{(A_1)} &= \emptyset & \Pi_1^{(A_2)} &= \emptyset \\ \Pi_1^{(A_3)} &= \{p \leftarrow\} & \Pi_1^{(A_4)} &= \{p \leftarrow\} \end{aligned}$$

Therefore,

$$\Pi_1^{(A_1)} \cup \overleftarrow{A_1} = \emptyset \quad \Pi_1^{(A_2)} \cup \overleftarrow{A_2} = \{p \leftarrow\}$$

both $\Pi_1^{(A_3)} \cup \overleftarrow{A_3}$ and $\Pi_1^{(A_4)} \cup \overleftarrow{A_4}$ are:

$$\begin{aligned} p \leftarrow \\ q \leftarrow \end{aligned}$$

Then,

$$\begin{aligned} AS(\Pi_1^{(A_1)} \cup \overleftarrow{A_1}) &= \{\emptyset\} & AS(\Pi_1^{(A_2)} \cup \overleftarrow{A_2}) &= \{\{p\}\} \\ AS(\Pi_1^{(A_3)} \cup \overleftarrow{A_3}) &= \{\{p, q\}\} & AS(\Pi_1^{(A_4)} \cup \overleftarrow{A_4}) &= \{\{p, q\}\} \end{aligned}$$

Thus, we have

$$\begin{aligned} A_1 &\in AS(\Pi_1^{(A_1)} \cup \overleftarrow{A_1}) & A_2 &\in AS(\Pi_1^{(A_2)} \cup \overleftarrow{A_2}) \\ A_3 &\notin AS(\Pi_1^{(A_3)} \cup \overleftarrow{A_3}) & A_4 &\in AS(\Pi_1^{(A_4)} \cup \overleftarrow{A_4}) \end{aligned}$$

Hence, \emptyset , $\{p\}$, and $\{q, p\}$ are assumption sets of Π_1 , $\{q\}$ is not.

Definition 2. Given an AASP program Π , an arbitrary set $M \subseteq \text{lit}(\Pi)$, M is an assumable answer set of Π if and only if there exists an assumption set A of Π such that

1. $M \in AS(\Pi^{(A)})$, and
2. $M \subseteq A$

We say that M is an assumable answer set of Π on the assumption set A , and that (M, A) is a view of Π .

$AAS(\Pi)$ is used to denote the collection of all assumable answer sets of an AASP program Π . $VIEW(\Pi)$ is used to denote the collection of all views of an AASP program Π .

Example 2. Continue Π_1 mentioned in the above example. Consider $M_1 = \emptyset$ and $M_2 = \{p\}$, obviously, we have

$$AS(\Pi_1^{(A_1)}) = \{\emptyset\} \quad AS(\Pi_1^{(A_2)}) = \{\emptyset\} \quad AS(\Pi_1^{(A_4)}) = \{p\}$$

Then,

$$M_1 \in AS(\Pi_1^{(A_1)}) \text{ and } M_1 \subseteq A_1$$

$$M_1 \in AS(\Pi_1^{(A_2)}) \text{ and } M_1 \subseteq A_2$$

$$M_2 \in AS(\Pi_1^{(A_4)}) \text{ and } M_2 \subseteq A_4$$

Hence, \emptyset is an assumable answer set of Π_1 on A_1 , \emptyset is also an assumable answer set of Π_1 on A_2 , and $\{p\}$ is an assumable answer set of Π_1 on A_4 .

The intuitions implied in the definitions are as the following principles based on the ASP-based reasoning:

1. Rationality of Assumption Making. This principle tells that, in assuming a formula, the agent reasons and behaves as if it is a fact.
2. Rationality of Belief Building on Assumptions. This principle tells that the agent's beliefs are obtained by reasoning within the scope of assumptions.
3. Consistency between Assumptions and Beliefs. This principle tells that an agent's assumptions and beliefs must be consistent.

The notion of *Assumption Set* is defined by the first principle. By the second and the third principles, the notion of *Assumable Answer Set* is defined.

In the scenario of incomplete information, different assumption sets may be generated by a different set of assumption rules in the program. Some of them are from more assumption rules and some of them from less. Just as the example above shows, the assumption set \emptyset satisfies no assumption of the rule in the program Π_1 , and $\{p, q\}$ satisfies the assumption of one rule in the program. Based on this observation, we define several strategies of assumption making while keeping the principles unchanged.

Definition 3. Given an AASP program Π , A is an assumption set of Π ,

1. A is a max_c assumption set of Π if there is no assumption set A' of Π such that $\Pi^{(A)} \subset \Pi^{(A')}$.
2. A is a min_c assumption set of Π if there is no assumption set A' of Π such that $\Pi^{(A)} \supset \Pi^{(A')}$.
3. A is a $max_\#$ assumption set of Π if there is no assumption set A' of Π such that $|\Pi^{(A)}| < |\Pi^{(A')}|$.
4. A is a $min_\#$ assumption set of Π if there is no assumption set A' of Π such that $|\Pi^{(A)}| > |\Pi^{(A')}|$.

Definition 4. Given an AASP program Π , M is called max_c (min_c or $max_\#$ or $min_\#$) assumable answer set of Π if (M, A) is a view of Π and A is a max_c (min_c or $max_\#$ or $min_\#$) assumption set of Π . Correspondingly, the pair (M, A) is called a max_c (min_c or $max_\#$ or $min_\#$) view of Π .

max_c (min_c or $max_\#$ or $min_\#$)- $AAS(\Pi)$ is used to denote the collection of all max_c (min_c or $max_\#$ or $min_\#$) assumable answer sets of an AASP program Π . max_c (min_c or $max_\#$ or $min_\#$)- $VIEW(\Pi)$ is used to denote the collection of all max_c (min_c or $max_\#$ or $min_\#$) views of an AASP program Π .

Example 3. Continue Π_1 mentioned in the above examples. Obviously, both \emptyset and $\{p\}$ are not only \min_{\subseteq} but also $\min_{\#}$ assumption sets, while $\{p, q\}$ is both \max_{\subseteq} and $\max_{\#}$ assumption set. Thus, \emptyset is a \min_{\subseteq} assumable answer set and also a $\min_{\#}$ assumable answer set of Π_1 . $\{p\}$ is a \max_{\subseteq} assumable answer set and also a $\max_{\#}$ assumable answer set of Π_1 .

The intuitions of *max* and *min* strategies of assumption making are direct: *max* means that the reasoner is positive/optimistic/credulous in making assumptions. *min* is just the opposite. Let us consider a common advice “Work hard and you will succeed” that can be represented by an AASP program with one rule

$$\text{succeed} \leftarrow : \text{workhard}$$

Then, a positive reasoner’s view is the $\max_{\subseteq}(\max_{\#})$ view $(\{\text{succeed}\}, \{\text{workhard}, \text{succeed}\})$ of the program, a passive reasoner’s view is the $\min_{\subseteq}(\min_{\#})$ view (\emptyset, \emptyset) of the program.

3.3. Properties

In this subsection, some properties of AASP are given.

Theorem 1. For a assumption-free AASP program Π :

$$\text{AAS}(\Pi) = X_{\star} - \text{AAS}(\Pi) = \text{AS}(\Pi)$$

where $X \in \{\max, \min\}$ and $\star \in \{\subseteq, \#\}$.

Theorem 2. For an AASP program Π , an assumption set of Π is a model of Π .

Theorem 3. For an AASP program Π and a literal subset $A \subseteq \text{lit}(\Pi)$

$$A \in \text{AS}(\Pi^{(A)} \cup \overleftarrow{A}) \text{ if and only if } A \in \text{AS}(\Pi \downarrow_A \cup \overleftarrow{A})$$

where

$$\Pi \downarrow_A = \{\text{head}(r) \leftarrow \text{pbody}(r), \text{abody}(r) \mid r \in \Pi \text{ and } A \vDash_{\text{AASP}} \text{abody}(r)\}$$

Theorem 3 tells that an alternative definition of the notion of *Assumption Set* is: A is an assumption set of Π if $A \in \text{AS}(\Pi \downarrow_A \cup \overleftarrow{A})$, that can be viewed as another way of formalizing the principle of *Rationality of Assumption Making*.

Theorem 4. For an AASP program $\Pi = (\Pi^D, \Pi^W)$, let A be an assumption set of Π

$$\Pi^W \subseteq \Pi^{(A)}$$

Theorem 5. For an AASP program $\Pi = (\Pi^D, \Pi^W)$, let M be an assumable answer set of Π

$$M \vDash_{\text{AASP}} \Pi^W$$

Theorem 5 tells that an assumable answer set of an AASP program always satisfies the assumption-free rules in the program.

Theorem 6. AASP is constraint monotonic under the assumable answer set semantics.

Theorem 7. AASP is not constraint monotonic under the X_{\star} assumable answer set semantics, where $X \in \{\max, \min\}$ and $\star \in \{\subseteq, \#\}$.

Theorem 7 can be demonstrated by the following example.

Example 4. Π_2 is an AASP program containing one rule:

$$p \leftarrow : p$$

Π_2 has only one \max_c (also $\max_\#$) assumable answer set $\{p\}$ and only one \min_c (also $\min_\#$) assumable answer set \emptyset . Consider Π'_2 :

$$p \leftarrow : p$$

$$\leftarrow p$$

Clearly, Π'_2 has only one \max_c ($\max_\#$) assumable answer set \emptyset that is not a \max_c (or $\max_\#$) assumable answer set of Π_2 . Consider Π''_2 :

$$p \leftarrow : p$$

$$\leftarrow \text{not } p$$

Obviously, Π''_2 has only one \min_c (or $\min_\#$) assumable answer set $\{p\}$ that is not a \min_c (or $\min_\#$) assumable answer set of Π_2 .

4. Relations

For exploring the power of AASP in commonsense reasoning, this section presents the relationship between AASP and several well-known knowledge representation languages, including constrained default logic [28], CR-Prolog [31], and abductive logic programming [25] [26]. First of all, we compare AASP and ASP in representing assumptions.

4.1. Comparison of AASP and ASP in Representing Assumptions

It seems that the assumption rule *On the assumption of α , β is believed if γ is believed* can also be coded into an ASP rule $\beta \leftarrow \gamma, \text{not } \neg\alpha$ where *not $\neg\alpha$* is used to express *α is assumable or it is consistent to assume α* . However, the following cases demonstrate the difference between ASP encodings and AASP encodings of the assumptions.

Firstly, let us consider a case containing:

- An assumption: *p if it is consistent to assume p .*
- A constraint: *p is impossible.*

Intuitively, the constraint is a denial of p such that the assumption is blocked, thus the result is \emptyset . If the case is modeled by an ASP program

$$p \leftarrow \text{not } \neg p$$

$$\leftarrow p$$

there is no solution because the ASP program is unsatisfiable. If the case is represented by an AASP program

$$p \leftarrow : p$$

$$\leftarrow p$$

there is an assumable answer set \emptyset as expected.

Now, let us consider another case with two assumptions:

- p if it is consistent to assume r , and
- q if it is consistent to assume $\neg r$.

If they are represented as an ASP program

$$p \leftarrow \text{not } \neg r$$

$$q \leftarrow \text{not } r$$

The result is its answer set $\{p, q\}$. Meanwhile, if they are represented as an AASP program

$$p \leftarrow : r$$

$$q \leftarrow : \neg r$$

There are three assumable answer sets $\{p\}$, $\{q\}$, and \emptyset . Among them, both $\{p\}$ and $\{q\}$ are max_c (and max_*) assumable answer sets, and \emptyset is a min_c (and min_*) assumable answer set. Consider that r and its contrary $\neg r$ cannot appear in one world, the results given by the AASP program should be more praised than that of the ASP program.

Another seeming ASP-based encoding of *it is consistent to assume α* is *not not α* . But, it is easy to verify that the encoding of the second case in this way

$$p \leftarrow \text{not not } r$$

$$q \leftarrow \text{not not } \neg r$$

has only one answer set \emptyset .

The above two examples demonstrate that it is hard to represent assumptions using the regular ASP language, and that AASP provides an easy approach to handling assumptions by extending ASP with a new operator $:$.

4.2. Relation to Constrained Default Logic

Constrained default logic meets some properties that are in line with human intuition in common-sense reasoning. The following theorem shows that AASP is closely related to the constrained default logic.

Define a mapping ϕ from a disjunction-free and NAF-free program of AASP to default theories, identifies an AASP rule r :

$$l_0 \leftarrow l_1, \dots, l_k : l_{k+1}, \dots, l_n \quad (1)$$

with the default $\phi(r)$:

$$\frac{l_1 \wedge \dots \wedge l_k : Ml_{k+1} \wedge \dots \wedge l_n}{l_0} \quad (2)$$

Then we have

Theorem 8. *For any disjunction-free and NAF-free AASP program Π , if (S, A) is a max_c view of Π , then $(Th(S), Th(A))$ is a constrained extension of $\phi(\Pi)$.*

Example 5. Consider Π_3 :

$$c \leftarrow : b$$

$$d \leftarrow : \neg b$$

has two \max_c views: $(\{c\}, \{b, c\})$, and $(\{d\}, \{d, \neg b\})$. $\{c, d\}$ is not an assumable answer set of Π_3 because $\{b, \neg b, c, d\}$ is not an assumption set of the program. Correspondingly, $\phi(\Pi_3)$ is a default theory containing two defaults:

$$\frac{: Mb}{c} \quad \frac{: M\neg b}{d}$$

that has two extensions:

$$(Th(c), Th(b, c))$$

and

$$(Th(d), Th(\neg b, d))$$

4.3. Relation to CR-Prolog

CR-Prolog is an ASP extension that greatly extends the expression ability of ASP language. The following theorem provides an approach to converting a CR-Prolog program into an AASP program.

Define a mapping η from a CR-prolog to AASP, identifies a CR-rule r :

$$l_1 \text{ or } \dots \text{ or } l_k \leftarrow^+ l_{k+1}, \dots, l_m, \text{ not } l_{m+1}, \dots, \text{ not } l_n \quad (3)$$

with an AASP rule $\eta(r)$:

$$l_1 \text{ or } \dots \text{ or } l_k \leftarrow l_{k+1}, \dots, l_m, \text{ not } l_{m+1}, \dots, \text{ not } l_n : \text{apply}_r \quad (4)$$

where apply_r is used to denote the fresh atom obtained from a CR-rule r . Besides, η identifies a regular ASP rule in the CR-prolog program with itself.

Theorem 9. For any CR-Prolog program Ω

$$AS_\star(\Omega) = \min_\star - AAS(\eta(\Omega))$$

where $\star \in \{c, \#\}$.

Example 6. Consider a simple CR-Prolog program Ω that contains only one CR-rule r :

$$a \leftarrow^+$$

It is easy to see Ω has only one answer set \emptyset . Then, an AASP program $\eta(\Omega)$ is

$$a \leftarrow : \text{apply}_r$$

whose $\min_\#$ and \min_c assumable answer set is also \emptyset .

4.4. Relation to Abductive ASP

Define a mapping θ , for an ALP93 program $\Gamma = \langle P, \mathcal{A} \rangle$, $\theta(\Gamma)$ is an AASP program:

$$P \cup \{l \leftarrow : l \in \mathcal{A}\}$$

We have

Theorem 10. *For the ALP93 program $\Gamma = \langle P, \mathcal{A} \rangle$, S is a \min_{\subseteq} assumable answer set of $\theta(\Gamma)$ if and only if S is a \mathcal{A} -minimal belief set of Γ .*

Example 7. *Consider an abductive logic program $\Gamma_1 = \langle P, \mathcal{A} \rangle$ in [32]:*

- $P: p \leftarrow \text{not } a$
- $\mathcal{A}: a$

The program has one \mathcal{A} -minimal belief set $\{p\}$. $\theta(\Gamma_1)$ is an AASP

$$\begin{aligned} p &\leftarrow \text{not } a \\ a &\leftarrow : a \end{aligned}$$

Both $\{a\}$ and $\{p\}$ are its assumable answer sets, only $\{p\}$ is its \min_{\subseteq} assumable answer set as expected.

Theorem 11. *For the ALP93 program $\Gamma = \langle P, \mathcal{A} \rangle$ and a positive observation G .*

1. *If S is a \min_{\subseteq} assumable answer set of the AASP program $\theta(\Gamma) \cup \{\leftarrow \text{not } G\}$, then $S \cap \mathcal{A}$ is a credulous explanation of G with respect to Γ .*
2. *If E is a credulous explanation of G with respect to Γ , then there exists a \min_{\subseteq} assumable answer set S of the AASP program $\theta(\Gamma) \cup \{\leftarrow \text{not } G\}$ such that $E = S \cap \mathcal{A}$.*

Example 8. *Continue to consider Γ_1 mentioned in Example 7, given a positive observation $G = p$. Clearly, \emptyset is a credulous explanation of p with respect to Γ_1 . Now, we have $\theta(\Gamma_1) \cup \{\leftarrow \text{not } G\}$:*

$$\begin{aligned} p &\leftarrow \text{not } a \\ a &\leftarrow : a \\ &\leftarrow \text{not } p \end{aligned}$$

that has a \min_{\subseteq} assumable answer set \emptyset such that $E = \emptyset$.

Define a mapping θ' , for the ALP95 program $\Gamma = \langle P, \mathcal{A} \rangle$, $\theta'(\Gamma)$ is an AASP program:

$$\begin{aligned} (P - \mathcal{A}) \cup \{ \text{head}(r) \leftarrow \text{body}(r), \text{apply}_r \mid r \in (\mathcal{A} - P) \} \cup \\ \{ \text{apply}_r \leftarrow : \text{apply}_r \mid r \in (\mathcal{A} - P) \} \cup \\ \{ \text{head}(r) \leftarrow \text{body}(r), \text{not } \text{block}_r \mid r \in (\mathcal{A} \cap P) \} \cup \\ \{ \text{block}_r \leftarrow : \text{block}_r \mid r \in (\mathcal{A} \cap P) \} \end{aligned}$$

where both apply_r and block_r are used to denote the fresh atoms obtained from $r \in \mathcal{A}$. We have

Theorem 12. *For the ALP95 program $\Gamma = \langle P, \mathcal{A} \rangle$ and a positive observation G .*

1. If S is a \min_c assumable answer set of the AASP program $\theta'(\Gamma) \cup \{\leftarrow \text{not } G\}$, then there exists a minimal explanation of G with respect to Γ is

$$(\{r|apply_r \in S\}, \{r|block_r \in S\})$$

2. If (E, F) is a minimal explanation of G with respect to Γ , then there exists a \min_c assumable answer set S of the AASP program $\theta'(\Gamma) \cup \{\leftarrow \text{not } G\}$ such that

$$E = \{r|apply_r \in S\}$$

$$F = \{r|block_r \in S\}$$

Example 9. Consider an abductive logic program $\Gamma_2 = \langle P, \mathcal{A} \rangle$ and a positive observation G :

- P :

$$\begin{aligned} fly &\leftarrow bird \\ bird &\leftarrow penguin \\ penguin &\leftarrow \end{aligned}$$

- \mathcal{A} :

$$\begin{aligned} (1). fly &\leftarrow bird \\ (2). \neg fly &\leftarrow penguin \end{aligned}$$

- $G: \neg fly$

Thus, $\theta'(\Gamma_2)$ is:

$$\begin{aligned} bird &\leftarrow penguin \\ penguin &\leftarrow \\ fly &\leftarrow bird, \text{not } block_1 \\ \neg fly &\leftarrow penguin, apply_2 \\ block_1 &\leftarrow : block_1 \\ apply_2 &\leftarrow : apply_2 \end{aligned}$$

Then, $\theta'(\Gamma_2) \cup \{\leftarrow \text{not } \neg fly\}$ has a \min_c assumable answer set

$$\{apply_2, block_1, \neg fly, penguin, bird\}$$

by which

$$\begin{aligned} E &= \{\neg fly \leftarrow penguin\} \\ F &= \{fly \leftarrow bird\} \end{aligned}$$

such that (E, F) is a minimal explanation of $\neg fly$ with respect to Γ_2 .

Theorem 13. For the ALP95 program $\Gamma = \langle P, \mathcal{A} \rangle$ and a positive observation G .

1. If S is a \min_c assumable answer set of $\theta'(\Gamma) \cup \{\leftarrow G\}$, then there exists a minimal anti-explanation of G with respect to Γ is

$$(\{r|apply_r \in S\}, \{r|block_r \in S\})$$

2. If (E, F) is a minimal anti-explanation of G with respect to Γ , then there exists a \min_{\subseteq} assumable answer set S of $\theta'(\Gamma) \cup \{\leftarrow G\}$ such that

$$E = \{r|apply_r \in S\}, F = \{r|block_r \in S\}$$

Example 10. Continue to consider the abductive logic program Γ_2 used in the Example 9. By the Theorem 13, the anti-explanation of fly with respect to Γ_2 is the \min_{\subseteq} assumable answer set of the AASP program

$$\theta'(\Gamma_2) \cup \{\leftarrow fly\}$$

Obviously, a \min_{\subseteq} assumable answer set of the program is $\{penguin, bird\}$ that tells neither (1) nor (2) is used, and the corresponding minimal anti-explanation (E, F) of fly is

$$E = \emptyset \quad F = \{fly \leftarrow bird\}$$

5. Conclusion

This paper introduces AASP that extends ASP with an operator \vdash to express the notion of assumption in logic programming. AASP can be viewed as a tool to design the intelligent agent capable of assumption-based reasoning that is a framework of many intelligent behaviors in the case of incomplete information. AASP provides a rational approach to assumption-based reasoning by using the answer set-based reasoning in both assumption making and belief building, which makes the existing ASP solvers can be used to facilitate the implementation of the AASP solver: ASP solvers can be directly used to generate assumption sets and then the assumable answer set of an AASP program. Several strategies of assumption making are given in the definition of the semantics of AASP. Those strategies depict the attitude of agents to assumptions. The preliminary exploration results on the relationship between AASP and other existing knowledge representations show that AASP provides a more general way to model the problems with defaults, exceptions, and to solve the explanation problems. Especially, the close relationship between AASP and constrained default logic implies that AASP is able to address several limitations of the original ASP in representing defaults¹.

Future work includes more properties of the AASP languages, implementation, and applications. The first next step is to investigate the properties of the assumption set of the AASP programs, and the algorithm and its complexity in solving AASP programs. Besides, it is still needed to explore the relation of AASP to ASP and its extensions, and the relation of AASP to other knowledge representation languages more deeply.

Acknowledgments

We are grateful to the anonymous referees for their useful comments on the earlier version of this paper. The work was supported by the Pre-research Key Laboratory Fund for Equipment (Grant No. 6142101190304).

¹The limitations of Reiter's default logic are extensively studied in [28]. Because of the close relationship between ASP and Reiter's default logic shown in [33] and [22], ASP cannot meet some properties, such as the property of joint consistency of the justifications of applying default rules

References

- [1] M. Jago, Modelling assumption-based reasoning using contexts, in: Proceedings of workshop on Context Representation and Reasoning (CRR'05), 2005.
- [2] J. de Kleer, An assumption-based tms, *Artificial Intelligence* 28 (1986) 127–162.
- [3] J. de Kleer, Extending the atms, *Artificial Intelligence* 28 (1986) 163–196.
- [4] R. Reiter, J. de Kleer, Foundations of assumption-based truth maintenance systems: Preliminary report, in: Proceedings of AAAI-87, 1987.
- [5] J. de Kleer, A general labeling algorithm for assumption-based truth maintenance, in: Proceedings of AAAI-88, 1988, pp. 188–192.
- [6] J. Kohlas, P.-A. Monney, Probabilistic assumption-based reasoning, in: *Uncertainty in Artificial Intelligence*, 1993, pp. 485–491.
- [7] B. Anrig, R. Haenni, J. Kohlas, N. Lehmann, Assumption-based modeling using abel, in: Proceedings of ECSQARU-FAPR, 1997, pp. 171–182.
- [8] D. L. Poole, A logical framework for default reasoning, *Artif. Intell.* 36 (1988) 27–47.
- [9] D. L. Poole, Who chooses the assumptions, in: *Abductive Reasoning*, MIT Press, 1997.
- [10] P. T. Cox, T. Pietrzykowski, Causes for events: their computation and applications, in: *International Conference on Automated Deduction*, Springer, 1986, pp. 608–621.
- [11] H. Reichgelt, N. Shadbolt, Planning as theory extension, in: Proceedings of AISB-89, 1989, pp. 191–199.
- [12] H. Reichgelt, N. Shadbolt, A specification tool for planning systems, in: *ECAI-1990*, 1990, pp. 541–546.
- [13] A. Bondarenko, F. Toni, R. A. Kowalski, An assumption-based framework for non-monotonic reasoning, in: *LPNMR-93*, 1993, pp. 171–189.
- [14] R. A. Kowalski, F. Sadri, Reconciling the event calculus with the situation calculus, *J. Log. Program.* 31 (1997) 39–58.
- [15] D. Pellier, H. Fiorino, Multi-agent assumption-based planning, in: *IJCAI-05*, 2005, pp. 1717–1718.
- [16] G. K. Giannikis, A. Daskalopulu, Defeasible reasoning with e-contracts, in: *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2006, pp. 690–694.
- [17] D. Stamate, Assumption based multi-valued semantics for extended logic programs, in: *36th International Symposium on Multiple-Valued Logic (ISMVL'06)*, 2006, pp. 10–10.
- [18] M. Kaminski, A comparative study of open default theories, *Artif. Intell.* 77 (1995) 285–319.
- [19] G. K. Giannikis, A. Daskalopulu, The representation of e-contracts as default theories, in: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, Springer, 2007, pp. 963–973.
- [20] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: *ICLP/SLP*, 1988.
- [21] C. Baral, *Knowledge representation, reasoning and declarative problem solving*, 2003.
- [22] M. Gelfond, Y. Kahl, *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*, Cambridge University Press, 2014.
- [23] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, *AI Mag.* 37 (2016) 53–68.
- [24] M. Balduccini, M. Gelfond, Logic programs with consistency-restoring rules, in: *AAAI Technical Report SS-03-05*, 2003.
- [25] K. Inoue, C. Sakama, Transforming abductive logic programs to disjunctive programs, in: *ICLP-93*, 1993, p. 335.
- [26] K. Inoue, C. Sakama, Abductive framework for nonmonotonic theory change., in: *IJCAI*, volume 95, Citeseer, 1995, pp. 204–210.

- [27] R. Reiter, A logic for default reasoning, *Artif. Intell.* 13 (1980) 81–132.
- [28] T. Schaub, On constrained default theories, in: *ECAI*, 1992, pp. 304–308.
- [29] W. Faber, G. Pfeifer, N. Leone, T. Dell’armi, G. Ielpa, Design and implementation of aggregate functions in the dlv system, *Theory Pract. Log. Program.* 8 (2008) 545–580.
- [30] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: From theory to practice, *Artif. Intell.* 187-188 (2012) 52–89.
- [31] M. Balduccini, Cr-models: An inference engine for cr-prolog, in: *International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer, 2007, pp. 18–30.
- [32] C. Sakama, K. Inoue, Abductive logic programming and disjunctive logic programming: their relationship and transferability, *J. Log. Program.* 44 (2000) 75–100.
- [33] V. Lifschitz, Answer set programming and plan generation, *Artif. Intell.* 138 (2002) 39–54.