Argumentation-based abduction in disjunctive logic programming

Kewen Wang *

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, People’s Republic of China

Received 25 July 1997; received in revised form 25 February 1999; accepted 20 January 2000

Abstract

In this paper, we propose an argumentation-based semantic framework, called DAS, for disjunctive logic programming. The basic idea is to translate a disjunctive logic program into an argumentation-theoretic framework. One unique feature of our proposed framework is to consider the disjunctions of negative literals as possible assumptions so as to represent incomplete information. In our framework, three semantics preferred disjunctive hypothesis (PDH), complete disjunctive hypothesis (CDH) and well-founded disjunctive hypothesis (WFDH) are defined by three kinds of acceptable hypotheses to represent credulous, moderate and skeptical reasoning in artificial intelligence (AI), respectively. Furthermore, our semantic framework can be extended to a wider class than that of disjunctive programs (called bi-disjunctive logic programs). In addition to being a first serious attempt in establishing an argumentation-theoretic framework for disjunctive logic programming, DAS integrates and naturally extends many key semantics, such as the minimal models, extended generalized closed world assumption (EGCWA), the well-founded model, and the disjunctive stable models. In particular, novel and interesting argumentation-theoretic characterizations of the EGCWA and the disjunctive stable semantics are shown. Thus the framework presented in this paper does not only provide a new way of performing argumentation (abduction) in disjunctive deductive databases, but also is a simple, intuitive and unifying semantic framework for disjunctive logic programming.

Keywords: Disjunctive logic programs; Argumentation; Abduction; Semantics

1. Introduction

Abduction is usually defined as inferring the best or most reasonable explanation (or hypothesis) for a given set of facts. Moreover, it is a form of non-monotonic reasoning, since explanations which are consistent in a given context may become
inconsistent when new information is obtained. In fact, abduction plays an important role in much of human inference. It is relevant to our everyday commonsense reasoning as well as in many expert problem-solving tasks. Several efforts have been devoted recently to extend non-disjunctive logic programming to perform abductive reasoning, such as Refs. [1,14,17,19,22–24,41,42]. Abduction with logic programs can be used in various fields of AI, including default reasoning, diagnosis and legal reasoning. Two key forms of approaches to abduction are well known in the community of logic programming: consistency-based abduction and argumentation-based abduction. The first exploits a special logical consistency and defines an acceptable hypothesis as the corresponding consistent set (some other constraints might also be applied), such as Refs. [19,23]; the latter depends on an attack relation among hypotheses and acceptable hypotheses are defined through some stability conditions [14,17,24,41,42]. We believe that the argumentation-based abduction allows an easier and more direct representation for reasoning of law and related knowledge than the consistency-based ones. This approach is currently only applicable to non-disjunctive logic programs.

We are often required to deal with disjunctive information in our everyday life as well as in various artificial intelligence (AI) applications, for example, reasoning by cases, approximate reasoning, legal reasoning, diagnosis, and natural language understanding [7]. To conveniently and properly handle the representation and reasoning of disjunctive information in logic programming, a great deal of efforts have been given to the problem of finding suitable extensions of logic programming. The extension of logic programs by introducing disjunction in the heads of program clauses (that is, disjunctive logic programming) has been widely accepted as a promising tool for representing incomplete knowledge and it is well known that the paradigm of disjunctive logic programming is significantly more expressive than non-disjunctive logic programming. The problem of finding a suitable (declarative) semantics for disjunctive logic programs, however, has been proven to be more difficult than the case of non-disjunctive logic programs. Many approaches have been proposed to tackle this problem, some of which are well known and implemented in deductive databases and non-monotonic reasoning systems. These include the disjunctive stable models [31], the static semantics [33], the generalized closed world assumption (GCWA) [28] and the extended GCWA (EGCWA) [45].

Despite some work being done in relating consistency-based abduction with disjunctive logic programs [3,11,26,29,38], the problem of how to perform argumentation-based abduction in disjunctive logic programming is rarely explored seriously [6,29]. There are many good reasons convincing the importance of argumentation-based abduction with disjunctive logic programming. For example, argumentation-based abduction can be used in deriving explanation or prediction to given observations in disjunctive deductive databases. Given a knowledge base $KB$: If one is not happy, he often likes to stay in a dark room; If the electricity is not supplied, the room will be dark; When the room is dark, one is sleeping or thinking. $KB$ can be expressed as a disjunctive logic program $P$:

\[
\begin{align*}
\text{RoomDark} & \leftarrow \sim \text{Happy} \\
\text{RoomDark} & \leftarrow \sim \text{ElectricitySupplied} \\
\text{Sleeping} | \text{Thinking} & \leftarrow \text{RoomDark}
\end{align*}
\]
where the intuitive meanings of ~ and | are ‘not’ and ‘or’, respectively. If we observe that, in the evening, Mike lies in bed (sleeping or thinking, but we are not exactly aware of which) and we want to know why this is so, an explanation to this observation should include a hypothesis (prediction) \( \Delta_1 = \{\sim Happy\} \) or \( \Delta_2 = \{\sim ElectricitySupplied\} \).

In this paper, we shall explore the relationship between argumentation-based abduction and disjunctive logic programming. As a result, we propose an argumentation-theoretic semantic framework called DAS for disjunctive logic programs and many interesting results are obtained (some of which are non-trivial generalizations of the corresponding ones for the case of non-disjunctive logic programming, others are quite new).

Besides providing for a suitable argumentation-based semantics for disjunctive logic programming, this framework is also motivated by the following reasons: (1) A unifying framework can often provide a tool for comparing different semantics (including their relationship, expressive power and complexities), and the implementation of different semantics can also be based on a unifying mechanism. (2) A framework often results in several new semantics and helps to overcome the weakness of some key semantics for disjunctive logic programs. (3) A semantic framework is in fact a non-deterministic semantics and thus it often has more expressive power that can enhance modeling capabilities of the corresponding systems. (4) Various approaches of defining semantics for disjunctive programs have shown that no single semantics is satisfactory for all applications. It is always possible to give an example where the existing semantics is not the intended meaning. This fact is leading a never ending story of seeking new semantics. A flexible framework that integrates different semantics is needed to solve this problem.

The fundamental idea of our work is to introduce a special resolution which resolves a default-negation literal with a disjunction and to interpret the disjunctions of negative literals as abducibles (or, assumptions). As a result, a given disjunctive program \( P \) is naturally transformed into an argument framework \( F_P = (P, H(P), \sim \rightarrow P) \), where \( H(P) \) is the set of all disjunctive hypotheses of \( P \), \( \sim \rightarrow P \) is an attack relation among the hypotheses. An admissible hypothesis \( \Delta \) is one that can attack every hypothesis which attacks it. Based on this intuitive idea, we introduce mainly three subclasses of admissible hypotheses: preferred disjunctive hypothesis (PDH); complete disjunctive hypothesis (CDH) and well-founded disjunctive hypothesis (WFDH). Each of these subclasses defines a declarative semantics for disjunctive programs and all of them are complete for the class of disjunctive programs, that is, every disjunctive program has at least one PDH (resp. CDH, WFDH).

As noted in Ref. [17], skepticism and credulism are two major semantic intuitions for knowledge representation. A skeptical reasoner does not infer any conclusion in uncertainty conditions, but a credulous reasoner tries to give conclusions as much as possible. The framework in this paper integrates these two opposite semantic intuitions and, in particular, PDH and WFDH characterize credulism and skepticism, respectively. This observation will be further convinced by the related results and examples in subsequent sections. Our abductive framework cannot only handle the problems of commonsense reasoning properly, but many interesting theoretical results are obtained. We shall show that this semantic framework characterizes and extends many key semantics: (1) WFDH extends both the well-found semantics for non-disjunctive logic programs [21] and the extended generalized closed world
assumption (EGCWA) [45]. As a result, Theorem 4.2 provides a unifying characterization for these two different semantics through argumentation-based abduction and suggests a new way of performing argumentation and abduction in disjunctive deductive databases. In fact, Theorem 4.2 may be one of the most interesting results in this paper. (2) PDH is not only complete but also naturally extends the disjunctive stable semantics [31]. Thus, PDH provides a complete extension for disjunctive stable semantics. DAS can be considered as a generalization of Dung’s preferred scenarios [14,17] and Torres’ non-deterministic well-founded semantics [41,42]. In fact, this paper is heavily influenced by their work and our study also shows that such a generalization is non-trivial and interesting.

At the same time, our semantic framework can be naturally established for a wider class than that of disjunctive programs, called bi-disjunctive logic programs, which is a subclass of super logic programs [7].

The rest of this paper is arranged as follows: Section 2 is devoted to establish the basic argumentation-theoretic framework DAS for disjunctive programs. In Section 3 we mainly define three declarative semantics PDH, CDH and WFDH in DAS, give some examples and extend DAS to the class of bi-disjunctive logic programs. In Section 4, we investigate WFDH and its relation to other skeptical semantics. In particular, we show that WFDH is a natural characterization and extension of EGCWA. To examine the relation of PDH to some other credulous semantics, in Section 5, we first define a program transformation $Lft$ for disjunctive programs and then introduce a simple subclass of PDHs, called the stable PDHs. We show that there is a one-to-one correspondence between the stable PDHs and the disjunctive stable models of a disjunctive program. Section 6 is our conclusion.

An extended abstract of this paper has appeared as Ref. [44], the main contents of which are a part of Ref. [43]. Technical results and some proofs of theorems are given in Appendix A.

2. Argumentation in disjunctive logic programming

In this section, we first introduce some necessary definitions and notions, then establish our basic argumentation-theoretic framework for disjunctive logic programs; in the last subsection we shall introduce a syntactical extension of disjunctive logic programs (bi-disjunctive logic programs) and generalize our argumentation-theoretic framework to this class of bi-disjunctive programs. As usual, without loss of generality, we consider only propositional logic programs, this means that a logic program is often understood as its ground instantiation.

2.1. Basic notions and definitions

Throughout the paper we shall refer to the following different classes of logic programs:

A Horn logic program is a set of Horn clauses of the form

$$a \leftarrow a_1, \ldots, a_m,$$

where $a$ and $a_i$ ($i = 1, \ldots, m$) are atoms and $m \geq 0$. 


A non-disjunctive logic program is a set of non-disjunctive clauses of the form

\[ a \leftarrow a_1, \ldots, a_s, \neg a_{s+1}, \ldots, \neg a_t, \]

where \( a \) and \( a_i \) \((i = 1, \ldots, t)\) are atoms and \( t \geq s \geq 0 \). The symbol \( \neg \) denotes negation by default, rather than classical negation.

A positive disjunctive logic program is a set of positive disjunctive clauses of the form

\[ a_1| \cdots |a_r \leftarrow a_{r+1}, \ldots, a_s, \]

where \( a_i \) \((i = 1, \ldots, s)\) are atoms and \( s \geq r > 0 \). The symbol | is the epistemic disjunction rather than the disjunction in classical logic.

A negative disjunctive logic program is a set of negative disjunctive clauses of the form

\[ a_1| \cdots |a_r \leftarrow \neg a_{r+1}, \ldots, \neg a_s, \]

where \( a_i \) \((i = 1, \ldots, s)\) are atoms and \( s \geq r > 0 \).

A (general) disjunctive logic program is a set of disjunctive clauses of the form

\[ a_1| \cdots |a_r \leftarrow a_{r+1}, \ldots, a_s, \neg a_{s+1}, \ldots, \neg a_t, \]

where \( a_i \) \((i = 1, \ldots, t)\) are atoms and \( t \geq s \geq r > 0 \).

Notice that the body of the above clause will be empty if \( s = t = r \). A clause with empty body is also called a fact. Since \( r > 0 \), we will not allow a clause with empty head.

As usual, \( B_P \) denotes the Herbrand base of disjunctive logic program \( P \), that is, the set of all (ground) atoms in \( P \). The set \( DB_P^+ \) of all disjunctions of the atoms in \( P \) is called the disjunctive Herbrand base of \( P \); the set \( DB_P^- \) of all disjunctions of the negative literals in \( P \) is called the negative disjunctive Herbrand base of \( P \). \( \bot \) denotes the empty disjunction.

If \( S \) is an expression, then atoms\((S)\) is the set of all atoms appearing in \( S \).

For \( x, \beta \in DB_P^+ \), if atoms\((x) \subseteq \text{atoms}(\beta)\) then we say \( x \) implies \( \beta \), denoted as \( x \Rightarrow \beta \). If \( x \in DB_P^+ \), then the smallest factor \( sfac(x) \) of \( x \) is the disjunction of atoms obtained from \( x \) by deleting all repeated occurrence of atoms in \( x \) (if \( x \) is not propositional, the definition will not be so simple, see [27]). For instance, the smallest factor of \( a|b|a \) is \( a|b \). For \( S \subseteq DB_P^+ \), \( sfac(S) = \{ sfac(x) : x \in S \} \). The expansion of \( x \) is defined as \( ||x|| = \{ \beta \in DB_P^- : x \Rightarrow \beta \} \); the expansion of \( S \) is \( ||S|| = \{ \beta \in DB_P^- : \text{there exists } x \in S \text{ such that } x \Rightarrow \beta \} \).

The canonical form of \( S \) is defined as \( can(S) = \{ x \in sfac(S) : \text{there exists no } x' \in sfac(S) \text{ such that } x' \Rightarrow x \text{ and } x' \neq x \} \).

For \( x \in DB_P^- \) and \( S \subseteq DB_P^- \), \( sfac(x), sfac(S), ||x|| \) and \( ||S|| \) can be defined similarly.

A subset of \( DB_P^+ \) is called a state of the disjunctive logic program \( P \); a state-pair of \( P \) is defined as \( S = (S^+, S^-) \), where \( S^+ \subseteq DB_P^+ \) and \( S^- \subseteq DB_P^- \).

The minimal models and the least model-state are two important declarative semantics for positive disjunctive programs, both of which extend the least model theory of Horn logic programs. The minimal model semantics captures the disjunctive consequences from a positive disjunctive program as a set of models. The least model-state captures the disjunctive consequences as a set of disjunctions of atoms and leads to a unique ‘model’ characterization.
Let $P$ be a positive disjunctive program, then the least model-state of $P$ is defined as

$$ms(P) = \{ x \in DB^+_P : \models x \},$$

where $\models$ is the inference of the first-order logic and $P$ is considered as the corresponding first-order formulas.

The least model-state $ms(P)$ of a positive disjunctive program $P$ can be characterized by the operator $T^S_P : 2^{DB^+_P} \rightarrow 2^{DB^+_P}$: for any $J \subseteq DB^+_P$,

$$T^S_P(J) = \{ x \in DB^+_P : \text{there exist a disjunctive clause } a' \leftarrow a_1, \ldots, a_n \text{ in } P \text{ and } a_i| x_i \in J, i = 1, \ldots, n, \text{ such that } x'' = a'| x_1| \cdots | x_n, \text{ where } x_1, \ldots, x_n \in DB^+_P \cup \{ \bot \}, \text{ and } x = sfac(x'') \}.$$

Minker and Rajasekar [30] have shown that $T^S_P$ has the least fixpoint $lfp(T^S_P) = T^S_P \uparrow \omega$, and the following result:

**Theorem 2.1.** Let $P$ be a positive disjunctive program, then $ms(P) = ||T^S_P \uparrow \omega||$, and $ms(P)$ has the same set of minimal models as $P$.

### 2.2. The basic argumentation-theoretic framework

This subsection will be devoted to establish our argumentation-theoretic framework for disjunctive programs. In general, argumentation-based abduction is based on argument frameworks $F = \langle K, H, \rightarrow \rangle$, where $K$ is a logical theory representing the given knowledge base, $H$ a set of formulae representing the possible hypotheses, and $\rightarrow$ is an attack relation among the hypotheses. The basic idea in this subsection is based on Refs. [17,42] but our approach is a little different from Dung’s. As mentioned in the introduction, their framework is established only for non-disjunctive programs. In particular, argument framework in disjunctive logic programming will be quite different from that of non-disjunctive logic programming.

Given a disjunctive program $P$, a disjunctive assumption (or simply, assumption) of $P$ means an element of $DB^+_P$; a disjunctive hypothesis (or simply, hypothesis) of $P$ is defined as a subset $\Delta$ of $DB^+_P$ such that $\Delta$ is expansion-closed: $||\Delta|| = \Delta$. In this paper, we shall take each disjunctive program $P$ as a special argument framework $F_p = \langle P, H(P), \rightarrow_p \rangle$, where $H(P)$ is the set of all hypotheses of $P$, and $\rightarrow_p$ is a (binary) attack relation on $H(P)$, also referred as the attack relation of $P$.

An assumption $\beta = \sim b_1| \cdots | \sim b_n$ is true disjunctive if $n > 1$.

To define the above attack relation $\rightarrow_p$ of $F_p$, similar to the definition of GL-transformation [20], we first define a generalized GL-transformation for the class of disjunctive programs, by which a positive disjunctive program $P^*_{\Delta}$ is obtained from any given disjunctive program $P$ and a (disjunctive) hypothesis $\Delta$ of $P$.

**Definition 2.1.** Let $\Delta$ be a hypothesis, then
1. For each disjunctive clause $C$ in $P$, delete all the negative literals in the body of $C$ that belong to $\Delta$. The resulting disjunctive program is denoted as $P_\Delta$.
2. The positive disjunctive program consisting of all the positive disjunctive clauses of $P_\Delta$ is denoted as $P^*_{\Delta}$, and is said to be the generalized GL-transformation of $P$. 
Example 2.1. Consider the disjunctive program $P$:

\[
\begin{align*}
  a & \leftarrow b, \sim c \\
  b | c & \leftarrow \sim e \\
  b | c | d & \leftarrow
\end{align*}
\]

If $\Delta = \| \sim c \|$, then $P^+_\Delta$ is the positive disjunctive program:

\[
\begin{align*}
  a & \leftarrow b \\
  b | c | d & \leftarrow
\end{align*}
\]

Based on this transformation, we shall introduce a special resolution $\vdash_P$ which resolves default-negation literals with a disjunction and can be intuitively illustrated as the following principle:

Assume that there is an agent who
1. holds the assumptions $\sim b_1, \ldots, \sim b_m$; and
2. can ‘derive’ a disjunctive information $b_1 | \cdots | b_m | b_{m+1} | \cdots | b_n$ from the knowledge base $P$ with these assumptions. Then the disjunctive information $b_{m+1} | \cdots | b_n$ is obtained.

The following definition precisely formulates this principle in the setting of disjunctive logic programming.

**Definition 2.2.** Let $\Delta$ be a (disjunctive) hypothesis of disjunctive program $P$, $x \in DB^+_P$. If there exist $\beta \in DB^+_P$ and $\sim b_1, \ldots, \sim b_m \in \Delta$ such that the following two conditions are satisfied:
1. $\beta = x | b_1 | \cdots | b_m$; and
2. $\beta \in \text{can}(ms(P^+_\Delta))$.

Then $\Delta$ is said to be a supporting hypothesis for $x$, denoted as $\Delta \vdash_P x$. The set of all disjunctions of positive literals that are supported by $\Delta$ is $V_P(\Delta) = \{x \in DB^+_P : \Delta \vdash_P x\}$.

The above condition (2) means that $\beta$ is a logical consequence of $P^+_\Delta$ with respect to the least model-state.

In Example 2.1, it is obvious that $ms(P^+_\Delta) = \{a | c | d; b | c | d\}$. Hence $a | d$ and $b | d$ can be obtained from $P$ with hypothesis $\Delta = \| \sim c \|$. In fact, $V_P(\Delta) = \|a | d; b | d\|$.

**Definition 2.3.** For any hypothesis $\Delta$ of disjunctive program $P$, we say that the tuple $S_\Delta = (\|V_P(\Delta)||; \Delta)$ is a supported state-pair of $P$.

The state-pair here for DLP corresponds to the scenario of Dung [17].

The task of defining a semantics for a (disjunctive) logic program $P$ is to determine the state-pairs that can represent the intended meaning of $P$. That is, a semantics of logic program $P$ is only a set of its state-pairs. Since all the state-pairs considered in this paper are determined by the corresponding hypotheses, we can also understand a semantics as a set of hypotheses. Though each hypothesis $\Delta$ corresponds to a state-pair of $P$, not every state-pair represents the intended meaning of $P$. For example $P = \{a | b \leftarrow \sim a, \sim b\}$. If $\Delta = \| \sim a, \sim b \|$, then $V_P(\Delta) = \{a | b\}$ and thus $S_\Delta = (\|a | b\|; \| \sim a, \sim b\|)$. It is obvious that $S_\Delta$ does not represent the correct meaning of $P$. 

To derive suitable hypotheses for a given disjunctive program, some constraints will be required to filter unintuitive hypotheses. The accomplishment of this task will be based on the following fundamental definition.

**Definition 2.4.** Let $\Delta$ and $\Delta'$ be two hypotheses of disjunctive program $P$. If at least one of the following two conditions holds:

1. There exists $b = \sim b_1 | \cdots | \sim b_m \in \Delta'$, $m > 0$, such that $\Delta \vdash_P b_i$, for all $i = 1, \ldots, m$; or
2. There exist $\sim b_1, \ldots, \sim b_m \in \Delta'$, $m > 0$, such that $\Delta \vdash_P b_1 | \cdots | b_m$, then we say $\Delta$ attacks $\Delta'$, and denoted as $\Delta \rightarrow_P \Delta'$.

Intuitively, $\Delta \rightarrow_P \Delta'$ means that $\Delta$ causes the direct contradiction with $\Delta'$ and the contradiction may come from one of the above two cases. In Example 2.1, if $\Delta' = \| \sim a, \sim b \|$, then $\Delta \rightarrow_P \Delta'$ by Definition 2.4, but $\Delta' \not\rightarrow_P \Delta$. Thus, in general, the attack relation $\rightarrow_P$ is not symmetric. In fact, the asymmetricity of the attack relation is one attracting feature. For otherwise, this relation will have not much use.

In the remaining of this subsection, we seek to define suitable constraints on (disjunctive) hypotheses by using the above fundamental definition.

First, a plausible hypothesis should not attack itself.

**Definition 2.5.** A hypothesis $\Delta$ of disjunctive program $P$ is self-consistent if $\Delta \rightarrow_P \Delta$.

Obviously, the hypothesis $\| \sim c \|$ in Example 2.1 is self-consistent but the hypothesis $\Delta = \| \sim a, \sim b \|$ of disjunctive program $P = \{a|b \leftarrow \sim a, \sim b\}$ is not self-consistent. The empty hypothesis $\emptyset$ is always self-consistent, called trivial hypothesis. Our Example 2.1 shows that there exist non-trivial hypotheses that are not self-consistent.

The following simple corollary of Definition 2.5 will be often used in our proofs of subsequent results.

**Corollary 2.1.** A hypothesis $\Delta$ of disjunctive program $P$ is not self-consistent if and only if there exists $\sim b_1 | \cdots | \sim b_n \in \Delta$ such that $\Delta \vdash_P b_i$, for all $i = 1, \ldots, n$.

**Proof.** The condition is obviously sufficient. For the necessity, suppose that $\Delta \rightarrow_P \Delta$, According to Definition 2.4, there are two possible cases. If the case 1 holds, then it is just the required conclusion; If the case 2 holds, then there exist $\sim b_1, \ldots, \sim b_m \in \Delta$ ($m > 0$) such that $\Delta \vdash_P b_1 | \cdots | b_m$. It follows from Definition 2.2 that $\Delta \vdash_P b_1$ and $\sim b_1 \in \Delta$. Thus, the required conclusion also holds. □

**Definition 2.6.** For any self-consistent hypothesis $\Delta$ of disjunctive program $P$, the corresponding state-pair $S_\Delta$ is called a self-consistent state-pair of $P$.

From Definition 2.2 and 2.4, it is not hard to see that the self-consistency of a hypothesis guarantees that there exists no ‘direct’ contradiction within the corresponding state-pair of this hypothesis. That is, given a self-consistent hypothesis $\Delta$ of disjunctive program $P$, neither of the following two conditions hold for the state-pair $S_\Delta = (S^+; S^-)$:
1. there exist $a_1, \ldots, a_r \in S^+$, such that $\sim a_1 | \cdots | \sim a_r \in S^-$; or
2. there exists $a_1 | \cdots | a_r \in S^+$, such that $\sim a_1, \ldots, \sim a_r \in S^-$.

However, a self-consistent state-pair may not be ‘consistent’. That is, from a self-consistent hypothesis, it is possible that conflict conclusions may be derived as the following Example 2.2 will show.

**Definition 2.7.** A state-pair $S = \langle S^+; S^- \rangle$ is **consistent** if the set of the corresponding first-order formulas of $S^+ \cup S^-$ is consistent.

A hypothesis is consistent if its corresponding state-pair is consistent.

For example, the corresponding first-order formulae of disjunctions $a_1 | \cdots | a_m$ and $\sim a_1 | \cdots | \sim a_m$ are $a_1 \lor \cdots \lor a_m$ and $\sim a_1 \lor \cdots \lor \sim a_m$, respectively.

As the following example will show, a self-consistent state-pair is not necessarily consistent though there is no direct contradiction within it.

**Example 2.2.** Let $P$ be the following disjunctive program:

\[
\begin{align*}
  a | b & \leftarrow \\
  b | c & \leftarrow \\
  c | a & \leftarrow
\end{align*}
\]

Take $\Delta = \| \sim a | \sim b, \sim b | \sim c, \sim c | \sim a \|$, then $\Delta$ is a self-consistent hypothesis. It is easy to see that $V_P(\Delta) = \{a|b, b|c, c|a\}$ but $\|V_P(\Delta)\| \cup \Delta$ as a set of first-order formulas is not consistent, thus the state-pair $S_\Delta = \langle \|V_P(\Delta)\|; \Delta \rangle$ is not consistent.

In particular, in many cases, self-consistency of state-pairs can still not provide suitable constraints for abductive semantics of disjunctive programs. The next example illustrates that we need other constraints on self-consistency hypotheses.

**Example 2.3.** Given that one can arrange his trip from Hong Kong to Changsha either by air or by train, and there is no preference between these two alternatives in principle, assume that, from the Website of the airline company, Joe found no flight Saturday from Hong Kong to Changsha (there is some chance that such a flight has not been added in the Web). Moreover, Joe has no other information available. Then on Saturday, Joe still has two alternatives: either first try to take plane or first try to take train. We form this situation as a disjunctive logic program $P$:

\[
\begin{align*}
  \text{train} | \text{plane} & \leftarrow \sim \text{plane}
\end{align*}
\]

It is not difficult to see that both $\Delta = \| \sim \text{plane} \|$ and $\Delta' = \| \sim \text{train} \|$ are self-consistent hypotheses of $P$, but $\Delta'$ should not be the intended meaning of $P$. In fact, though the preference between taking train or taking plane is neglected, one would rather go to the train station first if he has got some negative information on taking plane. Thus, the desired semantics of this program should be $\{\text{train}, \sim \text{plane}\}$.

We should determine the self-consistent hypotheses of $P$ that capture the intended semantics of disjunctive programs. In other words, we must specify when a self-consistent hypothesis of $P$ is acceptable. To accomplish this task, we need to exploit an intuitive and useful principle in argument reasoning:
If one hypothesis can attack every hypothesis that attacks it, then this hypothesis is acceptable.

This principle can be summarized vividly by an old saying: “The one who has the last word laughs best”. Many examples in our daily life can be found to illustrate this principle.

Now, we formulate this principle in the setting of disjunctive logic programming, which can really provide a suitable criteria for specifying acceptable hypotheses for disjunctive programs and forms the basis of our abductive framework for disjunctive logic programming as shown by the results in subsequent sections.

For short, if $b \vdash b_1 \ldots \vdash b_n \in DB_P$, and $\Delta'$ is a hypothesis of $P$ such that $\Delta' \vdash_P b_i$, for all $i = 1, \ldots, m$, then we say that $\Delta'$ denies $b$, or, $\Delta$ is an attack on $b$.

Given a rational hypothesis $\Delta$ of $P$, an assumption $b$ of $P$ is acceptable with respect to $\Delta$ if $\Delta$ can defend $b$ against all attacks on $b$. This motivates the following definition of admissible assumptions.

**Definition 2.8.** Let $\Delta$ be a hypothesis of disjunctive program $P$. An assumption $b$ of $P$ is admissible with respect to $\Delta$ if $\Delta$ denies $b$. Write $A_P(\Delta) = \{b \in DB_P : b$ is admissible with respect to $\Delta\}$.

In Example 2.3, the assumption $\sim plane$ is admissible with respect to $\Delta = \| \sim plane \|$ but $\sim train$ is not admissible with respect to $\| \sim train \|$.

$A_P$ in fact defines an operator from the set $H(P)$ of all hypotheses of $P$ to itself. Intuitively, an acceptable hypothesis should be one whose assumptions are all admissible with respect to itself. Thus the following definition is in order.

**Definition 2.9.** A hypothesis $\Delta$ of disjunctive program $P$ is admissible if $\Delta$ is self-consistent and $\Delta \subseteq A_P(\Delta)$.

Again, consider the disjunctive program in Example 2.3. It can be verified that $\Delta = \| \sim plane \|$ is an admissible hypothesis, but $\Delta' = \| \sim train \|$ is not admissible.

**Lemma 2.1.** Let $\Delta$ be a hypothesis of disjunctive program $P$. If an assumption $\beta = \sim b_1 \ldots \sim b_r$ of $P$ is admissible with respect to $\Delta$, then $\beta' = \sim b_1 \ldots \sim b_i \sim b_{r+1} \ldots \sim b_n$ is also admissible with respect to $\Delta$ for any $b_{r+1}, \ldots, b_n \in B_P$ and $r \leq n$.

**Proof.** For any hypothesis $\Delta'$ of $P$ such that $\Delta'$ denies $\beta'$, then $\Delta' \vdash_P b_i$, for all $i = 1, \ldots, n$, which also means that $\Delta'$ denies $\beta$ because $r \leq n$. Since $\beta$ is admissible, it follows that $\Delta \vdash_P \Delta'$. This implies $\beta'$ is also admissible with respect to $\Delta$. □

This lemma is especially useful when we want to show that some hypotheses of a disjunctive program are admissible: To show that a hypothesis $\Delta = \| \beta_1, \ldots, \beta_n \|$ is admissible, it suffices to show that all assumptions $\beta_i$ ($i = 1, \ldots, n$) are admissible with respect to $\Delta$.

**Example 2.4.** If it is not cloudy, we often think that it is a good day; If it is not a good day, we would have to stay at home. This commonsense knowledge is represented as the program $P$: 

\[
\begin{align*}
\end{align*}
\]
By Lemma 2.1, it can be verified that the hypothesis $\Delta = \{\sim \text{Cloudy}; \sim \text{StayAtHome}\}$ is an admissible hypothesis of $P$.

From Definition 2.8 and Lemma 2.1, it is direct that the operator $A_P$ possesses the following two properties, which are fundamental to some of our subsequent results:

**Corollary 2.2.** If $\Delta$ and $\Delta'$ are two hypotheses of disjunctive program $P$, then
1. $\|A_P(\Delta)\| = A_P(\Delta)$. That is, $A_P(\Delta)$ is a hypothesis of $P$.
2. If $\Delta \subseteq \Delta'$, then $A_P(\Delta) \subseteq A_P(\Delta')$. This means that $A_P$ is a monotonic operator.

Notice that, in general, the operator $A_P$ may not be continuous as pointed by one referee. For example, let $P$ be the logic program:

\[
\begin{align*}
a & \leftarrow \sim b \\
b & \leftarrow \sim a \\
c & \leftarrow \sim a, \sim b
\end{align*}
\]

Take $\Delta_1 = \{\sim a\}$ and $\Delta_2 = \{\sim b\}$. Then $A_P(\Delta_1) = \{\sim a\}$, $A_P(\Delta_2) = \{\sim b\}$, and $A_P(\Delta_1 \cup \Delta_2) = \{\sim a, \sim b, \sim c\}$. Thus, $A_P(\Delta_1 \cup \Delta_2) \neq A_P(\Delta_1) \cup A_P(\Delta_2)$. That is, $A_P$ is not continuous.

In this section we have established an abductive framework for disjunctive logic programming (abbreviated as DAS), in which various semantics for performing argumentation-based abduction with disjunctive programs can be defined. Each semantics in our framework will be defined as a subclass of admissible hypotheses (equivalently, admissible state-pairs).

As shown in Example 2.2, the self-consistency of a hypothesis is weaker than the consistency in general. However, we believe that every ADH of a disjunctive program is consistent, but such an accurate proof has not been found.

### 3. Some important disjunctive semantics

The semantics of stratified non-disjunctive programs lead to unique minimal models (that is, the perfect model) [2], which is well accepted as the intended meaning of stratified programs. However, this is not the case when we consider the class of non-stratified programs or disjunctive programs (even positive disjunctive programs) and a lot of approaches have been proposed to determine semantics for non-stratified programs and/or disjunctive programs. Though some semantics are promising in logic programming and knowledge representation, such as the well-founded semantics for non-disjunctive programs, the EGCWA for positive disjunctive programs and the disjunctive stable semantics for general logic programs, they are often

---

1 This example is due to a referee.
criticized for their shortcomings. For example, the problem of the (disjunctive) stable semantics is its incompleteness: some disjunctive programs do not possess any stable models; the well-founded semantics is not able to express the non-deterministic nature of non-stratified programs and most of its extensions to disjunctive programs are quite unnatural, etc. The diversity of various approaches in semantics for (disjunctive) logic programs shows that there is probably not a unique suitable semantics for applications in logic programming. Therefore, as argued in Section 1, a suitable non-deterministic semantics rather than only a single semantics for disjunctive logic programming should be provided, in which most of the existing key semantics should be embedded and their shortcomings should be overcome. Both theoretical research and implementation of systems in disjunctive logic programming will benefit from such a semantic framework. As well as investigating the inherent relationship between argumentation (abduction) and disjunctive logic programming, we shall attempt to show that our argumentation-theoretic semantic framework defined in Section 2 can provide such a (at least potentially) suitable framework for disjunctive logic programming by defining some abductive semantics and relating them to some key semantics for logic programs, such as the well-founded model, minimal models, disjunctive stable models and EGCWA.

3.1. Argumentation-theoretic semantics

Based on the intuition of commonsense reasoning, such as credulism and skepticism, we introduce the following three subclasses of admissible hypotheses, which define three important declarative semantics for disjunctive logic programs. As mentioned before, a credulous argumentation reasoner should infer as more assumptions as possible and is often defined by a kind of maximality; a skeptical one will be very cautious and is often defined by a kind of minimality; but a moderate one should be between the above two.

**Definition 3.1.** Let $\Delta$ be a hypothesis of disjunctive program $P$:

1. A preferred disjunctive hypothesis (PDH) $\Delta$ of $P$ is defined as a maximal ADH of $P$ with respect to set inclusion;
2. $\Delta$ is a complete disjunctive hypothesis (CDH) of $P$ if $\Delta$ is self-consistent and $\Delta = A_P(\Delta)$;
3. The well-founded disjunctive hypothesis (WFDH) of $P$ is its least CDH, denoted as $\text{WFDH}(P)$.

The semantics PDH, CDH and WFDH of $P$ are defined as the set of all ADHs, the set of all CDHs and the WFDH($P$), respectively.

Notice that the definition of WFDH is well-defined, since we shall show, in Section 4, that every disjunctive logic program possesses the (unique) least CDH.

If $\Delta$ is an ADH (resp. PDH, CDH, WFDH), then the corresponding supported state-pair $S_\Delta$ is called an ADS (resp. PDS, CDS, WFDS) of $P$.

It follows easily from the above definition that a CDH must be an ADH and, in Section 3.2, we shall show that each PDH is also a CDH. But the converses do not hold.
Example 3.1. Let $P$ consist of only one program clause: $a \leftarrow b$. Take $\Delta_0 = \emptyset$, then $A_P(\Delta_0) = \{ \lnot a \}. \\ \ \text{Hence $\Delta_0$ is an ADH of $P$ but not a CDH.}$ If $\Delta_1 = \{ \lnot a \}$, then $A_P(\Delta_1) = \Delta_1$ and thus $\Delta_1$ is a CDH of $P$ but not a PDH, since $\Delta_2 = \{ \lnot a \}$ is an ADH of $P$ and $\Delta_1 \subseteq \Delta_2$.

The following logic programs are well known in the community of logic programming and non-monotonic reasoning, all these programs are abstracted from examples of commonsense reasoning as benchmarks to justify the suitability of semantics for logic programs. By these examples, thus, one can examine the suitability of our semantics before some theoretical results are presented.

Example 3.2. Again, consider the simplest (positive) disjunctive program $P = \{ a \leftarrow b \}$.

This program has four ADHs: $\Delta_0 = \emptyset$, $\Delta_1 = \{ \lnot a \}$, $\Delta_2 = \{ \lnot b \}$, $\Delta_3 = \{ \lnot a \} \cup \{ \lnot b \}$, among which $\Delta_1, \Delta_2, \Delta_3$ are all CDHs of $P$, but $P$ has only two PDHs: $\Delta_1, \Delta_2$ (just corresponding to two stable models of $P$, respectively; the definition of the disjunctive stable models see Section 5.1 or [31]). Thus, a credulous reasoner can make two choices: (i) inferring $a$ but denies $b$; or (ii) inferring $b$ but denies $a$. The unique WFDH $\Delta_3 = \{ \lnot a \} \cup \{ \lnot b \}$ is exactly the EGCWA($P$). This means that a skeptical reasoner can only say \( \lnot a \) and $b$ cannot be inferred from $P$.

Example 3.3 (A variant of the Barber’s Paradox [17]). Assume that the barber Noel shaves every one who does not shave himself and Casanova is a teacher. If $a$ denotes the proposition: Noel shaves himself, and $b$ denote the proposition: Casanova is a teacher. Then we have a knowledge base $P$:

\[
\begin{align*}
    a & \leftarrow \lnot a \\
    b & \leftarrow
\end{align*}
\]

The above two rules seem unrelated, therefore, we would like to derive $b$ but leave $a$ unknown.

The possible disjunctive hypotheses of $P$ are: $\Delta_0 = \emptyset$, $\Delta_1 = \{ \lnot a \}$, $\Delta_2 = \{ \lnot b \}$, $\Delta_3 = \{ \lnot a \} \cup \{ \lnot b \}$, among which $\Delta_1, \Delta_2$ are not self-consistent. Since $\Delta_1 \not\subseteq \Delta_3$ but $\Delta_3$ is an ADH of $P$, thus $P$ has only one ADH $\Delta_0 = \emptyset$ and the corresponding state-pair $S_{\Delta_0} = (\{ b \}; \emptyset)$. This conclusion coincides with our intuition on $P$, that is, $P$ provides no information about $a$ for us and thus, from $P$, we can infer neither $a$ nor $\lnot a$, but can infer $b$. Because the PDH, CDH and WFDH all are $\emptyset$, no matter a reasoner is skeptical, moderate or credulous, he will arrive at the same conclusion: $b$ is true but $a$ is unknown. Notice that the Clark’s completion $\comp(P)$ is not consistent and $P$ also has no stable model. This example shows that DAS can handle the inconsistency of disjunctive programs properly.

Example 3.4. Suppose that we have an incomplete knowledge $KB$ about John, who is teaching in a university:

(1) If John is not excellent in academic, he will be fired.
(2) If John is not excellent in teaching, he will be fired.
(3) We only know that John is excellent at least in one of academic or teaching.
Now, we may ask a question: Will John be fired? Intuitively, the correct answer should be unknown. That is, one can neither say that John will be fired nor say that John will not be fired, since the knowledge at hand is not enough to enable us to make a prediction about John’s tenure status.

Let \( a = \text{ExcellentInTeaching}, \ b = \text{ExcellentInAcademic} \) and \( c = \text{Fired} \), then this knowledge base \( KB \) can be expressed as the following disjunctive program \( P \):

\[
\begin{align*}
a & \leftarrow b \\
c & \leftarrow \neg a \\
c & \leftarrow \neg b
\end{align*}
\]

We need to consider only the following seven assumptions of \( P \):

\[
\begin{align*}
\Delta_0 &= \emptyset, \\
\Delta_1 &= \| \neg a \|, \\
\Delta_2 &= \| \neg b \|, \\
\Delta_3 &= \| \neg c \|, \\
\Delta_4 &= \| \neg a \| \land \| \neg b \|, \\
\Delta_5 &= \| \neg b \| \land \| \neg c \|, \\
\Delta_6 &= \| \neg a \| \land \| \neg c \|, \\
\Delta_7 &= \| \neg a \| \land \| \neg b \| \land \| \neg c \|, \\
\Delta_8 &= \| \neg a \| \land \| \neg b \|, \\
\Delta_9 &= \| \neg a \| \land \| \neg c \|, \\
\Delta_{10} &= \| \neg b \| \land \| \neg c \|, \\
\Delta_{11} &= \| \neg a \| \land \| \neg b \| \land \| \neg c \|, \\
\Delta_{12} &= \| \neg a \| \land \| \neg c \| \land \| \neg b \|, \\
\Delta_{13} &= \| \neg a \| \land \| \neg b \| \land \| \neg c \|, \\
\Delta_{14} &= \| \neg a \| \land \| \neg b \| \land \| \neg c \| \land \| \neg a \|, \\
\Delta_{15} &= \| \neg a \| \land \| \neg b \| \land \| \neg c \|, \\
\Delta_{16} &= \| \neg a \| \land \| \neg c \| \land \| \neg b \|, \\
\Delta_{17} &= \| \neg a \| \land \| \neg b \| \land \| \neg c \|, \\
\Delta_{18} &= \| \neg a \| \land \| \neg b \| \land \| \neg c \| \land \| \neg a \|,
\end{align*}
\]

where \( \Delta_0, \Delta_1, \Delta_2, \Delta_4 \) are all the ADHs of \( P \); \( \Delta_1, \Delta_2, \Delta_4 \) are CDHs; the PDHs \( \Delta_1, \Delta_2 \) correspond to the stable models \( \{b, c\} \) and \( \{a, c\} \), respectively. WFDH of \( P \) is \( \Delta_4 \) and the state-pair \( S_{\Delta_4} = \langle \| a \| \land \| b \| \land \| c \| \rangle \).

WFDH of \( KB \) means that we are unsure whether John should be fired. Therefore, WFDH is the correct semantics for this disjunctive program.

Notice that the state-pair of \( P \) in the stationary semantics [32] and the static semantics [33] is \( S' = \langle \| a \| \land \| b \| \land \| c \| \rangle \). Thus DAS infers the same negative information as these two semantics but DAS does not allow \( c \) to be derived from \( P \). Ross’ DWFS [35] does not allow that \( c \) is inferred from \( P \) but the inference of negative information is different from our DAS. Baral, Lobo and Minker’s generalized well-founded semantics [4] interprets \( P \) into a positive disjunctive program and \( GDWFS(P) = \langle \| a \| \land \| b \| \land \| c \| \land \| a \| \rangle \). 

However, by no means we can say that the static semantics is not suitable. In fact, the following Example 3.5 shows that, in some other cases, \( c \) needs to be derived from the knowledge base. We consider an example in legal reasoning.

**Example 3.5.** According to the law, if a man keeps a marriage relation with at least two women at the same time, he will be punished; If the judge is unable to evidence that one man keeps a marriage relation with at least two women, the man will be claimed innocent.
Suppose that, in Ted’s case, the judge at present only possesses the knowledge that
(1) Ted keeps a marriage relation with at least one of Mary or Alice (maybe both, but
the judge does not know exactly) and (2) Ted keeps no marriage relation with other
women.

Now we can formulate the judge’s knowledge base about Ted as the following
three rules (facts):

R1: Ted keeps a marriage relation with at least one of Mary or Alice.

R2: If there is no enough evidence to prove that Ted keeps a marriage relation
with Mary, Ted will be claimed innocent.

R3: If there is no enough evidence to prove that Ted keeps a marriage relation
with Alice, Ted will be claimed innocent.

Let $a = \text{marriage}(Ted, Mary)$, $b = \text{marriage}(Ted, Alice)$ and $c = \text{TedInnocent}$, then
knowledge $KB$ can also be expressed as the disjunctive program $P$ in Example 3.4.
However, this knowledge base requires that $c$ should be inferred from $P$.

For this application domain, it is obvious that the static semantics is the desired
meaning.

At this stage, we may ask if there is a semantics for DLP that can deal with both
of the reasoning applications in Examples 3.4 and 3.5. Most of the existing semantics
for DLP are unable to represent the above-mentioned two kinds of opposite reason-
ing at the same time. However, we shall illustrate, in Section 3.4, that the reasoning
in Example 3.5 can also be correctly represented by WFDH in an extension of DAS,
called BDAS.

3.2. Properties of admissible disjunctive hypotheses (ADH)

In this section we shall show some fundamental properties of our DAS including:
(1) The completeness of ADH, CDH and PDH; (2) A quite intuitive and equivalent
definition of ADHs, which shows the suitability of Definition 2.9; (3) The cumulative
property of ADHs: The hypothesis $\Delta \cup \{z\}$ is still admissible if $\Delta$ is an admissible
hypothesis and $z$ is admissible with respect to $\Delta$.

The following theorem shows that the definition of ADH really reflects the intu-
ition of argumentative reasoning.

Theorem 3.1. For any self-consistent hypothesis $\Delta$ of disjunctive program $P$, $\Delta$ is an
ADH of $P$ if and only if $\Delta \rightarrow_P \Delta'$ for any hypothesis $\Delta'$ of $P$ satisfying $\Delta' \rightarrow_P \Delta$.

Theorem 3.1 provides a quite intuitive characterization for ADH and it means
that an ADH is such a hypothesis that can attack any hypothesis that attacks it.

The following proposition states that a non-decreasing sequence of ADHs of dis-
junctive program $P$ possesses the property of completeness.

Proposition 3.1. If $\Delta_1, \Delta_2, \ldots, \Delta_n, \ldots$ is a sequence of admissible hypotheses (ADHs)
of disjunctive program $P$ such that $\Delta_n \subseteq \Delta_{n+1}$ for any $n > 0$, then the hypothesis
$\Delta = \bigcup_{n=1}^{\infty} \Delta_n$ is an ADH of $P$.

In particular, we have the following result, which shows that every ADH can be
extended to a PDH in principle.
Corollary 3.1. Each ADH of a disjunctive program $P$ is contained in a PDH.

Proof. Let $\Delta$ be an ADH of $P$ and set $S(\Delta) = \{ \Delta' \in H(P) : \Delta \subseteq \Delta' \}$. By Proposition 3.1, every non-decrease sequence in the poset $(S(\Delta), \subseteq)$ has an upper bound in $S(\Delta)$. It follows from Zorn’s Lemma that $(S(\Delta), \subseteq)$ has a maximal element $\Delta_0$. It is easy to see $\Delta \subseteq \Delta_0$ and $\Delta_0$ is a PDH of $P$. Thus, the corollary is obtained. \hfill \Box

The following cumulative property of ADHs convinces the correctness of the definitions in previous sections.

Theorem 3.2 (Fundamental property of ADHs). For any ADH $\Delta$ of disjunctive program $P$, if $\alpha \in DB_P^\Delta$ is admissible wrt. $\Delta$, that is, $\alpha \in A_P^\Delta$, then $\Delta' = |\Delta \cup \{ \alpha \}|$ is also an ADH of $P$.

This theorem guarantees that, for any ADH $\Delta$ of disjunctive program $P$, if $\alpha$ is admissible wrt. $\Delta$ and $\alpha \notin \Delta$ then we can obtain a non-trivial admissible extension of $\Delta$ by simply adding $\alpha$ to $\Delta$.

As a corollary of Theorem 3.2, the following result shows that the non-monotonic inference determined by PDH is really more credulous than that determined by CDH.

Corollary 3.2. If $\Delta$ is a PDH of disjunctive program $P$, then $\Delta$ is also a CDH of $P$.

Proof. If $\Delta$ is an ADH of $P$, then it follows from Theorem 3.2 that $A_P^\Delta$ is also an ADH. Furthermore, by $\Delta \subseteq A_P^\Delta$ and the maximality of $\Delta$, we have $\Delta = A_P^\Delta$. \hfill \Box

The existence of at least one ADH (PDH, CDH) for every disjunctive program can be guaranteed by the following result.

Theorem 3.3. For any disjunctive program $P$, all of its semantics ADH, CDH and PDH are complete. That is, every disjunctive program possesses at least one ADH (resp., CDH and PDH).

Proof. Firstly, the trivial hypothesis $\emptyset$ is an ADH. From Zorn’s Lemma, it follows that there exists at least one PDH of $P$. By Corollary 3.2, a PDH is also a CDH and thus the existence of at least one CDH is guaranteed. \hfill \Box

At present, we do not know whether WFDH is complete. The completeness of WFDH for the class of disjunctive programs will be proved in Section 4 (Theorem 4.1).

3.3. ADH for non-disjunctive programs

As a special case, we consider the DAS for non-disjunctive logic programs. In particular, we shall show that our framework for disjunctive programs can be seen as a generalization of the argumentation-theoretic (declarative) semantics for non-disjunctive programs in Refs. [17,42].
In the rest of this section, $P$ will be a non-disjunctive program. Let $\Delta$ be a disjunctive hypothesis of $P$, that is, $\Delta \in H(P)$, and $L(\Delta)$ be the set of all negative literals in $\Delta$.

**Definition 3.2.** A hypothesis $\Delta$ of $P$ is a non-disjunctive hypothesis of $P$ if $L(\Delta) = \text{can}(\Delta)$.

It is known from Definition 2.2 that, for any non-disjunctive program $P$ and $a \in B_P$,

$$\Delta \vdash_P a \iff a \in LM(P^+_\Delta) \iff a \in LM(P^+_\text{L(\Delta)}),$$

where $LM(P)$ is the least Herbrand model of $P$ for a Horn program $P$ (i.e. positive and non-disjunctive).

**Corollary 3.3.** If $\Delta$ is a CDH of non-disjunctive program $P$, then $\Delta$ is non-disjunctive. That is, every CDH of a non-disjunctive program is non-disjunctive.

The above corollary also implies that we need only consider non-disjunctive hypotheses in non-disjunctive logic programming.

The following proposition is fundamental to reveal the relation of DAS to some major semantics for non-disjunctive logic programs. It asserts that CDH and Dung’s complete extension are equivalent concepts for the class of non-disjunctive programs.

**Proposition 3.2.** If $\Delta$ is a hypothesis of non-disjunctive program $P$, then the following two statements are equivalent:

1. $\Delta$ is a CDH of $P$;
2. $P \cup L(\Delta)$ is a complete extension.

**Proof.** From the above Corollary 3.3, it follows that one will get the equivalent definition of the CDHs if the resolution $\Delta \vdash_P a$ is replaced by the first-order inference $P \cup \Delta \vdash a$. Thus, the proposition is proven.

Notice that, in the inference relation $P \cup \Delta \vdash a$, each program clause $b \leftarrow a_1, \ldots, a_r, \neg a_{r+1}, \ldots, \neg a_s$ is interpreted into a formula $b \lor \neg a_1 \lor \cdots \lor \neg a_r \lor a_{r+1} \lor \cdots \lor a_s$ in the first-order logic. For example, the program clause $b \leftarrow \neg c$ is interpreted into $b \lor \neg c$. This proposition shows that our framework really generalizes the (declarative) semantic frameworks in Refs. [17,42].

### 3.4. Argumentation and bi-disjunctive logic programs

The paradigm of disjunctive logic programming is still not expressive enough to give direct representation for some problems in commonsense reasoning. For example, suppose that we have a knowledge base consisting of four rules, which is a variant of the example in [7]:

---


---
R₁ Mike is able to visit London or Paris
R₂ If Mike is able to visit London, he will be happy
R₃ If Mike is able to visit Paris, he will be happy
R₄ If Mike is not able to visit both London and Paris, he will be prudent

It is easy to see that the rules R₁, R₂ and R₃ can be directly expressed with ordinary disjunctive logic programs as

\[
\begin{align*}
R₁ & : \text{VisitLondon} \lor \text{VisitParis} \leftarrow \\
R₂ & : \text{Happy} \leftarrow \text{VisitLondon} \\
R₃ & : \text{Happy} \leftarrow \text{VisitParis}
\end{align*}
\]

However, the rule R₄ has no direct transformation in disjunctive logic programming. Thus, it would be also desirable that the syntax of disjunctive programs should be extended to a broader class of disjunctive logic programs so that the syntax of the new class resembles that of ordinary logic programs and the new class should include ordinary disjunctive programs as a subclass. Brass et al. [7] propose a generalization for the syntax of disjunctive programs (called super logic programs) and the static semantics [33] of super logic programs is discussed. However, argumentation is not treated in their work. In this subsection, we shall first introduce an extended class of disjunctive logic programs by allowing disjunctions in the bodies of program clauses (bi-disjunctive logic programs) and then, similar to DAS, establish the corresponding argumentation-theoretic framework BDAS for bi-disjunctive programs.

Definition 3.3. A bi-disjunctive clause C is a rule of the form

\[
a₁ \mid \cdots \mid a_r \leftarrow a_{r+1}, \ldots, a_s, \beta_{s+1}, \ldots, \beta_t,
\]

where \( a_i (i = 1, \ldots, s) \) are atoms, \( \beta_j (j = s + 1, \ldots, t) \) are disjunctions of negative literals, and \( t \geq s \geq r > 0 \), where \( \mid \) is epistemic disjunction and \( \sim \) is default negation. A bi-disjunctive logic program \( P \) is defined as a finite set of bi-disjunctive clauses.

Example 3.6. The following logic program is a bi-disjunctive program:

\[
\begin{align*}
da \mid b & \leftarrow \\
ed \mid c & \leftarrow d, \sim a \mid \sim b \\
ed & \leftarrow \sim e
\end{align*}
\]

Some reasons for introducing bi-disjunctive programs can be enumerated as follows:

(1) It makes formalisms of disjunctive reasoning more expressive and natural to use since it permits direct representation of disjunctive information in logic programs from informal specifications and natural language. For example, the above rule R₄ cannot be directly expressed by a rule in ordinary disjunctive programs but it corresponds to a rule in bi-disjunctive programs, \( r₃: \text{Prudent} \leftarrow \sim \text{VisitLondon} \mid \sim \text{VisitParis} \), where the intended meaning of \( \sim \text{VisitLondon} \mid \sim \text{VisitParis} \) is that Mike is not able to visit both London and Paris.

(2) The class of bi-disjunctive programs forms a subclass of super logic programs [7] and includes the class of disjunctive programs as a subclass.

We again stress the difference between the epistemic disjunction \( \mid \) and the classical disjunction \( \lor \). For example, \( a \lor \sim a \) is a tautology but the truth of the disjunction \( a \mid \sim a \) is unknown in the disjunctive program \( P = \{a \mid b \leftarrow\} \). In particular, the intended
meaning of a disjunction \( \beta = \sim b_1 \cdots \sim b_n \) of negative literals is similar to the default atom \( \text{not}(b_1 \land \cdots \land b_n) \) in super logic programs [7]. That is, \( \beta \) means that \( b_1, \ldots, b_n \) cannot be proven at the same time. Therefore, bi-disjunctive programs can be regarded as a subclass of super programs. This means that the following inclusions hold:

Super Logic Programs \( \supset \) Bi-Disjunctive Programs \( \supset \) Disjunctive Programs \( \supset \) Non-disjunctive Programs

(3) Compared to the definitions for DAS, we shall see from the following discussions in this subsection that our argumentation-theoretic framework seems more natural in bi-disjunctive programs than in ordinary disjunctive programs.

Similar to DAS, each bi-disjunctive program \( P \) can also be transformed into an argument framework \( F_P = \langle P, H(P), \sim P \rangle \), where \( H(P) \) is the set of all disjunctive hypotheses, and \( \sim P \) is an attack relation among the hypotheses.

The generalized GL-transformation of disjunctive programs can be directly extended to the class of bi-disjunctive programs.

**Definition 3.4.** Let \( \Delta \) be a (disjunctive) hypothesis of a bi-disjunctive program \( P \):

1. For each bi-disjunctive clause \( C \) in \( P \), delete all the disjuncts of negative literals in the body of \( C \) that belong to \( \Delta \). The resulting bi-disjunctive program is denoted as \( P_\Delta \).
2. The positive disjunctive program consisting of all the positive disjunctive clauses of \( P_\Delta \) is denoted as \( P_\Delta^+ \), and is said to be the generalized GL-transformation of \( P \).

From Definition 3.4, we can see that a resolution for default negation in bi-disjunctive programs is the same as Definition 2.2. In fact, by Definition 3.4, since any bi-disjunctive program will be transformed to a positive program by the generalized GL-transformation, the definitions about DAS in Section 2 and 3 are still well-defined for the class of bi-disjunctive programs. The generalization of DAS in bi-disjunctive logic programs is denoted as BDAS.

**Example 3.7.** Let \( P \) be the bi-disjunctive program of Example 3.6. Take \( \Delta = \| \sim a \sim b, \sim e \|, \Delta' = \| \sim c \sim d \| \). Then \( P_\Delta^+ = P_\Delta = \{ a | b \leftarrow e | c \leftarrow d; d \leftarrow \}; P_\Delta^+ = \{ a | b \leftarrow \}. \) Since \( V_P(\Delta) = \{ a | b, c, d \} \), that is, \( \Delta \vdash_P c, d \) thus \( \Delta \vdash_P \Delta' \), but \( \Delta' \not\vdash_P \Delta \).

It is not difficult to see that true disjunctive assumptions do not affect the inference \( \vdash_P \) for ordinary disjunctive program \( P \), but it is not the case for bi-disjunctive programs.

For most semantics of logic programs, the rule \( c \leftarrow a | b \) is interpreted into two rules \( c \leftarrow a \) and \( c \leftarrow b \), so it is unnecessary to introduce bi-disjunctive programs. However, Example 3.4 and the following example show that the introducing of bi-disjunctive programs is not only a syntactical generalization of ordinary disjunctive programs but the BDAS for bi-disjunctive programs also possesses more expressive power than other semantics for disjunctive logic programming.

**Example 3.8.** We can also represent the knowledge base in Example 3.5 as the following bi-disjunctive program \( P' \):

\[
\begin{align*}
| a & \leftarrow b \\
| c & \leftarrow \sim a \sim b
\end{align*}
\]
Similar to Example 3.4, it can be shown that the state pair defined by $WFDH(P)$ is $WFDS(P') = < |a|b, c|, | ~ a| ~ b| >$. Therefore, we can infer $c$ from $P'$ under WFDH. This means that the reasoning in Example 3.5 is also dealt with in BDAS as well as the reasoning in Example 3.4.

4. Skeptical argumentation with disjunctive programs

As mentioned before, a skeptical agent should say nothing in ambiguous situations and our WFDH is just to represent such argumentation in disjunctive logic programming. In this section, we shall relate WFDH to a well-known and very important non-monotonic mechanism, that is, the extended generalized closed world assumption EGCWA (defined for positive disjunctive programs) [45], as well as the relation of WFDH to the well-founded semantics. In particular, we shall show that WFDH coincides with the EGCWA for positive programs in Theorem 4.2. This result has many implications: (1) WFDH naturally extends both the well-founded semantics for non-disjunctive programs and the EGCWA for positive disjunctive programs to general disjunctive programs; (2) For the first time, WFDH provides a novel and interesting argumentation-theoretic (abductive) characterization for the EGCWA; (3) Since the EGCWA has been implemented in deductive databases, WFDH suggests a new way to perform (skeptical) argumentation.

As noted in Section 3, an important feature of WFDH is its completeness. Thus, it is not difficult to see that our skeptical argumentation-theoretic semantics WFDH is complete for the class of general disjunctive logic programs:

**Theorem 4.1.** Every disjunctive program $P$ possesses the unique well-founded disjunctive hypothesis (WFDH).

**Proof.** It follows from Corollary 2.2 and Tarski’s theorem [40] that $A_P$ has the least fixpoint $lfp(A_P) = A_P \uparrow \gamma$ for some ordinal $\gamma$. This least fixpoint is, therefore, the unique WFDH of $P$. ☐

For simplicity, we assume that, from now on, each logic program contains only finite number of clauses. This is not an essential restriction and thus most of the subsequent results still hold for logic programs containing infinite number of clauses.

For any finite disjunctive program $P$, it is obvious that $lfp(A_P) = A_P \uparrow \omega$.

We then state the following two propositions, which show the relation of WFDH to the well-founded semantics and the stationary semantics, respectively.

**Proposition 4.1.** For any non-disjunctive logic program $P$, the well-founded model $WFM(P)$ coincides with the well-founded disjunctive state-pair $WFDS(P)$ in the sense:

$$WFM(P) = L(WFDS(P)),$$

where $L(WFDH(P))$ is the state-pair consisting of only non-disjunctive literals in $WFDS(P)$.

**Proof.** From Proposition 3.2 in Section 3 and Theorem 6 in Ref. [17], it is easy to see the conclusion of the proposition holds. ☐
Proposition 4.2. For any non-disjunctive program $P$, the least stationary model (least partial stable model) $LSM(P)$ coincides with the well-founded disjunctive state-pair $WFDS(P)$.

Proof. From the above Proposition 4.1 and Corollary 19 in Ref. [33], it follows that $L(LSM(P)) = L(WFDS(P))$, therefore, $LSM(P) = WFDS(P)$. 

For any positive disjunctive program $P$, its WFDH does not only exist, but also can be obtained by only one step iteration of $A_P$ from $\emptyset$. This result provides a simple characterization for the WFDH of positive disjunctive programs and will also be used in proving the main Theorem 4.2 of this section.

Proposition 4.3. For any positive disjunctive program $P$, its skeptical semantics WFDH is determined by all assumptions that are admissible with respect to the trivial hypothesis:

$$WFDH(P) = A_P(\emptyset).$$

In general, negative information is not explicitly represented in databases and thus a meta-rule is often employed to derive negative information from deductive databases. Reiter’s [34] closed world assumption (CWA) provides such an excellent mechanism for non-disjunctive databases. As first observed by Minker [28], CWA becomes inconsistency for disjunctive programs and, thus, the GCWA for positive disjunctive programs is proposed for inferring negative information in disjunctive deductive databases. However, an important deficiency of GCWA is that it is unable to infer disjunctions of negative literals. For this motivation, GCWA is generalized to the extended generalized closed world assumption (EGCWA) [45], which has now become one of the most important non-monotonic mechanisms in deductive databases. We first review the model-theoretic definitions of GCWA and EGCWA.

Definition 4.1 [27]. Let $P$ be a positive disjunctive program, then

$$GCWA(P) = \{ \sim a : a \in B_P, P \models_{\text{min}} \sim a \},$$

$$EGCWA(P) = \{ \beta \in DB_P^- : P \models_{\text{min}} \beta \},$$

where $P \models_{\text{min}} \beta$ means that $\beta$ is satisfied by every minimal model of $P$.

The following theorem may be one of the most important results in this paper and it asserts that EGCWA coincides with WFDH for the class of positive disjunctive programs.

Theorem 4.2. For any positive disjunctive program $P$, the following holds:

$$EGCWA(P) = WFDH(P).$$

Now we present three corollaries that can be directly obtained from Theorem 4.2 and the results in Ref. [45]. Firstly, since EGCWA is consistent, then WFDH is also consistent.
Corollary 4.1. For any positive disjunctive program \( P \), its WFDH is consistent.

From Definition 4.1, it is obvious that \( GCWA(P) \) consists of all negative literals in \( EGCWA(P) \). Thus the generalized closed world assumption (GCWA) can also be characterized by WFDH.

Corollary 4.2. For any positive disjunctive program \( P \), \( GCWA(P) = L(WFDH(P)) = \{ \sim a : \sim a \in WFDH(P) \} \).

In fact, if one wants to give a direct characterization of GCWA by argumentation rather than through EGCWA, he/she can take only non-disjunctive negative literals as possible assumptions and establish an argumentation-theoretic framework similar to our DAS.

Corollary 4.3. For any positive disjunctive program \( P \), \( M \) is a minimal model of \( P \) if and only if \( M \) is a minimal model of \( P \cup WFDH(P) \).

The WFDH provides a natural and suitable generalization for EGCWA. The problem of extending (E)GCWA and the well-founded model (WFM) to the class of general disjunctive programs is pursued by many researchers, such as Refs. [4,37,35] etc. However, in our opinion, most of the generalizations of (E)GCWA and WFM are not so intuitive and simple as our WFDH. GDWFS in Ref. [4] is one of the earlier attempts to generalize WFM to the class of disjunctive logic programs but an often-mentioned deficiency of this semantics is that it interprets some normal disjunctive programs into positive programs. For example, under GDWFS, disjunctive program \( P = \{ a | b \leftarrow ; c \leftarrow \sim a ; c \leftarrow \sim b \} \) is equivalent to disjunctive program \( P = \{ a | b \leftarrow ; b | c \leftarrow ; c | a \leftarrow \} \). By employing the stable models, Sakama [37] defined an extension of GCWA, called GCWA^+. This generalization is incomplete since some disjunctive programs do not have stable models. By employing two efficient disjunctive logic program systems \( DisLog \) [39] and \( dlv \) [25], we have tested our semantics and others with many disjunctive programs, and the test results also demonstrate that DAS is a suitable semantic framework. Recently, Brass and Dix [9,10] proposed a new approach D-WFS in which the well-founded semantics for DLP is defined as the weakest semantics that satisfies some abstract properties. In particular, this semantics also provides an abstract extension of both WFM and GCWA. Though WFDH and D-WFS have quite different intuitions, it is quite possible that these two semantics coincide. In fact, we are currently working on clarifying the relationship between WFDH and D-WFS.

5. Credulous argumentation with disjunctive programs

Both the disjunctive stable semantics and our PDH represent credulous reasoning in disjunctive logic programming but the former is not complete. In this section, by studying PDH and its relation to the disjunctive stable semantics we shall show that PDH is really a natural extension of the disjunctive stable semantics. A preliminary result about PDH and the stationary semantics is also given. To this end, we introduce a simple subclass of PDHs, called the stable PDHs. We show that the stable
PDHs and the disjunctive stable models have a one-to-one correspondence. Hence the abductive semantics PDH is not only complete but can also be considered as a natural and complete extension of the disjunctive stable semantics.

5.1. The Least Fixpoint Transformation

To simplify the proof of the results in this section, we first define a program transformation for disjunctive programs, called the least fixpoint transformation, which cannot only make our proofs simpler but also provides a canonical form for disjunctive programs with respect to various semantics, including our argumentation-theoretic semantics and the disjunctive stable semantics. The program transformation Lft is based on the idea of Dung and Kanchansut [13] and Bry [12]. It is also independently defined by Brass and Dix [8,10].

To define Lft for disjunctive programs, we first extend the notion of the Herbrand base BP to the generalized disjunctive base GDBP of a disjunctive logic program P.

GDBP is defined as the set of all negative disjunctive programs whose atoms are in BP:

\[ GDB_P = \{ a_1 \cdots a_r \leftarrow b_1, \ldots, b_s : a_i, b_j \in B_P, \quad i = 1, \ldots, r; j = 1, \ldots, s \quad \text{and} \quad r > 0, s \geq 0 \} \]

In addition, \( \leftarrow \) will denote the empty clause.

Thus, we can introduce an immediate consequence operator \( T^G_P \) for general disjunctive program \( P \), which is similar to the immediate consequence operator \( T^S_P \) for positive program \( P' \). The operator \( T^G_P \) will provide a basis for defining our program transformation Lft.

**Definition 5.1.** For any disjunctive program \( P \), the generalized consequence operator \( T^G_P : 2^{GDB_P} \rightarrow 2^{GDB_P} \) is defined as, for any \( J \subseteq GDB_P \),

\[
T^G_P(J) = \{ C \in GDB_P : \text{there exist a disjunctive clause } \alpha' \leftarrow b_1, \ldots, b_m, \\
\text{ } \quad \sim b_{m+1}, \ldots, \sim b_s \text{ and } C_1, \ldots, C_m \in GDB_P \cup \{ \leftarrow \} \text{ such that} \\
(1) \ b_i | \text{head}(C_i) \leftarrow \text{body}(C_i) \text{ is in } J \text{, for all } i = 1, \ldots, m; \\
(2) \ C \text{ is the clause } sfac(\alpha' | \text{head}(C_1)) \cdots | \text{head}(C_m)) \\
\leftarrow \text{body}(C_1), \ldots, \text{body}(C_m), \sim b_{m+1}, \ldots, \sim b_s \}. 
\]

This definition looks a little tedious at first sight. In fact, its intuition is quite simple and it defines the following form of resolution:

\[
\frac{\alpha' \leftarrow b_1, \ldots, b_m, \beta_1, \ldots, \beta_s; \ b_1 | x_1 \leftarrow \beta_1, \ldots, \beta_{11}; \ldots; b_m | x_m \leftarrow \beta_{m1}, \ldots, \beta_{mt}}{\alpha' \leftarrow x_1 \cdots x_m \leftarrow \beta_1, \ldots, \beta_{11}, \ldots, \beta_{m1}, \ldots, \beta_{mt}, \beta_1, \ldots, \beta_s},
\]

where \( \alpha ' \)’s with subscripts are positive disjunctive literals and \( \beta ' \)’s with subscripts are negative literals.
Example 5.1. Suppose that $P \hat{=} a_1 \land a_3 \land a_4; a_3 \land a_5 \land a_6$ and $J = \emptyset$. Then $T^G_P(J) = T^G_P(\emptyset) = \{a_3 | a_5 \land a_6 \}$. If $J' = T^G_P(\emptyset)$. Then $T^G_P(J') = T^G_P(T^G_P(\emptyset)) = \{a_3 | a_5 \land a_6; a_1 | a_2 | a_5 \land a_4, a_6 \}.$

Notice that $T^G_P$ is a generalization of $T^S_P$ if a disjunctive program clause $a_1 | \cdots | a_n \leftarrow$ is treated as the disjunction $a_1 | \cdots | a_n$. We can prove that $T^G_P$ possesses the least fixpoint by showing $T^G_P$ continuous.

Proposition 5.1. For any disjunctive program $P$, its generalized consequence operator $T^G_P$ is continuous and hence possesses the least fixpoint $T^G_P \uparrow \omega$.

Proof. Similar to the proof of the corresponding result of $T^S_P$, see Ref. [27].

It is obvious that the least fixpoint of $T^G_P$ does not only exist but also is computable. Since $T^G_P \uparrow \omega$ is a negative disjunctive program, $T^G_P$ results in a computable program transformation which will be defined in the next definition.

Definition 5.2. Denote $T^G_P \uparrow \omega$ as $Lft(P)$, then the mapping $Lft : P \rightarrow Lft(P)$ defines a transformation from the set of all disjunctive programs to the set of all negative disjunctive programs, and we say that $Lft(P)$ is the least fixpoint transformation of $P$.

The following lemma asserts that $Lft(P)$ has the same least model-state as $P$ and it is fundamental to prove some invariance properties of $Lft$ under various semantics for disjunctive programs.

Lemma 5.1. For any hypothesis $\Delta$ of disjunctive program $P$, $(Lft(P)_{\Delta}^+)$ possesses the same least model-state as $P_{\Delta}^+$:

$ms(Lft(P)_{\Delta}^+) = ms(P_{\Delta}^+)$.

Firstly, we show that the program transformation $Lft(P)$ preserves our abductive semantics.

Proposition 5.2. For any disjunctive program $P$, $P$ is equivalent to its least fixpoint transformation $Lft(P)$ with respect to DAS. As a result, $Lft(P)$ has the same ADH (res. CDH, PDH) as $P$.

Proof. By Lemma 5.1, it follows that, for any $\alpha \in DB_p^+$ and $\Delta \in H(P)$,

$\Delta \vdash P \alpha \quad \text{if and only if} \quad \Delta \vdash Lft(P) \alpha$.

Therefore, the conclusion of the theorem is true.

The following proposition shows that the least fixpoint transformation also preserves the (disjunctive) stable models. This proposition is also independently proved by Brass and Dix [10].

For any disjunctive program $P$, and $M \subseteq B_p$. Set
If $M$ is a minimal model of $P/M$, then it is a (disjunctive) stable model of $P$. The disjunctive stable semantics of $P$ is defined as the set of its all disjunctive stable models.

**Proposition 5.3.** For any disjunctive program $P$, $P$ is equivalent to its least fixpoint transformation $\text{Lft}(P)$ with respect to the stable semantics. That is, $P$ has the same set of the stable models as $\text{Lft}(P)$.

**Proof.** Let $M \subseteq B_P$ and $\Delta_M = \| \{ \sim a : a \in B_P \setminus M \} \|$, then $P/M = P^+_\Delta$. By Lemma 5.1, $P/M$ and $\text{Lft}(P/M)$ have the same least model-state and hence have the same set of minimal models. Again, $\text{Lft}(P/M) = \text{Lft}(P)/M$. Therefore,

- $M$ is a stable model of $P$ if and only if $M$ is a minimal model of $P/M$ if and only if $M$ is a minimal model of $\text{Lft}(P)/M$ if and only if $M$ is a stable model of $\text{Lft}(P)$. □

5.2. Relation to disjunctive stable semantics

In this subsection, we first introduce a subclass of PDHs (the stable PDHs) and then, show that there is a one-to-one correspondence between the set of the stable PDHs and the set of the disjunctive stable models for any general disjunctive program.

**Definition 5.3.** A PDH $\Delta$ of disjunctive program $P$ is stable if, for any atom $a \in B_P$, either $\sim a \in \Delta$ or $\Delta \vdash_P a$.

It is easy to see that the stability of a PDH guarantees that its state-pair corresponds to a two-valued model.

In general, a PDH may not be a stable PDH. For example, consider the disjunctive program $P = \{ a | \ sim b \sim c, \ sim b ; c \sim d \}$. The unique PDH of $P$ is $\Delta = \| \sim d \|$. It is easy to see that $\Delta \vdash_P c$, but $\Delta \not\vdash_P a, b$. Thus, $\Delta$ is not stable.

The main theorem of this section can be stated as follows.

**Theorem 5.1.** Let $P$ be a general disjunctive logic program, then the following two items hold:

1. If $M$ is a stable model of $P$, then $\Delta_M = \| \{ \sim a : a \in B_P \setminus M \} \|$ is a stable PDH of $P$.
2. If $\Delta$ is a stable PDH of $P$, then $I_\Delta = \{ a \in B_P : \sim a \not\in \Delta \}$ is a stable model of $P$.

The above theorem shows that our stable PDH coincides with the stable semantics for any disjunctive programs. Thus, PDH is really a natural and complete extension for the (disjunctive) stable semantics. In addition, though we have not find an
accurate proof, we guess that all PDHs of a disjunctive program is stable when one of its PDHs is stable. If so, the form of Theorem 5.1 will be more elegant.

**Corollary 5.1.** Any (local) stratified disjunctive program $P$ has the unique stable PDH.

**Proof.** Since a stratified disjunctive program $P$ must be strongly stable, by Theorem 5.1, the stable PDH of $P$ is determined by its unique stable model (or perfect model). □

The relationship between the stationary semantics and PDH can be formulated as the following result.

**Proposition 5.4.** For any disjunctive program $P$, its stationary models coincide with the preferred disjunctive state-pairs corresponding to the stable PDHs.

**Proof.** It follows easily from Theorem 5.1 and Proposition 18 in Ref. [33]. □

5.3. An abductive procedure for PDH

In the previous sections, we have shown that WFDH generalizes both the well-founded semantics for non-disjunctive programs and EGCWA for positive disjunctive programs to the class of (general) disjunctive programs. Therefore, WFDH naturally inherits the corresponding procedural interpretations for the well-founded models and EGCWA on these classes of logic programs. It is known that the stable semantics for non-disjunctive programs has an abductive procedure (we shall call it EK-abductive procedure) [19], which naturally extends SLDNF resolution. This procedure is not only a refutation, but can also be used to compute the abductive solutions. A natural question arises: As a form of credulous reasoning for disjunctive programs, does PDH also possess a similar abductive procedure? In this subsection, we shall show that, for a useful subclass of the stratified disjunctive logic programs, PDH indeed possesses an EK-abductive procedure as that of non-disjunctive programs by exploiting a result of Dung’s [15].

Dung [15] generalizes the notion of acyclicity for non-disjunctive programs and identifies the so-called acyclic disjunctive programs, which forms a subclass of the stratified disjunctive programs.

For each disjunctive clause $C : a_1 \leftarrow \cdots \leftarrow a_r \leftarrow \text{body}(C)$, the canonical form of $C$ is defined as

$$N(C) = \{a_i \leftarrow \text{body}(C) \mid \sim a_1, \ldots, \sim a_{i-1}, \sim a_{i+1}, \ldots, \sim a_r : i = 1, \ldots, r\}.$$

The canonical form of a disjunctive program $P$ is the non-disjunctive program $N(P) = \cup\{N(C) : C \in P\}$.

**Lemma 5.2** [15]. Let $P$ be an acyclic disjunctive program and $M \subseteq B_P$. Then $M$ is a stable model of $P$ if and only if $M$ is a stable model of $N(P)$.

This lemma shows that the computation of the stable models for an acyclic disjunctive program can be transformed into the task of computing the stable models for the corresponding non-disjunctive program $N(P)$. 

Therefore, the EK-abductive procedure is sound with respect to PDH for the class of acyclic disjunctive programs.

**Theorem 5.2** (Soundness of EK-abductive procedure with respect to PDH). If $P$ is an acyclic disjunctive program and $(\leftarrow a, \emptyset), \ldots, (\square, H)$ is an EK-abductive refutation, then $H$ is an ADH. Moreover, there exists a PDH $\Delta$ such that $H \subseteq \Delta$ and $a \in M_\Delta$.

**Proof.** By Theorem 5.1 in this section and Theorem 8 in [17] about the soundness of EK-abductive procedure for non-disjunctive programs, it is easy to get the conclusion. □

In a word, some declarative semantics in our semantic framework possess corresponding procedural interpretations on some particular classes of disjunctive programs. However, it is obvious that we do not touch much on the problem of seeking tractable and/or more general algorithms for our semantics.

6. Conclusion

In this paper, we have defined an argumentation-theoretic framework DAS for disjunctive logic programs. This semantic framework has at least the following positive important features:

1. As a non-deterministic disjunctive semantics, DAS provides a semantic framework for performing abduction (argumentation) in disjunctive logic programming and disjunctive deductive databases. To our best knowledge, this work is also a first serious attempt to establish an argumentation-theoretic framework for disjunctive logic programming, in which various forms of argumentation (abduction) can be performed. Based on three-valued autoepistemic logics, the work in Refs. [5,32,33] has made attempts to embed different semantics for disjunctive logic programs into a unifying semantic framework. Since our DAS is established on the intuition of argumentation, it seems simpler and more intuitive than most of existing semantic framework for disjunctive logic programming.

2. DAS integrates many key semantics for disjunctive programs into a unifying framework, such as the well-founded semantics, the disjunctive stable semantics, the minimal model semantics, EGCWA and GCWA. Among many results obtained in this paper, Theorem 4.2 is a quite useful and interesting result since, for the first time, it shows an argumentation-theoretic characterization for the most useful non-monotonic mechanism EGCWA (including GCWA) in deductive databases. As a result, WFDH also provides a new way for performing argumentation and abduction in disjunctive logic programming. Another important result in this paper is Theorem 5.1, which provides a one-to-one correspondence between the set of the stable PDHs and the set of the stable models for any disjunctive program.

3. Shortcomings of some key semantics for disjunctive programs, which are often criticized in literature, are successfully overcome in our DAS. It is well-known that the (disjunctive) stable semantics is not complete (some disjunctive programs have no stable models); the EGCWA (GCWA) is defined only on the class of positive disjunctive programs; the well-founded semantics is defined only on non-disjunctive programs and cannot derive anything from some logic programs. In addition, most
extensions of the EGCWA and well-founded semantics are unintuitive and tedious. As noted before, DAS is a non-deterministic semantics and it can perform both skeptical and credulous reasoning. Another important feature of DAS is its completeness (or, robust), that is, each semantics introduced in DAS is defined for all disjunctive programs (of course, except for the stable PDHs). The results in this paper also show that, in a unifying framework, PDH and WFDH naturally extends two key kinds of (skeptical and credulous) semantics for disjunctive logic programming: (1) WFDH generalizes both the well-founded semantics and EGCWA (GCWA) to the whole class of disjunctive logic programs; (2) PDH extends the (disjunctive) stable semantics to the whole class of disjunctive logic programs.

Recently, the relationship between consistency-based abduction and disjunctive logic programming has been discussed by some authors [3,11]. A work that is most related to ours is the approach of [6], which aims at providing an argumentation-theoretic framework for general non-monotonic reasoning. In particular, this work discusses many kinds of attacks among hypotheses and their relations to various non-monotonic formalizations. Another related approach is the abstract framework for argumentation in Ref. [18]. These two approaches pay little attention to argumentation with disjunctive logic programs. However, our framework can be considered as a realization of their work in the setting of disjunctive logic programming.

We plan to expand our work in three directions:

1. As one referee suggested, the relationship between argumentation (abduction) and extended disjunctive programs should be investigated. The fundamental idea in this paper, however, cannot be directly generalized when the classical negation is taken into consideration. So we are working in establishing a new argumentation-theoretic framework for extended disjunctive programs.

2. More general and more efficient procedures for our semantics should be found. One possibility is to extend the abductive procedures for non-disjunctive programs by allowing disjunctions. This is a very important problem for knowledge representation in DLP but, obviously, it is also a difficult one. Dung [16] generalized the well-known Eshghi–Kowalski procedure to DLP and it is proved that this procedure computes the regular extension semantics in Ref. [46]. In addition, the complexity of our semantics should be explored.

3. Though some preliminary results are shown in Sections 4 and 5, the deep relation of our semantics to some other semantics for disjunctive programs, such as the stationary semantics [32] and the possible model semantics [36], is another direction of research.

Acknowledgements

The author would like to thank Huowang Chen, Jürgen Dix, Guoding Hu, Fangzhen Lin, Ji Wang and Lizhu Zhou for useful suggestions/discussions. Special thanks also to Michael Gelfond and the three anonymous referees for comments which helped to significantly improve the quality of this paper. This work was supported in part by the Natural Science Foundation of China (No. 69883008), the National Foundation Research Project of China (No. 61999032704), and the IT College of Tsinghua University.
Appendix A

In this appendix, we shall give some technical results and prove the theorems stated in the earlier sections.

Theorem 3.1. For any self-consistent hypothesis $\Delta$ of disjunctive program $P$, $\Delta$ is an ADH of $P$ if and only if $\Delta \rightarrow_p \Delta'$ for any hypothesis $\Delta'$ of $P$ satisfying $\Delta' \rightarrow_p \Delta$.

Proof. ($\Leftarrow$) : To show that $\Delta$ is admissible in the sense of Definition 2.9. Because $\Delta$ is self-consistent, we need only to prove that $\Delta \subseteq A_P(\Delta)$. For any $\beta = \sim b_1 | \cdots | \sim b_m \in \Delta$, if $\Delta'$ is a hypothesis of $P$ such that $\Delta' \vdash_p b_i$ for all $i = 1, \ldots, m$, then $\Delta' \rightarrow_p \Delta$ and thus $\Delta \rightarrow_p \Delta'$ by the assumption of the theorem. This means that $\beta \in A_P(\Delta)$, that is, $\Delta \subseteq A_P(\Delta)$. Hence $\Delta$ is an ADH.

(⇒) Suppose that $\Delta$ is an ADH of $P$. If $\Delta'$ is any hypothesis of $P$ such that $\Delta' \rightarrow_p \Delta$, then there are two possible cases:

Case 1. There exists a disjunction of negative literals $\sim b_1 | \cdots | \sim b_m \in \Delta$ such that $\Delta' \vdash_p b_i$, for all $i = 1, \ldots, m$; or

Case 2. There exist negative literals $\sim b_1, \ldots, \sim b_m \in \Delta$ such that $\Delta' \vdash_p b_1 | \cdots | b_m$. If Case 1 holds, since $\sim b_1 | \cdots | \sim b_m \in \Delta \subseteq A_P(\Delta)$, it is obvious that $\Delta \rightarrow_p \Delta'$; for Case 2, $\| \Delta' \cup \{ \sim b_1, \ldots, \sim b_m \} \| \vdash_p b_1$ and $\sim b_1 \in \Delta$ is admissible with respect to $\Delta$. Thus $\Delta \rightarrow_p \Delta''$, where $\Delta'' = \| \Delta' \cup \{ \sim b_2, \ldots, \sim b_m \} \|$. This case can be again divided into two subcases:

Subcase 1. There exists $\sim c_1 | \cdots | \sim c_n \in \Delta''$ such that $\Delta \vdash_p c_i$, for all $i = 1, \ldots, n$: Suppose that there exists $i(1 \leq i \leq n)$ such that $\sim c_i \in \{ \sim b_2, \ldots, \sim b_m \}$ and $m \geq 2$, then $\sim c_i \in \Delta$. On the other hand, $\Delta \vdash_p c_i$, we have $\Delta \rightarrow_p \Delta$. This is impossible. Therefore, $\{ \sim c_1, \ldots, \sim c_n \} \cap \{ \sim b_2, \ldots, \sim b_m \} = 0$. It is the case that $\sim c_1 | \cdots | \sim c_n \in \Delta'$ and $\Delta \rightarrow_p \Delta'$.

Subcase 2. There exist $\sim c_1, \ldots, \sim c_n \in \Delta''$ such that $\Delta \vdash_p c_1 | \cdots | c_n$, let $\{ \sim c_1, \ldots, \sim c_t \} \cap \{ \sim b_2, \ldots, \sim b_m \} = 0$, and $\{ \sim c_{t+1}, \ldots, \sim c_n \} \subseteq \{ \sim b_2, \ldots, \sim b_m \} \subseteq \Delta$, $0 \leq t \leq n$. We can assert that $t \neq 0$: otherwise, if $t = 0$, then $\{ \sim c_1, \ldots, \sim c_t \} \subseteq \{ \sim b_2, \ldots, \sim b_m \} \subseteq \Delta$, this contradicts the self-consistency of $\Delta$. Hence, $1 \leq t \leq n$. We have $\sim c_1, \ldots, \sim c_t \in \Delta'$ and $\Delta \vdash_p c_1 | \cdots | c_t$, this also means that $\Delta \rightarrow_p \Delta$. Therefore, in any case, we have $\Delta \rightarrow_p \Delta'$ whenever $\Delta' \rightarrow_p \Delta$. □

Proposition 3.1. If $\Delta_1, \Delta_2, \ldots, \Delta_n, \ldots$ is a sequence of admissible hypotheses of disjunctive program $P$ such that $\Delta_n \subseteq \Delta_{n+1}$ for any $n > 0$, then the hypothesis $\Delta = \bigcup_{n=1}^{\infty} \Delta_n$ is an ADH of $P$.

Proof. If $\alpha \in \Delta$, then there exists $n \geq 1$ such that $\alpha \in \Delta_n$. Because $\Delta_n$ is admissible, then $\alpha \in \Delta_n \subseteq A_P(\Delta_n) \subseteq A_P(\Delta)$ and thus $\alpha \in A_P(\Delta)$. It remains to prove that $\Delta$ is self-consistent: To the contrary, suppose that $\Delta \rightarrow_p \Delta$. By Corollary 2.1, there exists $\alpha = \sim a_1 | \cdots | \sim a_r \in \Delta$ such that $\Delta \vdash_p a_j$ for all $j = 1, \ldots, r$. Since each $a_j$ is derived by a finite number of resolutions from $P^{+}_\Delta$ and $\Delta$, we can choose the number $n$ is large enough such that $\Delta_n \vdash_p a_j$, for all $j = 1, \ldots, r$ and $\alpha \in \Delta_n$. Therefore, $\Delta_n \rightarrow_p \Delta_n$, a contradiction. Thus $\Delta$ is self-consistent. □
If $\Delta \vdash \Delta \cup \{a\} \not\vdash a$ and $a \in A_P(\Delta) \setminus \Delta$, then $\Delta \not\vdash_P a$.

**Proof.** To the contrary, suppose that $\Delta \vdash_P a$. From $\sim a \in A_P(\Delta)$, then it follows that $\Delta \not\vdash_P \Delta'$. We distinguish two cases:

**Case 1.** There exist negative literals $\sim b_1, \ldots, \sim b_m \in \Delta \cup \{a\}$ such that $\Delta \vdash_P b_1 \cdots b_m$. Since $\Delta$ is self-consistent, there exists at least one $i (1 \leq i \leq m)$ such that $\sim b_i \sim a$. Without loss of generality, suppose that $\sim b_1, \ldots, \sim b_{m-1} \in \Delta$ and $\sim b_m \sim a$. Then, by $\Delta \vdash_P b_1 \cdots |b_m|a$, we have $\Delta \vdash_P a$ and $\sim a \in A_P(\Delta)$. Hence $\Delta \not\vdash_P \Delta$, a contradiction.

**Case 2.** There exists a disjunction $\beta = \sim b_1 \cdots |b_m \in \Delta \cup \{a\}$ such that $m > 1$ and $\Delta \vdash_P b_j$, for all $j = 1, \ldots, m$. $\beta \not\vdash_P \{a\}$ implies $\beta \in \Delta$. Therefore, $\Delta \not\vdash_P \Delta$, a contradiction.

Thus, in any case, we have that $\Delta \not\vdash_P a$. \qed

**Theorem 3.2** (Fundamental property of ADHs). For any ADH $\Delta$ of disjunctive program $P$, if $z \in D_B P$ is admissible w.r.t. $\Delta$, that is, $z \in A_P(\Delta)$, then $\Delta = \{z\}$ is also an ADH of $P$.

**Proof.** By Corollary 2.2, $\Delta' = \{z\} \subseteq A_P(\Delta) \subseteq A_P(\Delta')$. It is enough to show that $\Delta'$ is self-consistent. We consider two cases:

**Case 1.** $z = a_1 \cdots a_n$ and $m > 1$: To the contrary, suppose that $\Delta' \not\vdash_P \Delta'$, by Corollary 2.1, there exists $\gamma = c_1 \cdots c_n \in \Delta'$ such that $\Delta' \vdash_P c_j$, for all $j = 1, \ldots, n$. By $m > 1$, we have $P_N^+ = P_A^+$ and thus $\Delta \vdash_P c_j$, for all $j = 1, \ldots, n$. Since $\Delta$ is self-consistent, it has to be the case that $\gamma \in \{z\}$. From Corollary 2.2(1) and $z \in A_P(\Delta)$, it follows that $\gamma \in A_P(\Delta)$. Therefore, $\Delta' \not\vdash_P \Delta'$, a contradiction. That is, $\Delta'$ is self-consistent.

**Case 2.** $z = \sim a$: To the contrary, suppose that $\Delta' \not\vdash_P \Delta'$. Then there are two subcases:

**Subcase 1:** There exists $\gamma = c_1 \cdots c_n \in \Delta'$ such that $\Delta' \vdash_P c_i$, for all $i = 1, \ldots, n$.

Consider two sub-subcases: (1) If $\gamma \in \Delta$, then $\Delta' \not\vdash_P \Delta$. Since $\Delta$ is an ADH of $P$, it follows from Theorem 3.1 that $\Delta \not\vdash_P \Delta'$, and hence $\Delta \not\vdash_P a$, a contradiction. (2) If $\gamma \notin \Delta$, that is, $\gamma \in \{a\}$. Then there exists $i (1 \leq i \leq n)$ such that $\sim c_i \sim a$. Thus $\Delta \not\vdash_P a$, also a contradiction.

**Subcase 2:** There exist $c_1, \ldots, c_n \in \Delta'$ such that $\Delta' \vdash_P c_1 \cdots c_n$: Because $\Delta$ is self-consistent, there exist $i (1 \leq i \leq n)$, such that $\sim c_i \sim a$. Without loss of generality, suppose that $i = n$ and $\sim c_1, \ldots, c_{n-1} \in \Delta$. From $\Delta' \vdash_P c_1 \cdots c_n \vdash_P a$ and $\sim c_1, \ldots, c_{n-1} \in \Delta$, we have $\Delta' \vdash_P a$, contradiction to Lemma A. Therefore, $\Delta'$ is self-consistent. \qed

**Corollary 3.3.** If $\Delta$ is a CDH of non-disjunctive program $P$, then $\Delta$ is non-disjunctive. That is, every CDH of a non-disjunctive program is non-disjunctive.

**Proof.** Without loss of generality, it is enough to show that: $\sim a_1 \sim b \in \Delta = A_P(\Delta)$ implies either $\sim a \in \Delta$ or $\sim b \in \Delta$. On the contrary, suppose that $\sim a, \sim b \notin \Delta = A_P(\Delta)$, then there exist non-disjunctive hypotheses $\Delta_a$ and $\Delta_b$ of $P$ such that
Lemma B. For positive disjunctive program \( P \), if \( A_P(\emptyset) \vdash \vdash b_1 \cdots b_m, m \geq 1 \), then \( \emptyset \vdash b_1 \cdots b_m \).

Proof. By the assumption, there is a disjunction \( \beta = b_1 \cdots | b_m | b_{m+1} \cdots | b_n \in \text{can}(ms(P)) \) such that \( n \geq m \) and \( \sim b_{m+1}, \ldots, \sim b_n \in A_P(\emptyset) \). We can assert that \( n = m \). Otherwise, \( n > m \), set \( \Delta'' = \| \sim b_1, \ldots, \sim b_m, \sim b_{m+2}, \ldots, \sim b_n \| \), then \( \Delta'' \vdash b_{m+1} \). Again, from \( \beta \in ms(P) \), we have \( b_1 | b_m | b_{m+2} \cdots | b_n \notin \text{can}(ms(P)) \). Therefore, \( \emptyset \not\vdash \Delta'' \), this contradicts \( \sim b_{m+1} \in A_P(\emptyset) \). \( \square \)

Proof of Proposition 4.3. It suffices to show \( A_P(A_P(\emptyset)) = A_P(\emptyset) \). By the monotonicity of \( A_P \), it follows easily that \( A_P(\emptyset) \subseteq A_P(A_P(\emptyset)) \). For the converse, suppose that \( x \in A_P(A_P(\emptyset)) \). If \( \Delta' \) is any hypothesis of \( P \) such that \( \Delta' \) denies \( x \), then \( A_P(\emptyset) \vdash \vdash P \Delta' \). To prove that \( x \) is admissible wrt \( \emptyset \), it is enough to show \( \emptyset \vdash \vdash P \Delta' \). We still consider two cases:

Case 1. There exists a hypothesis \( \sim b_1 | \cdots | \sim b_m \) such that \( A_P(\emptyset) \vdash \vdash b_1, \ldots, b_m \). By Lemma B, \( \emptyset \vdash \vdash P \Delta' \).

Case 2. There exist \( \sim b_1, \ldots, \sim b_m \in \Delta' \) such that \( A_P(\emptyset) \vdash \vdash b_1, \ldots, b_m \). Similar to Case 1, \( \emptyset \vdash \vdash P \Delta' \). Hence \( x \in A_P(\emptyset) \). That is, \( A_P(A_P(\emptyset)) \subseteq A_P(\emptyset) \). \( \square \)

Theorem 4.2. For any positive disjunctive program \( P \), EGCWA(P) = WFDH(P).

Proof. First, to prove EGCWA(P) \( \subseteq \) WFDH(P): It suffices to show that, for any assumption \( \beta = \sim b_1 | \cdots | \sim b_m \), \( \beta \notin \text{WFDH}(P) \) implies that \( \beta \notin \text{EGCWA}(P) \), where \( b_1, \ldots, b_m \) are distinct from each other. If \( \beta \notin \text{WFDH}(P) \), then \( \beta \notin A_P(\emptyset) \) by Proposition 4.3, thus, for some hypothesis \( \Delta' \) of \( P \), \( \Delta' \vdash \vdash P b_i \) for all \( i = 1, \ldots, m \), but \( \emptyset \not\vdash \vdash P \Delta' \). Hence there exists at least one \( \alpha_i \in \text{can}(ms(P^+)) = \text{can}(ms(P)) \), for every \( i = 1, \ldots, m \), such that
\[
\alpha_i = b_1 | b_{11} | \cdots | b_{1t_1}, \quad t_1 \geq 0,
\]
\[
\vdots
\]
\[
\alpha_m = b_m | b_{m1} | \cdots | b_{mt_m}, \quad t_m \geq 0,
\]
and \( \sim b_{ij} \in \Delta', i = 1, \ldots, m; j = 1, \ldots, t_i \).

We assert that \( \Delta' \) is self-consistent. Otherwise, i.e. \( \Delta' \not\vdash \vdash P \Delta' \), then there exist \( \sim a_1, \ldots, \sim a_s \in \Delta' \) satisfying \( \Delta' \vdash \vdash P a_i | \cdots | a_s \). This implies that there is a disjunction \( a_1 | \cdots | a_{r_1} | a_{r_1+1} | \cdots | a_r \in \text{can}(ms(P)) \) such that \( \sim a_{r_1+1}, \ldots, \sim a_s \in \Delta' \). Therefore, \( \emptyset \vdash \vdash P a_i | \cdots | a_r \). This implies that \( \theta \vdash P \Delta' \), a contradiction.

Since \( \Delta' \) is self-consistent, no clause \( b_{ij} \leftarrow \) is in \( P \), that is, \( b_{ij} \notin \text{can}(ms(P)) \), and \( b_j \) does not appear in \( \alpha_i \) for any \( i \neq j \).
Now we prepare to construct a minimal model \( M \) of \( \text{can}(\text{ms}(P)) \) such that \( \{b_1, \ldots, b_m\} \subseteq M \).

For any disjunction \( x = c_1 | \cdots | c_s \), let \( \tilde{x} \) denote the disjunction obtained from \( x \) by deleting all atoms that appear in \( \Delta' \). Set \( S_1 = \{ x \in \text{can}(\text{ms}(P)) : x \text{ contains at least one atom } b_i \text{ for } i = 1, \ldots, m \} \);

\[
S_2 = \{ x \in \text{can}(\text{ms}(P)) : x \text{ contains no atom } b_i \text{ for } i = 1, \ldots, m \}.
\]

Then \( \text{can}(\text{ms}(P)) \) is divided into two disjoint parts: \( \text{can}(\text{ms}(P)) = S_1 \cup S_2 \).

Let \( \bar{S}_2 = \{ \tilde{x} : x \in S_2 \} \). We can assert that no element of \( \bar{S}_2 \) is empty disjunction. In fact, if otherwise, then there exists an \( x = c_1 | \cdots | c_s \) in \( \text{can}(\text{ms}(P)) \) such that \( \text{atoms}(\tilde{x}) \subseteq \text{atoms}(\Delta') \). This will implies that \( \emptyset \not\rightarrow_p \Delta' \). Thus every \( \tilde{x} \) in \( \bar{S}_2 \) is a non-empty disjunction of atoms. This implies that \( S_2 \) has at least one model. This also guarantees that \( \bar{S}_2 \) has at least one minimal model \( M_2 \).

Let \( M_1 = \{ b_1, \ldots, b_m \} \). We shall prove \( M = M_1 \cup M_2 \) is a minimal model of \( \text{can}(\text{ms}(P)) \). First, it is easy to see that \( M \) is a model of \( \text{can}(\text{ms}(P)) \). Next, it suffices to show that \( M \) is minimal. Suppose that \( M \) is not a minimal model of \( \text{can}(\text{ms}(P)) \). Then there exists an atom \( b \in M \) such that \( M \setminus \{ b \} \) is still a model of \( \text{can}(\text{ms}(P)) \) since \( \text{can}(\text{ms}(P)) \) consists of disjunctions of atoms. If \( b = b_i \) for some \( i = 1, \ldots, m \), then \( M \not\models \tilde{x}_i \), impossible. If \( b \in M_2 \), by the definition of \( S_2 \), \( M_2 \setminus \{ b \} \) should also be a model of \( S_2 \). This contradicts the minimality of \( M_2 \). Thus, \( M \) is a minimal model of \( \text{can}(\text{ms}(P)) \).

However, \( b_1, \ldots, b_m \in M \), this means that \( \sim b_1 | \cdots | \sim b_m \not\in \text{EGCWA}(P) \).

Next, to show that \( \text{WFDH}(P) \subseteq \text{EGCWA}(P) \): Suppose that \( \beta = \sim b_1 | \cdots | \sim b_m \not\in \text{EGCWA}(P) \), we shall prove that \( \beta \not\in \text{WFDH}(P) \). Suppose that \( \beta \not\in \text{WFDH}(P) \) means that there is a minimal model \( M \) of \( P \) such that \( \{b_1, \ldots, b_m\} \subseteq M \). For each \( i \) (\( 1 \leq i \leq m \)), there must be at least one \( x_i \in \text{can}(\text{ms}(P)) \) such that \( x_i = b_i | b_{i_1} | \cdots | b_{i_i}, t_i \geq 0 \), and \( b_{i_j} \in M, i = 1, \ldots, m; j = 1, \ldots, t_i \) (otherwise, for some \( b_j \), if every \( x \in \text{can}(\text{ms}(P)) \) satisfying \( b_j \in \text{atoms}(x) \) contains an atom in \( M \) distinct from \( b_j \), then \( M \setminus \{ b_j \} \) is still a model of \( \text{can}(\text{ms}(P)) \), contradiction to the minimality of \( M \)). Set \( \Delta' = \| \cup_{i=1}^{m} \cup_{j=1}^{t_i} \{ \sim b_{i_j} \} \| \), then

1. \( \Delta' \) denies \( \beta \): \( \Delta' \models_p b_i \) for all \( i = 1, \ldots, m \);
2. \( \Delta' \) is a self-consistent hypothesis of \( P \);
3. \( \emptyset \not\rightarrow_p \Delta' \).

It is easy to see that the above (1) holds. We now show (2): On the contrary, suppose that \( \Delta' \rightarrow_p \Delta' \). Since \( \Delta' \) is generated only by negative literals, there must exists a \( \sim b'_1 | b'_1 | \cdots | b'_s \in \Delta' \) such that \( \Delta' \models_p b'_1 \), and hence there is a disjunction of atoms \( b'_1 | b'_2 | \cdots | b'_s \in \text{can}(\text{ms}(P)) \) satisfying \( \sim b'_1, \ldots, \sim b'_s \in \Delta', s \geq 1 \). Because \( M \models b'_1 | b'_2 | \cdots | b'_s \), there is some \( b'_i \in M, 1 \leq i \leq s \). Again, \( \{ \sim b'_1, \ldots, \sim b'_s \} \subseteq \Delta' \), then, for some \( j, k (1 \leq j \leq m; 1 \leq k \leq t_j \), \( b'_j = b_{jk} \). This implies that \( x_j \) contains at least two atoms of \( M \) (\( b_j \) and \( b_{jk} \)), a contradiction.

So \( \Delta' \) is self-consistent.

(3) is an immediate result of (2): for otherwise, \( \emptyset \not\rightarrow_p \Delta' \) implies \( \Delta' \rightarrow_p \Delta' \).

From (1) and (3), \( \beta = \sim b_1 | \cdots | \sim b_m \not\in A_p(\emptyset) = \text{WFDH}(P) \).

To prove Proposition 5.2, we need some preparation. First, we observe a quite simple result.
Lemma C. Let $C: \alpha \leftarrow \text{body}(C)$ be a program clause of disjunctive program $P$ and $\alpha' \in DB^+_P$ such that $\alpha \Rightarrow \alpha'$. If $Q = P \cup \{\alpha' \leftarrow \text{body}(C)\}$, then for any hypothesis $\Delta$ of $P$,

$$ms(Q^+_\Delta) = ms(P^+_\Delta).$$

That is, if $C$ is in $P$, then for any program clause $C'$ such that $\text{body}(C') = \text{body}(C)$ and $\text{head}(C) \Rightarrow \text{head}(C')$, the adding of $C'$ to $P$ will not change the semantics in DAS. For this reason, we will occasionally say that $C'$ is a clause of $P$ even if $C'$ is in fact not in $P$.

Proof. It is easy to see that $ms(P^+_\Delta) \subseteq ms(Q^+_\Delta)$ since $P^+_\Delta \subseteq Q^+_\Delta$.

For the converse inclusion, it suffices to show that $T^3_{P^+_\Delta} \uparrow k \subseteq ms(P^+_\Delta)$ by using transfinite induction on $k$. The remaining proof is direct and easy, we omit it here. \(\square\)

Lemma 5.1. For any hypothesis $\Delta$ of disjunctive program $P$, $(\text{Lft}(P^+_\Delta))$ possesses the same least model-state as $P^+_\Delta$:

$$ms(\text{Lft}(P^+_\Delta)) = ms(P^+_\Delta).$$

Proof. First, to show that $ms(\text{Lft}(P^+_\Delta)) \subseteq ms(P^+_\Delta)$: If $\alpha = a_1 | \cdots | a_r \in \text{can}(ms(\text{Lft}(P^+_\Delta)))$, since $\text{Lft}(P)$ is a negative disjunctive program, then the trivial clause $a_1 | \cdots | a_r \leftarrow \text{in Lft}(P^+_\Delta)$ and hence there exists a negative clause $C': a_1 | \cdots | a_r \leftarrow b_1, \ldots, b_s$ belonging to $\text{Lft}(P)$ such that $\sim b_1, \ldots, b_s \in \Delta$. Then $\text{Lft}(P) = T^2_{P^+_\Delta} \uparrow \omega$, using induction on $k$, we prove that $a_1 | \cdots | a_r \leftarrow \text{in Lft}(P^+_\Delta)$ holds for any $C'$ in $T^2_{P^+_\Delta} \uparrow k: a_1 | \cdots | a_r \leftarrow b_1, \ldots, b_s$.

If $C' \in T^2_{P^+_\Delta} \uparrow 1$, then the clause $a_1 | \cdots | a_r \leftarrow b_1, \ldots, b_s$ is in $P$. This means that the positive program clause $a_1 | \cdots | a_r \leftarrow \text{in} P^+_\Delta$ and therefore $a_1 | \cdots | a_r \in ms(P^+_\Delta)$.

Assume that $C' \in T^2_{P^+_\Delta} \uparrow k$ implies $a_1 | \cdots | a_r \in ms(P^+_\Delta)$. If $C' \in T^2_{P^+_\Delta} \uparrow k + 1 = T^2_{P^+_\Delta}(T^2_{P^+_\Delta} \uparrow k)$, then there exists a disjunctive clause $C': \alpha \leftarrow b_1, \ldots, b_m, \sim b'_1, \ldots, \sim b'_n$ in $P$ and $C'_1, \ldots, C'_m \in GDB_P \cup \{\langle\rangle\}$ satisfying the following two conditions:

1. For any $i = 1, \ldots, m$, the clause $b_i | \text{head}(C'_1) \leftarrow \text{body}(C'_1)$ is in $T^2_{P^+_\Delta} \uparrow k$;
2. $C'$ is the clause $s\text{fac}(\alpha | \text{head}(C'_1) | \cdots | \text{head}(C'_m)) \leftarrow b'_1, \ldots, b'_n, \text{body}(C'_1), \ldots, \text{body}(C'_m)$.

Since $\{\sim b'_1, \ldots, \sim b'_n\} \subseteq \text{body}(C') \subseteq \Delta$, corresponding to $C$, the positive clause $C^+_\Delta: \alpha \leftarrow b_1, \ldots, b_m$ is in $P^+_\Delta$. By the induction assumption, we have $b_i | \text{head}(C'_1) \in ms(P^+_\Delta)$ for all $i = 1, \ldots, m$. Again, from (2),

$$a_1 | \cdots | a_r = s\text{fac}(\alpha | \text{head}(C'_1) | \cdots | \text{head}(C'_m)) \in T^2_{P^+_\Delta}(ms(P^+_\Delta)) \subseteq ms(P^+_\Delta).$$

Hence $\text{can}(ms(\text{Lft}(P^+_\Delta))) \subseteq ms(P^+_\Delta)$. That is, $ms(\text{Lft}(P^+_\Delta)) \subseteq ms(P^+_\Delta)$.

Next, to prove $ms(P^+_\Delta) \subseteq ms(\text{Lft}(P^+_\Delta))$.

Since $ms(P^+_\Delta) = ||T^2_{P^+_\Delta} \uparrow \omega||$, by using induction on $k$, it suffices to prove that $T^2_{P^+_\Delta} \uparrow k \subseteq ms(\text{Lft}(P^+_\Delta))$. \(\diamondsuit\)
If \( k = 0, T^S_{P^+_\Delta} \uparrow k = \emptyset, (\star) \) is trivial. Assume that (\( \star \)) holds for \( k \). We need to show that

\[
T^S_{P^+\Delta} \uparrow k + 1 \subseteq ms(Lft(P^+_{\Delta})).
\]

In fact, if \( a_1|\cdots|a_r \in T^S_{P^+_\Delta} \uparrow k + 1 = T^S_{P^+\Delta}(T^S_{P^+\Delta} \uparrow k) \), then there is a positive disjunctive clause \( C^+_{\Delta} \) of \( P^+ \), \( \neg x \leftarrow b_1, \ldots, b_m, \) and \( \hat{x}_1, \ldots, \hat{x}_m \in DB^+_P \) such that the following two conditions are satisfied:

1. \( b_i|x_i \in T^S_{P^+_\Delta} \uparrow k \), for all \( i = 1, \ldots, m \); and
2. \( a_1|\cdots|a_r = sfac(x'|x_1| \cdots |x_m) \).

By induction assumption, \( b_1|x_i \in ms(Lft(P^+_{\Delta})) \). Corresponding to \( C^+_{\Delta} \), there is a disjunctive clause \( \neg x \leftarrow b_1, \ldots, b_m, \) \( \sim b_1', \ldots, \sim b_s' \) in \( P^+ \) such that \( \sim b_1', \ldots, \sim b_s' \in \Delta \).

For any \( i = 1, \ldots, m \), by Lemma C, we can take for granted that the positive clause \( b_i|x_i \leftarrow \) is in \( Lft(P^+_{\Delta}) \), then there is a negative disjunctive clause \( b_i|x_i \leftarrow b_1, \ldots, \sim b_i, \sim b_i', \ldots, \sim b_s' \) in \( Lft(P) \) such that \( \sim b_1, \ldots, \sim b_i, \sim b_s' \in \Delta \). By the definition of \( T^G_{P^+} \), we have \( x'|x_1| \cdots |x_m \leftarrow b_1, \ldots, \sim b_i, \sim b_1', \ldots, \sim b_i', \ldots, \sim b_s, \sim b_1', \ldots, \sim b_s', \sim b_m, \ldots, \sim b_s' \) is in \( T^G_{P}(Lft(P)) = Lft(P) \). It follows from \( body(C') \subseteq \Delta \) that \( a_1|\cdots|a_r \leftarrow \) belongs to \( Lft(P^+_{\Delta}) \), or \( a_1|\cdots|a_r \in ms(Lft(P^+_{\Delta})) \). Thus, \( ms(P^+_{\Delta}) \subseteq ms(Lft(P^+_{\Delta})) \). \( \Box \)

Before proving Theorem 5.1, we need the following lemma.

**Lemma 5.2.** If \( M \) is a stable model of negative disjunctive program \( P \), then \( \Delta_M = \{ \sim a : a \in B_P \backslash M \} \) is an ADH of \( P \).

**Proof.** To the contrary, suppose that \( \Delta_M \) is not self-consistent, i.e. \( \Delta_M \not\rightarrow_P \Delta_M \). Since \( can(\Delta_M) \) consists of only negative literals, there are \( \sim a_1, \ldots, \sim a_r \in \Delta_M \) such that \( a_1|\cdots|a_r \in ms(P^+_{\Delta_M}) \), this contradicts the fact that \( M \models P_{\Delta_M} \). Therefore, \( \Delta_M \) is self-consistent. Next, to show that \( \Delta_M \subseteq A_P(\Delta_M) \): If \( \sim a \in \Delta_M \), then \( a \notin M \). For any hypothesis \( \Lambda \) of \( P \) such that \( \Lambda \vdash_P a \), there is a negative disjunctive clause \( C \) of \( P \) such that \( a|a_1|\cdots|a_r \leftarrow b_1, \ldots, b_m \) satisfying \( \sim a_i, \sim b_j \in \Lambda \), for all \( i = 1, \ldots, r \) and \( j = 1, \ldots, m \). Because \( a \notin M \) and \( M \models C \), there are two possible cases: (1) \( b_j \in M \) for some \( j \); (2) \( a_i \in M \) for some \( i \). Thus, in any case, there is \( c \in M \) such that \( \sim c \in \Lambda \). Since \( M \) is a minimal model of \( P/M = P^+_{\Delta_M} \), there is a clause of \( P^+_{\Delta_M} \) such that \( \sim c_1, \ldots, \sim c_n \in \Delta_M \). Hence \( \Delta_M \vdash_P c \), this implies \( \Delta_M \not\rightarrow_P \Lambda \). That is, \( \sim a \in A_P(\Delta_M) \), or \( can(\Delta_M) \subseteq A_P(\Delta_M) \). Thus, \( \Delta_M \subseteq A_P(\Delta_M) \). \( \Box \)

**Theorem 5.1.** Let \( P \) be a disjunctive program, then

1. If \( M \) is a stable model of \( P \), then \( \Delta_M = \{ \sim a : a \in B_P \backslash M \} \) is a stable PDH of \( P \).
2. If \( \Delta \) is a stable PDH of \( P \), then \( I_\Delta = \{ a \in B_P : \sim a \notin \Delta \} \) is a stable model of \( P \).

**Proof.** By Lemma 5.1, it suffices to prove this theorem only for the class of negative disjunctive programs. Thus, without loss of generality, we can suppose that \( P \) is a negative disjunctive program.1. It follows from Lemma 5.2 that \( \Delta_M \) is an ADH of \( P \). Thus, it remains to show that \( \Delta_M \) is maximal: Let \( \Delta \) be a ADH of \( P \) such that \( \Delta_M \subseteq \Delta \). Then, there is an assumption \( \beta = \sim a_1|\cdots|\sim a_r \in \Delta \) but \( \beta \notin \Delta_M \) \( (r > 0) \).

This implies that \( \sim a_1, \ldots, \sim a_r \notin \Delta_M \) and thus \( a_1, \ldots, a_r \in M \). Since \( M \) is a minimal model of \( P/M = P^+_{\Delta_M} \), it is easy to see that \( \Delta_M \vdash_P a_i \) for \( i = 1, \ldots, r \). That is, \( \Delta_M \not\rightarrow_P \Delta \). Hence \( \Delta \not\rightarrow_P \Delta \), a contradiction. Thus, \( \Delta \) is a maximal element of the set of all ADHs.
Finally, it is obvious that $\Delta_M$ is stable. We first prove that $I_\Delta$ is a model of $P/I_\Delta$: Otherwise, there would be a clause $C^+ : b_1 \cdots b_m \leftarrow$ in $P/I_\Delta = P_\Delta^+$ such that $b_j \not\in I_\Delta$ for $i = 1, \ldots, m$. This implies that there exists a clause $C : b_1 \cdots b_m \leftarrow b_{m+1}, \ldots, b_n$ such that $b_j \in \Delta$ for $j = m+1, \ldots, n$ ($n \geq m > 0$). Hence $\Delta$ is not self-consistent, a contradiction. That is, $I_\Delta$ is a model of $P/I_\Delta$. Secondly, it is enough to show that $I_\Delta$ is a minimal element of the set of all models of $P/I_\Delta$: Suppose that $N$ is model of $P/I_\Delta$ such that $N \not\subset I_\Delta$. Then there exists $a \not\in N$ such that $a \not\in I_\Delta$. Since $\sim a \not\in \Delta$ and $\Delta$ is stable, we have $\Delta \models \neg a$, which implies $\Delta \models \neg P \models a$. Because $\Delta \subset \Delta_N$, it is the case $\Delta_N \models \neg P \models a$. Thus, $N$ cannot be a model of $P_\Delta^+$, contradiction. □

References


