Abdiction in Annotated Logic Programming

Krishnaprasad Thirunarayan

Wright State University - Main Campus, t.k.prasad@wright.edu

Follow this and additional works at: http://corescholar.libraries.wright.edu/knoesis

Part of the Bioinformatics Commons, Communication Technology and New Media Commons, Databases and Information Systems Commons, OS and Networks Commons, and the Science and Technology Studies Commons

Repository Citation

http://corescholar.libraries.wright.edu/knoesis/864

This Conference Proceeding is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu.
Abduction in Annotated Logic Programming

Krishnaprasad Thirunarayan *
Department of Computer Science and Engineering,
Wright State University,
Dayton, Ohio 45435.

1 Introduction

Pragmatically, the logic programming paradigm presents a reasonable trade-off between expressive power and computational efficiency [9, 14]. Here, we investigate techniques to make it more expressive for knowledge representation, while simultaneously retaining the computational advantages of efficiency and simplicity. In Section 1.1, we review annotated logic programs. In Section 1.2, we motivate the integration of deductive and abductive reasoning. In Section 2, we present an informal description of the proposed theory of abductive reasoning for annotated logic programs through examples. In Section 3, we describe the detailed syntax and develop a model-theoretic semantics of annotated logic programs by amalgamating concepts from logic programming [1] and multi-valued logics [8]. We identify a class of annotated logic programs called the stratified programs which can be given a unique minimal supported Herbrand model as their meaning [21]. We then show how to integrate abductive reasoning into this annotated logic framework by generalizing the work described in [18, 19]. We formalize the notion of an explanation, and specify when an explanation can be regarded as acceptable. Finally, we conclude in Section 5.

1.1 Annotated Logic Programs

[2, 3] pursued a four-valued logic approach as an alternative to classical predicate calculus to localize the effect of a contradiction in a first-order language. [24, 25, 7] used similar ideas to provide semantics to strict inheritance hierarchies and logic programs. [8] generalized it to multi-valued logics to make precise the book-keeping operations performed by a non-monotonic reasoning system that handles both strict and defeasible information. In all the above cases, the semantics is given by mapping literals to truth values that are ordered on two different scales — the truth scale and the information scale. For example, Tweety being a bird contributes a supporting evidence to its flying ability, while Tweety being a penguin contributes a defeating evidence. This can be represented by mapping fly(Tweety) to the constants +bird and -penguin respectively. Furthermore, knowing that Tweety is a penguin is more informative than knowing that it is a bird. This can be represented by letting the constant penguin have higher information-content than the constant bird.

In approaches described in [10, 11, 12, 13], the underlying language is extended using annotations, which can be thought of as abbreviated justifications or pieces of evidence. Some areas of application are rule-based expert systems with uncertainty [10], temporal reasoning [12], reasoning with inconsistency [4, 11], etc. In [13], we use them to represent nonmonotonic multiple inheritance networks.

1.2 Integrating deductive and abductive reasoning

In a number of areas such as medical diagnosis [16, 26], fault diagnosis [22, 18], temporal reasoning [23], recognition [19], planning [6], natural language processing [5], etc. a large fragment of relevant knowledge can be expressed by specifying the “causal” relationships that exist among the various entities. In such domains the two basic operations of interest for information retrieval are:

- Given a set of causes, determine their cumulative effect.
- Given a set of observations, hypothesize a set of possible causes that can explain the observa-

*Work supported in part by NSF grant IRI-9009587.
Reasoning from causes to effects is predominantly deductive in nature and its formalization is the subject matter of first-order logic and its various nonmonotonic extensions. Reasoning from effects to possible causes has been formalized through the various schemes for abductive reasoning.

In medical diagnosis, the symptoms are explained using disease hypotheses. Similarly, in fault diagnosis, the incorrect behaviour of a digital system can be explained by determining possible circuit faults. In temporal reasoning, the propositions that hold in the current state can be explained by determining a possible set of events that could have occurred. In the context of natural language processing, abductive reasoning can be used for word-sense disambiguation and story understanding. Furthermore, tasks like designing a treatment from a diagnosis of the disease, or inferring possible causes has been formalized through the various abductive extensions. Reasoning from effects to possible causes is an aspect of first-order logic and its various nonmonotonic extensions. Reasoning from causes to effects is predominantly deductive in nature and its formalization is the subject matter of first-order logic and its various nonmonotonic extensions. Reasoning from effects to possible causes has been formalized through the various schemes for abductive reasoning.

In temporal reasoning, the propositions that hold in the current state can be explained by determining a possible set of events that could have occurred. In the context of natural language processing, abductive reasoning can be used for word-sense disambiguation and story understanding. Furthermore, tasks like designing a treatment from a diagnosis of the disease, or inferring additional properties of the current state from plausible events, all call for deductive reasoning using hypothesized explanations. So we believe that marrying both deductive and abductive reasoning strategies in a unified framework can be useful in practice.

2 Abductive Reasoning in Annotated Logic Programs

We first motivate the relevant issues through examples.

Consider the representation of the diagnostic knowledge for the "burnout problem" in an engine [26].

\[
\begin{align*}
\text{cup_hit_by_stone} & \leftarrow \text{stony_road}, \text{low_car_ht}. \\
\text{oil_cup_holed} & \leftarrow \text{cup_hit_by_stone}. \\
\text{engine_temp_high} & \leftarrow \text{oil_loss}, \text{engine_started}. \\
\text{oil_loss} & \leftarrow \text{oil_cup_holed}.
\end{align*}
\]

To formalize abduction, we need to (1) specify the "vocabulary" for the hypotheses to account for the observation, and (2) make rigorous when a set of hypotheses can be regarded as explaining the observation. In our example, the hypotheses can be constructed using facts that do not appear in any rule-head. An explanation can be thought of as a set of hypotheses that deductively implies the observation. So, the set \{\text{stony_road}, \text{low_car_ht}, \text{engine_started}\} is a valid explanation of the observation engine_temp_high.

Consider the chemical identification problem. A base transforms a red litmus paper blue. Both NaOH and KOH are bases. Under flame test, NaOH gives a golden yellow flame, while KOH gives a violet flame. The observation that a sample changes a red litmus blue can be explained by the facts — the sample is a base, or it is NaOH, or it is KOH. However, among these, the explanation the sample is a base seems to be the most "appropriate" by virtue of being the least restrictive. So, we wish to regard it as an acceptable explanation. Furthermore, using well-known facts from Chemistry, one can predict through deductive reasoning that the sample will be soapy to touch and would neutralize acids. Subsequently, when we learn the outcome of the flame test, the acceptable explanation can be refined to one of NaOH or KOH.

Consider information about the flying abilities of birds and mammals. Typically, birds fly, while mammals do not. Similarly, bats fly, while injured-bats do not. This can be expressed in our formalism as follows:

\[
\begin{align*}
\text{fly}(X) & : +\text{bird} \leftarrow \text{bird}(X) : +\bot \\
\text{fly}(X) & : -\text{mammal} \leftarrow \text{mammal}(X) : +\bot \\
\text{fly}(X) & : +\text{bat} \leftarrow \text{bat}(X) : +\bot \\
\text{fly}(X) & : -\text{injured_bat} \leftarrow \text{injured_bat}(X) : +\bot
\end{align*}
\]

Informally, the rule \( p : \alpha \rightarrow q : \beta \) states that if there is evidence for \( q \) that is greater than or equal to \( \beta \) in information-content and that has the same truth-content as \( p \), then one can confer support to \( p \) to the degree \( \alpha \). \( \bot \) (resp. \( \omega \)) is a defeasible (resp. strict) evidence with the least (resp. greatest) information-content. The symbol + (resp. –, *) indicates a supporting (resp. defeating, ambiguous) evidence. The evidences can be written in the ascending order of their information-content as \( \bot \), mammal, bat, injured_bat and \( \omega \). Bird is incomparable with mammal, bat and injured_bat on the information-scale, but it is larger than \( \bot \) and smaller than \( \omega \). This ordering information can be derived by inspecting the corresponding inheritance hierarchy automatically [13].

Given that the specimen bates is a mammal, that is, mammal(bates) : +\omega, we wish to conclude that bates does not fly, that is, fly(bates) : -mammal. Subsequently, when we learn facts like bat(bates) : +\omega and injured_bat(bates) : +\omega, we wish to revise it to fly(bates) : +bat and fly(bates) : -injured_bat respectively. We formalize these ideas using the notion of a supported model.

Given that bates does not fly, we can hypothesize
that *bates* is probably a mammal. However, if *bates* is in fact a bat, then we can explain its inability to fly by assuming that it is an injured-bat. Formally, the two plausible explanations for *fly(bates)*: - a mammal and *fly(bates)*: - injured-bat respectively. However, notice that the former explanation is preferably to the latter one that has redundant information. We define the notion of acceptable explanation for this purpose. Furthermore, suppose we also learn that *bat(bates)*: +ω holds. Then, the explanation *fly(bates)*: - mammal is no longer tenable in the face of the "stronger" conclusion *fly(bates)*: + bat. So, the only (acceptable) explanation for the fact that *bates* does not fly is *injured-bat(bates)*: +⊥, which implies *fly(bates)*: -injured-bat.

3 Annotated Logic Programs

3.1 Syntax

A term is an individual constant or a variable, or a "pattern" *f*(t₁, ..., tₙ), where *f* is an n-ary function symbol and tᵢ's are terms. An atom is a propositional constant or a formula q₁, ..., qₘ where qᵢ is an n-ary predicate symbol and tᵢ's are terms. *Literals* are of the form *p*: τ, where *p* is an atom and τ is an annotation drawn from the domain of annotations *A*. A rule is an expression of the form *p*: τ → q₁: γ₁, ..., qₘ: γₘ where *p* and qᵢ are atoms and τ and γᵢ's are annotations. The literal τ: r is referred to as the rule-body, while qᵢ: γᵢ, ..., qₘ: γₘ are referred to as the rule-head. A fact is a ground (i.e., variable-free) literal. A clause is either a fact or a rule. An annotated logic program is a set of clauses.

Following [2, 3, 7, 8, 13], the annotations are ordered along two different dimensions: in one, they are partially ordered by ≤ₖ on the basis of their *information-content*; in the other, they are related by an equivalence relation ≈ₖ on the basis of their *truth-content*. We define:

(τ ≤ₖ γ) iff (τ ≤ₖ γ) ∧ (τ ≠ γ); and

(τ ≈ₖ γ) iff (τ ≤ₖ γ) ∧ (τ ≈ₖ γ).

Furthermore, for our purposes, we assume that ≈ₖ has only the following three equivalence classes: C⁺, C⁻, and C*, corresponding to the defeating, ambiguous, and supporting evidences, respectively.

We impose certain acyclicity restrictions on the annotated logic programs for reasons that will become clear in the next section. This acyclicity requirement is analogous to the local stratification condition of [21] for ordinary logic programs. We formalize this requirement by defining a relation < on the ground literals as follows: We say that q: β <ₚ p: α if there is a ground instance *p*: α ← q: β of a rule in the program *P*, or there is a ground instance *p*: α ← r: γ of a rule in *P* and q: β <ₚ r: τ and γ ≈ₖ τ. We demand that, for the annotated logic programs of interest to us, the following restriction on the relation <ₚ be met: For all annotations α and β, (q: α <ₚ p: β) ⇒ (α ≈ₖ β). This condition prohibits recursion through negation. In the sequel, we refer to annotated logic programs that satisfy this condition as *stratified annotated logic programs*.

3.2 Model-theoretic Semantics

Let *P* be an annotated logic program. The *domain* *D* of Herbrand interpretations of *P* is a collection of all individual constants mentioned in *P*, and the ground (variable-free) terms that can be formed from them; it is often called the *Herbrand universe* of *P*. A *Herbrand base*, *Bₚ* of *P* is a collection of all ground (variable-free) atoms *p* that use only terms of *D* and the predicate symbols mentioned in *P*.

A *Herbrand interpretation* *I* of *P* is a partial mapping from the Herbrand base, *Bₚ*, of *P* to the set of annotations *A*. Given two interpretations *I* and *J*, *I* ⊆ₖ *J* (resp., *J* ⊆ₖ *I*) iff for every atom *p* ∈ *Bₚ*, such that *J*(*p*) is defined, *I*(*p*) is also defined and *J*(*p*) ≤ₖ *I*(*p*) (resp., *J*(*p*) ≤ₖ *I*(*p*)). An interpretation *I* is minimal in a set of interpretations *S* if and only if there is no *J* ∈ *S* such that *J* ⊆ₖ *I* and *J* ≠ *I*; *I* is maximal if and only if there is no *J* ∈ *S* such that *I* ⊆ₖ *J* and *J* ≠ *I*. A ground atom *p*: τ is satisfied in *I*, denoted *I* |=ₖ *p*: τ, if and only if *I*(*p*) is defined and τ ≤ₖ *I*(*p*); it is strongly satisfied in *I*, denoted *I* |=ₖ *p*: τ, if also τ ≤ₖ *I*(*p*). A ground (variable-free) rule *p*: τ → q₁: γ₁, ..., qₘ: γₘ is satisfied in *I* if and only if (∀ i: 1 ≤ i ≤ n: ⇒ *I* |=ₖ qᵢ: γᵢ) implies *I* |=ₖ *p*: τ. A nonground rule *p*: τ → q: γ is satisfied in *I* if and only if all its ground instances are satisfied in *I*. (A ground instance of a rule *r* is a ground rule obtained from *r* by replacing variables with elements in the Herbrand universe, where different occurrences of the same variable are replaced by the same element.) An annotated logic program *P* is satisfied in *I*, if all its clauses are satisfied in *I*.

An interpretation *I* is a model of a set of clauses *P* if
and only if $P$ is satisfied under $I$. A Herbrand model $I$ of $P$ is supported by $P$ if for every atom $p$ such that $I(p)$ is defined, we have $I(p) = \text{lub}_k \{ \tau \mid p : \tau \in \text{BODY} \}$ is a ground instance of a rule in $P$ and $I \models_{-k} \text{BODY} \}$. We assume that $I$ is undefined on any atom that does not appear in the head of a ground instance of a rule in $P$. Informally, $I$ is supported by $P$ if the evidences that it assigns to atoms are not stronger than what is warranted by $P$.

It is possible that there does not exist a single supported model, or that there are several different supported models. This is in contrast with the ordinary logic programs, and the annotated programs for inheritance specifications considered in [13] where a unique minimal supported model always exists. For instance, consider the following program: $\{ p : -1 \leftarrow q : +1. q : +1 \leftarrow p : +1. p : +1. \}$ The minimal Herbrand model of this program is: $\{ p : +1. q : +1. \}$. This model is not supported because $q : +1$ is unsupervised as $p : +1$ is not strongly satisfied in the model. Similarly, the program $\{ p : \alpha \leftarrow q : \beta. q : +1 \leftarrow p : +1. \}$ has two supported models: $\{ \}$ and $\{ p : \alpha, q : \beta. \}$. However, if $\leq_k$ is a semilattice (that is, every pair of elements has a least-upper bound) then we have the following result.

**Theorem 3.1** Every stratified annotated logic program has a unique minimal supported model.

## 4 Formalization of Abductive Reasoning

We now formalize abduction along the lines of [18, 19]. The abductive framework consists of three sets — the set $P$ consisting of the annotated logic program, the set $O$ of observations, and the set $A$ of predicate names corresponding to the abducible literals. An observation is syntactically an atom with its truth value. That is, an observation is not required to specify the information-content of the annotation. The ground literals formed from $A$ serve as possible hypotheses to explain the observations $O$.

We can recast the complete flying ability problem of Section 2 into the abductive framework as follows. The set $P$ is:

- $\text{fly}(X) : +\text{bird} \leftarrow \text{bird}(X) : +\bot$.
- $\text{fly}(X) : -\text{mammal} \leftarrow \text{mammal}(X) : +\bot$.
- $\text{fly}(X) : +\text{bat} \leftarrow \text{bat}(X) : +\bot$.
- $\text{fly}(X) : -\text{injured_bat} \leftarrow \text{injured_bat}(X) : +\bot$.
- $\text{mammal}(X) : +\omega \leftarrow \text{bat}(X) : +\omega$.
- $\text{mammal}(X) : +\text{bat} \leftarrow \text{bat}(X) : +\bot$.
- $\text{bat}(X) : +\omega \leftarrow \text{injured_bat}(X) : +\omega$.
- $\text{bat}(X) : +\omega \leftarrow \text{injured_bat}(X) : +\bot$.

where, $\bot <_k \text{bird} <_k \omega$ and $\bot <_k \text{mammal} <_k \text{bat} <_k \text{injured_bat} <_k \omega$. The first four rules specify conditions under which we can derive that an entity flies or does not fly, while the last four rules capture the class-subclass relationship. For instance, bats are a subclass of mammals. So if an entity is a bat then it is a mammal too. However, if the entity is a bat only by default, then it is a mammal by default, to a degree determined by the evidence bat. The set $A$ consists of the "class" names $\{\text{mammals}, \text{bats}, \text{injured_bats}\}$. The set $O$ is either $\{\text{fly}(\text{bates}) : +\}$ or $\{\text{fly}(\text{bates}) : -\}$.

Now we make rigorous the notion of an explanation. Informally, an explanation of an observation $O$ is a set of hypotheses $H$ that, in conjunction with the program $P$, implies $O$.

**Definition 4.1** A scenario of $(P, A, O)$ is a set $H$ of ground literals containing only the predicates in $A$.

**Definition 4.2** An extension of $(P, A, O)$ is the unique minimal supported model associated with $P \cup H$, where $H$ is a scenario of $P$.

**Definition 4.3** An explanation of $O$ in $(P, A, O)$ is a scenario $H$ such that the observations $O$ are strongly satisfied (that is, $\leq_{ab}$) in the extension of $P \cup H$.

In our example, the observation $\text{fly}(\text{bates}) : -$ can be explained by the two scenarios: $\{\text{mammal}(\text{bates}) : +\bot\}$ and $\{\text{injured_bat}(\text{bates}) : +\bot\}$. Similarly, the observation $\text{fly}(\text{bates}) : +$ can be explained by the scenario $\{\text{bat}(\text{bates}) : +\bot\}$. Notice that, the scenario $\{\text{mammal}(\text{bates}) : +\bot\}$ is a "better" explanation of $\text{fly}(\text{bates}) : -$ than the scenario $\{\text{injured_bat}(\text{bates}) : +\bot\}$ because the latter explanation makes much "stronger" assumptions about the object in question than the former. To filter out such intuitively unsatisfactory explanations from all possible explanations the notion of acceptability is developed.

As a first cut, we say that an explanation $H$ of $O$ in $(P, A, O)$ is acceptable if there does not exist another explanation $H'$ such that $H' \leq_{ab} H$. But, this definition is unsatisfactory because, it does not rule out the scenarios such as $\{\text{injured_bat}(\text{bates}) : +\bot\}$ as unacceptable. (Note that the literals corresponding to the predicates mammal and injured.bat are
incomparable wrt $<_{kt}$.) We can attempt fixing this flaw by asserting that an explanation $H$ of $O$ is acceptable if there does not exist another explanation $H'$ such that $\text{extension}(H') \subset_{kt} \text{extension}(H)$. This criterion rules out the scenario \{injured.bat(bates) : +⊥\} as unacceptable in the presence of the scenario \{mammal(bates) : +⊥\}. However, this approach does not work in general as illustrated below: Given an abductive framework \((p : +1 \leftarrow q : \alpha, p : +2 \leftarrow r : \beta, \{q, r\}, \{p : +\})\), the above criterion does not unequivocally prefer the explanation \{q : α\} over the explanation \{r : β\} as the literals corresponding to the atoms $q$ and $r$ are not related. This analysis leads us to the following final definition of acceptability that concentrates on the strength of evidence supporting the explained observations.

**Definition 4.4** An explanation $H$ of $O$ in $(P, A, O)$ is acceptable if and only if there does not exist another explanation $H'$ such that $\text{extension}(H') \subset_{kt} \text{extension}(H)|O$, where $E|O$ stands for the extension $E$ restricted to the observation literals.

The extension corresponding to the explanation \{mammal(bates) : +⊥\} contains $fly(bates) : ¬mammal$, while that corresponding to the explanation \{injured.bat(bates) : +⊥\} contains $fly(bates) : ¬injured.bat$. Thus, the scenario \{mammal(bates) : +⊥\} is the acceptable explanation of the observation $fly(bates) : ¬$. Similarly, for the abductive framework $E'$, the explanation \{q : +α\} is acceptable, while the explanation \{r : +β\} is not.

In case we also know the following facts:

- $hairy(X) : +mammal ⨅ mammal(X) : +⊥.$
- $winged(X) : mammal ⨅ bat(X) : +⊥.$
- $winged(X) : +bat ⨅ bat(X) : +⊥.$

then, \{mammal(bates) : +⊥\} supports

- $hairy(bates) : +mammal, winged(bates) : ¬mammal,$
- and \{injured.bat(bates) : +⊥\} supports
- $hairy(bates) : +mammal, winged(bates) : +injured.bat.$

This illustrates how abductive reasoning allows us to generate explanations from observations, and how deductive reasoning can generate further predictions in this integrated framework [18, 23].

There can be several acceptable explanations for a particular observation depending on the underlying semi-lattice structure. For instance, consider the framework $\mathcal{F} = \{\{p : α \leftarrow q : γ, p : β \leftarrow r : τ\}, \{q, r\}, \{p\}\}$.

- If $α \equiv β \equiv +$, but they are unrelated wrt $<_{k}$, then there are two acceptable explanations for $p : +$, namely, \{q : γ\} and \{r : τ\}.
- If $α <_{tk} β$, then there is only one acceptable explanation for $p : +$, namely, \{q : γ\}.
- If $α \neq β$, but they are unrelated wrt $<_{k}$, then there is one acceptable explanation for $p : +$ and one acceptable explanation for $p : −$.

5 Conclusion

In this paper, we extended the annotated language of [13] in various directions to obtain an enriched representation language. In particular, we permitted rule-bodies to be conjunction of literals, and the rules to be recursive. We identified a class of annotated logic programs called the stratified programs which can be given a unique supported minimal Herbrand model as their meaning. We then smoothly integrated abductive reasoning into this annotated logic framework by generalizing the work described in [18, 19]. We formalized the notion of an explanation, and specified when an explanation can be regarded as acceptable.

The literals in our language do not contain variables as annotations, in contrast with that in [12]. As a consequence, the annotation in the rule-head does not explicitly depend on the annotations in the rule-body. The semantics of $\equiv$ in [12] is also different from our interpretation of it here. Similarly, we differ from the approach of [20] in that we do not interpret annotations probabilistically.

References


