Preferential Theory Revision

Pierangelo Dell’Acqua
Dept. of Science and Technology - ITN
Linköping University, Sweden

Luís Moniz Pereira
Centro de Inteligência Artificial - CENTRIA
Universidade Nova de Lisboa, Portugal

September, 2005  CMSRA 05, Lisbon Portugal
Summary

Employing a *logic program* approach, this work focuses on applying preferential reasoning to theory revision, both by means of:

- preferences among existing theory rules,
- preferences on the possible abductive extensions to the theory.

And, in particular, how to prefer among plausible abductive explanations justifying observations.
• Logic program semantics and procedures have been used to characterize preferences among the rules of a theory (cf. Brewka).

• Whereas the combination of such rule preferences with program updates and the updating of the preference rules themselves (cf. Alferes & Pereira) have been tackled, a crucial ingredient has been missing, namely:
  • the consideration of abductive extensions to a theory, and
  • the integration of revisable preferences among such extensions.

• The latter further issue is the main subject of this work.
• We take a theory expressed as a logic program under stable
model semantics, already infused with preferences between rules,
and we add a set of abducibles constituting possible extensions to
the theory, governed by conditional priority rules amongst
preferred extensions.

• Moreover, we cater for minimally revising the preferential
priority theory itself, so that a strict partial order is always
enforced, even as actual preferences are modified by new incoming
information.

• This is achieved by means of a diagnosis theory on revisable
preferences over abducibles, and its attending procedure.
• First we supply some epistemological background to the problem at hand.

• Then we introduce our preferential abduction framework, and proceed to apply it to exploratory data analysis.

• Next we consider the diagnosis and revision of preferences, theory and method, and illustrate it on the data exploration example.

• Finally, we exact general epistemic remarks on the approach.
The theoretical notions of preference and rationality with which we are most familiar are those of the economists'. Economic preference is a comparative choice between alternative outcomes, whereby a rational (economic) agent is one whose expressed preferences over a set of outcomes exhibits the structure of a complete pre-order.

However, preferences themselves may change. Viewing this phenomena as a comparative choice, however, entails that there are meta-level preferences whose outcomes are various preference rankings of beliefs, and that an agent chooses a change in preference based upon a comparative choice between the class of first-order preferences (cf. Doyle).
• But this is an unlikely model of actual change in preference, since we often evaluate changes -- including whether to abandon a change in preference -- based upon items we learn after a change in preference is made.

• Hence, a realistic model of preference change will not be one that is couched exclusively in decision theoretic terms.

• Rather, when a conflict occurs in updating beliefs by new information, the possible items for revision should include both the set of conflicting beliefs and a reified preference relation underlying the belief set.

• The reason for adopting this strategy is that we do not know, a priori, what is more important -- our data or our theory.
Preferences and rationality - 3

• Rather, as Isaac Levi has long advocated (cf. Levi), rational inquiry is guided by pragmatic considerations not a priori constraints on rational belief.

• On Levi's view, all justification for change in belief is pragmatic in the sense that justifications for belief fixation and change are rooted in strategies for promoting the goals of a given inquiry. Setting these parameters for a particular inquiry fixes the theoretical constraints for the inquiring agent.

• The important point to stress here is that there is no conflict between theoretical and practical reasoning on Levi's approach, since the prescriptions of Levi's theory are not derived from minimal principles of rational consistency or coherence.
Suppose your scientific theory predicts an observation, \( o \), but you in fact observe \( \neg o \). The problem of carrying out a principled revision of your theory in light of the observation \( \neg o \) is surprisingly difficult.

One issue that must be confronted is what the principle objects of change are. If theories are simply represented as sets of sentences and prediction is represented by material implication, then we are confronted with (Duhem's Problem):

*If a theory entails an observation for which we have disconfirming evidence, logic alone won't tell you which among the conjunction of accepted hypotheses to change in order to restore consistency.*

The serious issue raised by Duhem's problem is whether disconfirming evidence targets the items of a theory in need of revision in a principled manner.
The AGM conception of belief change differs to Duhem's conception of the problem in important respects:

- First, whereas the item of change on Duhem's account is a set of sentences, the item of change on the AGM conception is a belief state, represented as a pair consisting of a logically closed set of sentences (a belief set) and a selection function.

- Second, resulting theories are not explicitly represented, when replacing entailment by the AGM postulates.

What remains in common is what Hansson has called the input-assimilating model of revision, whereby the object of change is a set of sentences, the input item is a particular sentence, and the output is a new set of sentences.
Preferences and theory revision - 3

• One insight to emerge is that the input objects for change may not be single sentences, but a sentence-measure pair (cf. Nayak), where the value of the measure represents the entrenchment of the sentence and thereby encodes the ranking of this sentence in the replacement belief set (cf. Nayak, Rott, Spohn).

• But once we acknowledge that items of change are not belief simpliciter, but belief and order coordinates, then there are two potential items for change: the acceptance or rejection of a belief and the change of that belief in the ordering. Hence, implicitly, the problem of preference change appears here as well.

• Within the AGM model of belief change, belief states are the principal objects of change: propositional theory (belief set) changed according to the input-assimilating model, whereby the object of change (a belief set) is exposed to an input (a sentence) and yields a new belief set.
Preferences and defeasible reasoning - 1

• Computer science has adopted logic as its general foundational tool, while Artificial Intelligence (AI) has made viable the proposition of turning logic into a bona fide computer programming language.

• AI has developed logic beyond the confines of monotonic cumulativity, typical of the precise, complete, endurable, condensed, and closed mathematical domains, in order to open it up to the non-monotonic real world domain of imprecise, incomplete, contradictory, arguable, revisable, distributed, and evolving knowledge.

• In short, AI has added dynamics to erstwhile statics. Indeed, classical logic has been developed to study well-defined, consistent, and unchanging mathematical objects. It thereby acquired a static character.
• **AI** needs to deal with knowledge in flux, and less than perfect conditions, by means of more dynamic forms of logic. Too many things can go wrong in an open non-mathematical world, some of which we don't even suspect.

• In the real world, any setting is too complex already for us to define exhaustively each time. We have to allow for unforeseen exceptions to occur, based on new incoming information.

• Thus, instead of having to make sure or prove that some condition is not present, we may assume it is not (the Closed World Assumption - **CWA**).

• On condition that we are prepared to accept subsequent information to the contrary, i.e. we may assume a more general rule than warranted, but must henceforth be prepared to deal with arising exceptions.
Preferences and defeasible reasoning - 3

- Much of this has been the focus of research in logic programming.

- This is a field which uses logic directly as a programming language, and provides specific implementation methods and efficient working systems to do so.

- Logic programming is, moreover, much used as a staple implementation vehicle for logic approaches to AI.
Our Technical Approach

1. Framework (language, declarative semantics)

2. Preferring abducibles

3. Exploratory data analysis

4. Revising relevancy relations
1. Framework: language $\mathcal{L}$

**Domain literal** in $\mathcal{L}$ is a domain atom $A$ or its default negation not $A$.

**Domain rule** in $\mathcal{L}$ is a rule of the form:

$$A \leftarrow L_1, \ldots, L_t \quad (t \geq 0)$$

where $A$ is a domain atom and every $L_i$ is a domain literal.

Let $n_r$ and $n_u$ be the names of two domain rules $r$ and $u$. Then,

$n_r < n_u$ is a priority atom

meaning that $r$ has priority over $u$.

**Priority rule** in $\mathcal{L}$ is a rule of the form:

$$n_r < n_u \leftarrow L_1, \ldots, L_t \quad (t \geq 0)$$

where every $L_i$ is a domain literal or a priority literal.
Program over $\mathcal{L}$ is a set of domain rules and priority rules.

- Every program $P$ has associated a set $A_p$ of domain literals, called abducibles

- Abducibles in $A_p$ do not have rules in $P$ defining them
Declarative semantics

(2-valued) interpretation $M$ is any set of literals such that, for every atom $A$, precisely one of the literals $A$ or not $A$ belongs to $M$.

Set of default assumptions

- $\text{Default}(P, M) = \{ \text{not } A : \neg \exists (A \leftarrow L_1, \ldots, L_t) \text{ in } P \text{ and } M \models L_1, \ldots, L_t \}$

Stable models

- $M$ is a stable model (SM) of $P$ iff $M = \text{least}(P \cup \text{Default}(P, M))$

- Let $\Delta \subseteq A_P$. $M$ is an abductive stable model (ASM) with hypotheses $\Delta$ of $P$ iff $M = \text{least}(P^+ \cup \text{Default}(P^+, M))$, with $P^+ = P \cup \Delta$
Unsupported rules

\[ \text{Unsup}(P,M) = \{ r \in P : M \models \text{head}(r), M \models \text{body}^+(r) \text{ and } M \not\models \text{body}^-(r) \} \]

Unpreferred rules

\[ \text{Unpref}(P,M) = \text{least}( \text{Unsup}(P,M) \cup Q ) \]
\[ Q = \{ r \in P : \exists u \in (P - \text{Unpref}(P, M)), M \models n_u < n_r, M \models \text{body}^+(u) \text{ and } \]
\[ [ \text{not head}(u) \in \text{body}^-(r) \text{ or (not head}(r) \in \text{body}^-(u) \text{ and } M \models \text{body}(r) \} ] \]

Let \( \Delta \subseteq A_P \) and \( M \) an abductive stable model with hypotheses \( \Delta \) of \( P \).

\( M \) is a preferred abductive stable model iff \( < \) is a strict partial order, i.e. :

\[ \begin{align*}
&\text{if } M \models n_u < n_r \text{, then } M \not\models n_r < n_u \\
&\text{if } M \models n_u < n_r \text{ and } M \models n_r < n_z \text{, then } M \models n_u < n_z \\
\end{align*} \]

and if \( M = \text{least}(P^+ - \text{Unpref}(P^+,M) \cup \text{Default}(P^+,M)) \), with \( P^+ = P \cup \Delta \)
2. Preferring abducibles

The evaluation of alternative explanations is a central problem in abduction, because of

• combinatorial explosion of possible explanations to handle.

So, generate only the explanations that are relevant to the problem at hand.

Several approaches have been proposed:

• Some of them based on a global criterium
  Drawback: domain independent + computationally expensive

• Other approaches allow rules encoding domain specific information about the likelihood that a particular assumption be true.
In our approach, *preferences* among abducibles can be expressed in order to discard unwanted assumptions.

- Technically, preferences over alternative abducibles are coded into even cycles over default negation, and preferring a rule will break the cycle in favour of one abducible or another.

The notion of *expectation* is employed to express preconditions for assuming abducibles.

- An abducible can be assumed only if it is confirmed, i.e.:
  - there is an expectation for it, and
  - unless there is expectation to the contrary (*expect_not*)
Language $\mathcal{L}^*$

**Relevance atom** is an atom of the form $a \triangleleft b$, where $a$ and $b$ are abducibles.

$a \triangleleft b$ means that $a$ is preferred to $b$ (or more relevant than $b$)

**Relevance rule** is a rule of the form:

$$a \triangleleft b \leftarrow L_1, \ldots, L_t \quad (t \geq 0)$$

where every $L_i$ is a domain literal or a relevance literal.

Let $\mathcal{L}^*$ be a language consisting of domain rules and relevance rules.
Consider a situation where Claire drinks either tea or coffee (but not both). And Claire prefers coffee to tea when sleepy.

This situation can be represented by a program $Q$ over $\mathcal{L}^*$ with abducibles $A_Q= \{\text{tea, coffee}\}$.

```
program Q (\mathcal{L}^*)
?- drink

drink ← tea
drink ← coffee
expect(tea)
expect(coffee)
expect_not(coffee) ← blood_pressure_high
coffee ≺ tea ← sleepy
```
**Relevant ASMs**

We need to distinguish which abductive stable models (ASMs) are relevant wrt. relevancy relation $\triangleleft$.

Let $Q$ be a program over $\mathcal{L}^*$ with abducibles $A_Q$. Let $a \in A_Q$.

$M$ is a relevant abductive stable model of $Q$ with hypotheses $\Delta\{a\}$ iff:

- $\forall x, y \in A_Q$, if $M \models x \triangleleft y$ then $M \not\models y \triangleleft x$
- $\forall x, y \in A_Q$, if $M \models x \triangleleft y$ and $M \models y \triangleleft z$ then $M \models x \triangleleft z$
- $M \models \text{expect}(a), M \not\models \text{expect\_not}(a)$
- $\neg \exists (x \triangleleft a \leftarrow L_1,\ldots,L_t)$ in $Q$ such that $M \models L_1,\ldots,L_t$ and $M \models \text{expect}(x), M \not\models \text{expect\_not}(x)$
- $M = \text{least}(Q^+ \cup \text{Default}(Q^+,M))$, with $Q^+ = Q \cup \Delta$
Example

\[
\begin{align*}
\text{drink} & \leftarrow \text{tea} \\
\text{drink} & \leftarrow \text{coffee} \\
\text{expect}(\text{tea}) & \\
\text{expect}(\text{coffee}) & \\
\text{expect}\_\text{not}(\text{coffee}) & \leftarrow \text{blood\_pressure\_high} \\
\text{coffee} & \leftarrow \text{tea} \\
\text{tea} & \leftarrow \text{sleepy}
\end{align*}
\]

program Q (\(\mathcal{L}^*\))

\[
M_1 = \{\text{expect}(\text{tea}), \text{expect}(\text{coffee}), \text{coffee}, \text{drink}\} \text{ with } \Delta_1 = \{\text{coffee}\}
\]

\[
M_2 = \{\text{expect}(\text{tea}), \text{expect}(\text{coffee}), \text{tea}, \text{drink}\} \text{ with } \Delta_2 = \{\text{tea}\}
\]

\text{for which } M_1 \vDash \text{drink and } M_2 \vDash \text{drink}
program Q (\(L^*\))

drink \leftarrow \text{tea} \\
drink \leftarrow \text{coffee} \\
extpect(\text{tea}) \\
extpect(\text{coffee}) \\
extpect\neg(\text{coffee}) \leftarrow \text{blood\_pressure\_high} \\
\text{coffee} \triangleright \text{tea} \leftarrow \text{sleepy} \\
\text{sleepy}

Relevant ASMs:

\(M_1 = \{\text{expect(\text{tea}), expect(\text{coffee}), coffee, drink, sleepy}\} \) with \(\Delta_1 = \{\text{coffee}\}\)

for which \(M_1 \vDash \text{drink}\)
Transformation \( \Sigma \)

Proof procedure for \( \mathcal{L}^* \) based on a syntactic transformation mapping \( \mathcal{L}^* \) into \( \mathcal{L} \).

Let \( Q \) be a program over \( \mathcal{L}^* \) with abducibles \( A_Q = \{a_1, \ldots, a_m\} \).

The program \( P = \Sigma(Q) \) with abducibles \( A_P = \{\text{abduce}\} \) is obtained as follows:

- \( P \) contains all the domain rules in \( Q \);
- for every \( a_i \in A_Q \), \( P \) contains the domain rule:
  \[ \text{confirm}(a_i) \leftarrow \text{expect}(a_i), \neg \text{expect\_not}(a_i) \]
- for every \( a_i \in A_Q \), \( P \) contains the domain rule:
  \[ a_i \leftarrow \text{abduce}, \neg a_1, \ldots, \neg a_{i-1}, \neg a_{i+1}, \ldots, \neg a_m, \text{confirm}(a_i) \quad (r_i) \]
- for every rule \( a_i \triangleleft a_j \leftarrow L_1, \ldots, L_t \) in \( Q \), \( P \) contains the priority rule:
  \[ r_i \triangleleft r_j \leftarrow L_1, \ldots, L_t \]
Example

drink ← tea
drink ← coffee
expect(tea)
expect(coffee)
expect_not(coffee) ← blood_pressure_high
coffee ∼ tea ← sleepy

drink ← tea
drink ← coffee
expect(tea)
expect(coffee)
expect_not(coffee) ← blood_pressure_high
coffee ← abduce, not tea, confirm(coffee) (1)
tea ← abduce, not coffee, confirm(tea) (2)
confirm(tea) ← expect(tea), not expect_not(tea)
confirm(coffee) ← expect(coffee), not expect_not(coffee)
1 < 2 ← sleepy

P = Σ(Q) (L)

Q (L*)
Correctness of $\Sigma$

Let $\mathcal{M}$ be the interpretation obtained from $\mathcal{M}$ by removing the abducible $\text{abduce}$, the priority atoms, and all the domain atoms of the form $\text{confirm}(\cdot)$

Property

Let $Q$ be a program over $\mathcal{L}^*$ with abducibles $A_Q$ and $P = \Sigma(Q)$. The following are equivalent:

- $\mathcal{M}$ is a preferred abductive stable model with $\Delta = \{\text{abduce}\}$ of $P$,
- $\mathcal{M}$ is a relevant abductive stable model of $Q$. 
Example

\[
\begin{align*}
\text{drink} & \leftarrow \text{tea} \\
\text{drink} & \leftarrow \text{coffee} \\
\text{expect(tea)} & \\
\text{expect(coffee)} & \\
\text{expect_not(coffee)} & \leftarrow \text{blood_pressure_high} \\
\text{coffee} & \leftarrow \text{abduce, not tea, confirm(coffee)} \\
\text{tea} & \leftarrow \text{abduce, not coffee, confirm(tea)} \\
\text{confirm(tea)} & \leftarrow \text{expect(tea), not expect_not(tea)} \\
\text{confirm(coffee)} & \leftarrow \text{expect(coffee), not expect_not(coffee)} \\
1 < 2 & \leftarrow \text{sleepy}
\end{align*}
\]

Preferred ASMs with \( \Delta = \{\text{abduce}\} \) of \( P \):

\[ M_1 = \{\text{confirm(tea)}, \text{confirm(coffee)}, \text{expect(tea)}, \text{expect(coffee)}, \text{coffee}, \text{drink}\} \]

\[ M_2 = \{\text{confirm(tea)}, \text{confirm(coffee)}, \text{expect(tea)}, \text{expect(coffee)}, \text{tea}, \text{drink}\} \]

\[ P = \Sigma(Q) \quad (\mathcal{L}) \]
P = Σ(Q) \quad (L)

\begin{align*}
drink & \leftarrow \text{tea} \\
drink & \leftarrow \text{coffee} \\
\text{expect(tea)} & \\
\text{expect(coffee)} & \\
\text{expect_not(coffee)} & \leftarrow \text{blood_pressure_high} \\
\text{coffee} & \leftarrow \textbf{abduce}, \text{not tea, confirm(coffee)} \quad (1) \\
\text{tea} & \leftarrow \textbf{abduce}, \text{not coffee, confirm(tea)} \quad (2) \\
\text{confirm(tea)} & \leftarrow \text{expect(tea), not expect_not(tea)} \\
\text{confirm(coffee)} & \leftarrow \text{expect(coffee), not expect_not(coffee)} \\
1 < 2 & \leftarrow \text{sleepy} \\
\text{sleepy} & \\
\end{align*}

Preferred ASMs with \(\Delta = \{\textbf{abduce}\}\) of \(P\):

\(M_1 = \{ \text{confirm(tea)}, \text{confirm(coffee)}, \text{expect(tea)}, \text{expect(coffee)}, \text{coffee}, \text{drink}, \text{sleepy}, 1<2 \} \)
3. Exploratory data analyses

Exploratory data analysis aims at suggesting a pattern for further inquiry, and contributes to the conceptual and qualitative understanding of a phenomenon.

- Assume that an unexpected phenomenon, \( x \), is observed by an agent Bob. And Bob has three possible hypotheses (abducibles) \( a, b, c \), capable of explaining it.

- In exploratory data analysis, after observing some new facts, one abduces explanations and explores them to check predicted values against observations. Though there may be more than one convincing explanation, one abduces only the more plausible of them.
Example

Bob’s theory: Q with abducibles $A_Q=\{a, b, c\}$

\[
\begin{align*}
x & \leftarrow a \\
x & \leftarrow b \\
x & \leftarrow c \\
\text{expect}(a) \\
\text{expect}(b) \\
\text{expect}(c) \\
a & \triangleleft c \leftarrow \neg e \\
b & \triangleleft c \leftarrow \neg e \\
b & \triangleleft a \leftarrow d
\end{align*}
\]

$x$ - the car does not start

\[
\begin{align*}
a & - \text{the battery has problems} \\
b & - \text{the ignition is damaged} \\
c & - \text{there is no gasoline in the car} \\
d & - \text{the car's radio works} \\
e & - \text{Bob’s wife used the car} \\
\text{exp} & - \text{test if the car's radio works}
\end{align*}
\]

Relevant ASMs to explain observation $x$:

$M_1 = \{ \text{expect}(a), \text{expect}(b), \text{expect}(c), a \triangleleft c, b \triangleleft c, a, x \}$ with $\Delta_1 = \{a\}$

$M_2 = \{ \text{expect}(a), \text{expect}(b), \text{expect}(c), a \triangleleft c, b \triangleleft c, b, x \}$ with $\Delta_2 = \{b\}$
To prefer between a and b, one can perform some experiment \text{exp} to obtain confirmation (by observing the environment) about the most plausible hypothesis.

To do so, one can employ active rules:

\[ L_1, \ldots, L_t \Rightarrow \alpha: A \]

where \( L_1, \ldots, L_t \) are domain literals, and \( \alpha:A \) is an action literal.

Such a rule states: Update the theory of agent \( \alpha \) with \( A \) if the rule body is satisfied in all relevant ASMs of the present agent.
One can add the following rules (where env plays the role of the environment) to the theory Q of Bob:

\[
\begin{align*}
\text{choose} & \leftarrow a \\
\text{choose} & \leftarrow b \\
a & \Rightarrow \text{Bob:chosen} \\
b & \Rightarrow \text{Bob:chosen} \\
\text{choose} & \Rightarrow \text{Bob:}(\text{not chosen} \Rightarrow \text{env:exp})
\end{align*}
\]

Bob still has two relevant ASMs:

\[
M_3 = M_1 \cup \{\text{choose}\} \text{ and } M_4 = M_2 \cup \{\text{choose}\}.
\]

As choose holds in both models, the last active rule is triggerable.

When triggered, it will add to Bob's theory (at the next state) the active rule:

\[
\text{not chosen} \Rightarrow \text{env:exp}
\]
4. Revising relevancy relations

Relevancy relations are subject to be modified when:

- new information is brought to the knowledge of an individual,
- one needs to represent and reason about the simultaneous relevancy relations of several individuals.

The resulting relevancy relation may not satisfy the required properties (e.g., a strict partial order - spo) and must therefore be revised.

We investigate next the problem of revising relevancy relations by means of declarative debugging.
Consider the boolean composition of two relevancy relations: \( \triangleleft = \triangleleft_1 \cup \triangleleft_2 \).

\( \triangleleft \) might not be an spo:

\[
\begin{align*}
x & \leftarrow a & u \triangleleft v & \leftarrow u \triangleleft_1 v \\
x & \leftarrow b & u \triangleleft v & \leftarrow u \triangleleft_2 v \\
x & \leftarrow c & \\
\text{expect}(a) & & a \triangleleft_1 b \\
\text{expect}(b) & & b \triangleleft_1 c \\
\text{expect}(c) & & b \triangleleft_2 a
\end{align*}
\]

\( Q \) (\( \mathcal{L}^* \))

\( u, v \) variables ranging over abducibles

\( Q \) does not have any relevant ASM because \( \triangleleft \) is not a strict partial order.
Language $\mathcal{L}^+$

To revise relevancy relations, we introduce the language $\mathcal{L}^+$. 

**Integrity constraint** is a rule of the form:

$$\bot \leftarrow L_1, \ldots, L_t \quad (t \geq 0)$$

every $L_i$ is a domain literal or a relevance literal, and $\bot$ is a domain atom denoting contradiction.

$\mathcal{L}^+$ consists of domain rules, relevance rules, and integrity constraints.

- In $\mathcal{L}^+$ there are no abducibles, and its meaning is characterized by SMs.

Given a program $T$ and a literal $L$, $T \models L$ holds iff $L$ is true in every SM of $T$.

$T$ is **contradictory** if $T \models \bot$
**Diagnoses**

Given a contradictory program $T$, to revise its contradiction ($\bot$) we modify $T$ by adding and removing rules. In this framework, the diagnostic process reduces to finding such rules.

Given a set $C$ of predicate symbols of $\mathcal{L}^+$, $C$ induces a partition of $T$ into two disjoint parts: $T = T_c \cup T_s$

$T_c$ is the changeable part and $T_s$ the stable one.

Let $D$ be a pair $\langle U, I \rangle$ where $U \cap I = \emptyset$, $U \subseteq C$ and $I \subseteq T_c$. Then $D$ is a diagnosis for $T$ iff $(T-I) \cup U \not\equiv \bot$.

$D = \langle U, I \rangle$ is a minimal diagnosis if there exists no diagnosis $D_2 = \langle U_2, I_2 \rangle$ for $T$ such that $(U_2 \cup I_2) \subset (U \cup I)$. 
Example

Let $C = \{<_1, <_2\}$

$T$ admits three minimal diagnoses:

$D_1 = \langle \{\}, \{a <_1 b\} \rangle$, $D_2 = \langle \{\}, \{b <_1 c, b <_2 a\} \rangle$ and $D_3 = \langle \{a <_1 c\}, \{b <_2 a\} \rangle$.

It holds that $T \vDash \bot$. 

$T$ (\$L^+$)
**Computing minimal diagnoses**

To compute the minimal diagnoses of a contradictory program $T$, we employ a contradiction removal method.

- Based on the idea of revising (to $false$) some of the default atoms.
- A default atom 'not $A$' can be revised to $false$ simply by adding $A$ to $T$.
- The default literals 'not $A$' that are allowed to change their truth value are exactly those for which there exists no rule in $T$ defining $A$. Such literals are called revisables.
- A set $Z$ of revisables is a revision of $T$ iff $T \cup Z \not\models \bot$
Example

Consider the contradictory program \( T = T_c \cup T_s \)

\[
\begin{align*}
T_c & \quad : & a & \leftarrow \text{not } b, \text{not } c \\
& & a' & \leftarrow \text{not } d \\
& & c & \leftarrow e \\
T_s & \quad : & \bot & \leftarrow a, a' \\
& & \bot & \leftarrow b \\
& & \bot & \leftarrow d, \text{not } f
\end{align*}
\]

with revisables \( \{b, d, e, f\} \).

The revisions of \( T \) are \( \{e\}, \{d,f\}, \{e,f\} \) and \( \{d,e,f\} \), where the first two are minimal.
Transformation $\Gamma$

$\Gamma$ maps programs over $\mathcal{L}^+$ into equivalent programs that are suitable for contradiction removal.

The transformation $\Gamma$ that maps $T$ into a program $T' = \Gamma(T)$ is obtained by applying to $T$ the following two operations:

- Add not incorrect ($A \leftarrow \text{Body}$) to the body of each rule $A \leftarrow \text{Body}$ in $T_c$
- Add the rule:

$$p(x_1, \ldots, x_n) \leftarrow \text{uncovered}( p(x_1, \ldots, x_n) )$$

for each predicate $p$ with arity $n$ in $C$, where $x_1, \ldots, x_n$ are variables.

**Property:** Let $T$ be a program over $\mathcal{L}^+$ and $L$ be a literal. Then,

$$T \models L \iff \Gamma(T) \models L$$
Example

<table>
<thead>
<tr>
<th>x ← a</th>
<th>u ⊲ v ← u ⊲₁ v</th>
<th>⊥ ← u ⊲ u</th>
</tr>
</thead>
<tbody>
<tr>
<td>x ← b</td>
<td>u ⊲ v ← u ⊲₂ v</td>
<td>⊥ ← u ⊲ v, v ⊲ u</td>
</tr>
<tr>
<td>x ← c</td>
<td></td>
<td>⊥ ← u ⊲ v, v ⊲ z, not u ⊲ z</td>
</tr>
<tr>
<td>expect(a)</td>
<td>a ⊲₁ b ← not incorrect(a ⊲₁ b)</td>
<td></td>
</tr>
<tr>
<td>expect(b)</td>
<td>b ⊲₁ c ← not incorrect(b ⊲₁ c)</td>
<td></td>
</tr>
<tr>
<td>expect(c)</td>
<td>b ⊲₂ a ← not incorrect(b ⊲₂ a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>u ⊲₁ v ← uncovered(u ⊲₁ v)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>u ⊲₂ v ← uncovered(u ⊲₂ v)</td>
<td></td>
</tr>
</tbody>
</table>

\( \Gamma( T ) \) admits three minimal revisions wrt. the revisables of the form incorrect(.) and uncovered(.):

\[ Z_1 = \{ \text{incorrect}(a ⊲₁ b) \} \]

\[ Z_2 = \{ \text{incorrect}(b ⊲₁ c), \text{incorrect}(b ⊲₂ a) \} \]

\[ Z_3 = \{ \text{uncovered}(a ⊲₁ c), \text{incorrect}(b ⊲₂ a) \} \]
The following result relates the minimal diagnoses of $T$ with the minimal revisions of $\Gamma(T)$.

**Theorem**

The pair $D = \langle U, I \rangle$ is a diagnosis for $T$ iff

$$Z = \{\text{uncovered}(A): A \in U\} \cup \{\text{incorrect}(A \leftarrow \text{Body}): A \leftarrow \text{Body} \in I\}$$

is a revision of $\Gamma(T)$, where the revisables are all the literals of the form $\text{incorrect}(.)$ and $\text{uncovered}(.)$. Furthermore, $D$ is a minimal diagnosis iff $Z$ is a minimal revision.

To compute the minimal diagnosis of $T$ we consider the transformed program $\Gamma(T)$ and compute its minimal revisions. An algorithm for computing minimal revisions has been previously developed.
We have shown that preferences and priorities (they too a form of preferential expressiveness) can enact choices amongst rules and amongst abducibles, which are dependant on the specifics of situations, all in the context of theories and theory extensions expressible as logic programs.

As a result, using available transformations provided here and elsewhere (Alferes & Damásio & Pereira), these programs are executable by means of publicly available state-of-the-art systems.

Elsewhere, we have furthermore shown how preferences can be integrated with knowledge updates, and how they too fall under the purview of updating, again in the context of logic programs.

Preferences about preferences are also adumbrated therein.
Achievements - 2

• We have employed the two-valued Stable Models semantics to provide meaning to our logic programs, but we could just as well have employed the three-valued Well-Founded Semantics for a more skeptical preferential reasoning.

• Other logic program semantics are available too, such as the Revised Stable Model semantics, a two-valued semantics which resolves odd loops over default negation, arising from the unconstrained expression of preferences, by means of reductio ad absurdum (Pereira&Pinto). Indeed, when there are odd loops over default negation in a program, Stable Model semantics does not afford the program with a semantics.
Achievements - 3

• Also, we need not necessarily insist on a strict partial order for preferences, but have indicated that different conditions may be provided.

• The possible alternative revisions, required to satisfy the conditions, impart a non-monotonic or defeasible reading of the preferences given initially.

• Such a generalization permits us to go beyond a simply foundational view of preferences, and allows us to admit a coherent view as well, inasmuch several alternative consistent stable models may obtain for our preferences, as a result of each revision.
In (Rott:2001), arguments are given as to how epistemic entrenchment can be explicitly expressed as preferential reasoning. And, moreover, how preferences can be employed to determine believe revisions, or, conversely, how belief contractions can lead to the explicit expression of preferences.

(Doyle:2004) provides a stimulating survey of opportunities and problems in the use of preferences, reliant on AI techniques.

We advocate that the logic programming paradigm (LP) provides a well-defined, general, integrative, encompassing, and rigorous framework for systematically studying computation, be it syntax, semantics, procedures, or attending implementations, environments, tools, and standards.
Concluding remarks - 2

- **LP** approaches problems, and provides solutions, at a sufficient level of abstraction so that they generalize from problem domain to problem domain.

- This is afforded by the nature of its very foundation in logic, both in substance and method, and constitutes one of its major assets.

- Indeed, computational reasoning abilities such as assuming by default, abducing, revising beliefs, removing contradictions, preferring, updating, belief revision, learning, constraint handling, etc., by dint of their generality and abstract characterization, once developed can readily be adopted by, and integrated into, distinct topical application areas.
• No other computational paradigm affords us with the wherewithal for their coherent conceptual integration.

• And, all the while, the very vehicle that enables testing its specification, when not outright its very implementation (Pereira:2002).

• Consequently, it merits sustained attention from the community of researchers addressing the issues we have considered and have outlined.