Prospective Updating of Theories with Preferences

Luís Moniz Pereira[1], Pierangelo Dell'Acqua[1,2], Gonçalo Lopes[1]

[1]Centro de Inteligência Artificial – CENTRIA
Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
lmp@di.fct.unl.pt, goncaloclopes@gmail.com

[2]Department of Science and Technology – ITN
Linköping University, 60174 Norrköping, Sweden
pier@itn.liu.se

Abstract

This work focuses on updating and revising theories with preferences within the context of logic programming. This aim is achieved by first exploiting preferences to reduce the number of abductive extensions of the initial theory, then by using the observations to confirm or deny the abduced hypotheses. In case the observations disconfirm the preferred abduced hypotheses, a revision process is launched in order to revise the theory's preferences with respect to the new acquired observations. A methodology for model-based diagnosis is also proffered as an application of preferential theory revision, using observations to disambiguate among possible relevant revision scenarios.

1. Introduction

Logic program semantics and procedures have been used to characterize preferences among the rules of a theory [5]. Whereas the combination of such rule preferences with program updates and the updating of the preference rules themselves [2] have been tackled, a crucial ingredient had been missing, namely the consideration of abductive extensions to a theory, and the integration of revisable preferences among such extensions. The derivations from this preferential theory revision [6] to prospective reasoning [12] and model-based diagnosis are the main subject of this paper.

We take a theory expressed as a logic program under the stable model semantics, already infused with pairwise conditional preferences amongst rules. The possible abductive extensions to the theory can be expressed by a set of abducible literals, over which we should also be able to establish conditional priority relations.

First we supply some epistemological background to the problem at hand, and describe the general requirements of model-based diagnosis and its relation to abductive reasoning. We then consider a real-world example of abductive diagnosis in medical reasoning and proceed to explain the importance of the proposed approach to prospective logic programming. Then we introduce our preferential abduction framework, in order to address the issues raised by the medical example in a context of updatable abductive theories. Some additional examples of preference updating are presented and we conclude with general epistemic remarks about the entire approach and future directions of study.

2. Preferences, Rationality, Theory Revision and AI

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 phenomenon 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 pairwise choice between the class of first-order preferences [7].

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.

Rather, as Isaac Levi has long advocated [11], 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 ¬o. The problem of carrying out a principled revision of your theory in light of the observation ¬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 [8]: *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, in a principled manner, the items of a theory in need of revision.

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 of selection function. Second, the revised resulting theories are not explicitly represented, when replacing Duhem’s account by the AGM postulates approach. What remains in common is what Hansson [10] 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.

One insight to emerge is that the input objects for change may not be single sentences, but a

sentence-measure pair [13], 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 [13, 17, 18]. 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.

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.

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.

3. Model-Based Diagnosis and Future Prospecting

Abduction can also have an important rule in model-based diagnosis. In diagnosis, we are concerned with developing precise methodologies and techniques that can determine whether the behaviour of a system (or artefact) is correct. The artefact can in itself be a knowledge theory, in which case we are trying to ascertain whether the conclusions inferred from the theory are valid.

If the system is not functioning correctly, the diagnosis technique should be able to determine, as accurately as possible, which part of the system is failing, and the kind of fault it is facing. The process of diagnosis is usually based on observations which provide information on the current behaviour of the system.

Model-based diagnosis is an example of abductive reasoning using a model of the system that is to be diagnosed. This model describes the behaviour of the system, or artefact. The model can itself be the subject of diagnosis. This model is usually an abstraction of the behaviour of the system and can be incomplete. The faulty behaviour may also be unknown (e.g. no fault model is represented). Given the set of observations about the system, the diagnoser simulates the system using the model, and compares the observations actually made to the observations predicted by the simulation.

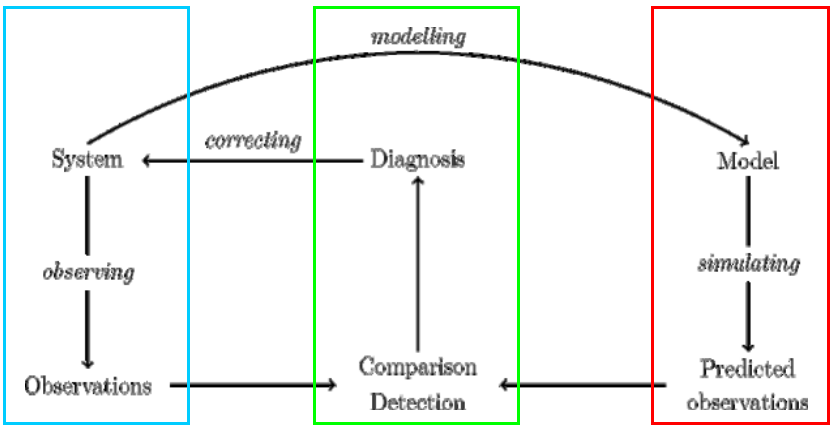

Figure 1: The three components of model-based diagnosis.

Construction of the model can be expressed by rules indicating those cases in which the behaviour of the system is normal or abnormal:

$$Abnormal(S) \Rightarrow InternalBehaviour(Int1), Observable(Obs1).$$
$$\neg Abnormal(S) \Rightarrow InternalBehaviour(Int2), Observable(Obs2).$$

Given the set of observations *Obs*, the problem is to determine whether the system behaviour is normal or not (i.e. $\neg Abnormal(S)$ or $Abnormal(S)$). A system is said to be diagnosable if we are able to determine, without ambiguity, a single diagnosis, independently of the behaviour it exhibits.

However, since the model of our system can be incomplete, this means that there may exist faults and behaviours unaccounted for by the model. Abductive reasoning in the context of evolving logic programs [3, 6, 12] can help us extend the theory to account for unforeseen situations, using expectations to generate the possible scenarios for diagnosis.

This involves a notion of simulation, in which the diagnoser is capable of conjuring up hypothetical *what-if* scenarios and formulating abductive explanations for both external and internal observations. Since we may have multiple possible scenarios to choose from, we need some form of preference specification, which can be either a priori or a posteriori. A priori preferences are embedded in the knowledge representation theory using the framework presented below and can be used to produce the most relevant hypothetical abductions for a given state and observations, in order to conjecture possible future states. A posteriori preferences represent choice mechanisms, which enable the program to commit to one of the hypothetical scenarios engendered by the relevant abductive theories. These mechanisms may trigger additional simulations in order to posit which new information to acquire, so more informed choices can be enacted, in particular by restricting and committing to some of the abductive explanations along the way.

3.1. Differential Medical Diagnosis

In medicine, differential diagnosis is the systematic method physicians use to identify the disease causing a patient's symptoms. Before a medical condition can be treated it must first be correctly diagnosed. The physician begins by observing the patient's symptoms, taking into consideration the patient's personal and family medical history and performing additional examinations if current information is lacking. Then the physician lists the most likely causes. The physician asks questions and performs tests to eliminate possibilities until he or she is satisfied that the single most likely cause has been identified. The term differential diagnosis also refers to medical information specially organized to aid in diagnosis, particularly a list of the most common causes of a given symptom, annotated with advice on how to narrow down the list.

This listing of the most likely causes is clearly an abduction process, supported by initial observations. Encoding this process in logic programming would result in a set of literals and rules representing both the expected causes and the knowledge representation theory leading up to them. This set could be subsequently refined through a method of prospection with a priori and a posteriori preferences, like the one described above. This prospective process can result in the decision to perform additional observations on the external environment, and be iterated.

In this case, the external observations correspond to the disease's signs and other examinations performed by the physician during the process of diagnosis. The abductive explanations correspond to the possible causes (e.g. medical conditions or simply bad habits) responsible for the symptoms. A priori preferences and expectations are necessarily determined according to the patient's symptoms and they can be updated during the course of the prospection mechanism. An appropriate knowledge representation theory can be derived from current medical knowledge about the causes of a given disease.

It being such a good example of abduction, we provide further on a practical example of iterated differential diagnosis, based upon real medical knowledge from the field of dentistry, in order to better illustrate the capabilities of abductive reasoning systems. An encoding of the problem in the ACORDA[1] system will then be proffered, along with results from an interactive diagnosis session.

4. Framework

4.1. Language

Let L be a first order language. A domain literal in L is a domain atom A or its default negation *not* A, the latter expressing that the atom is false by default (CWA). A domain rule in L is a rule of the form:

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

where A is a domain atom and $L_1,...,L_t$ are domain literals. To express preference information, L contains priority rules. Let $N = \{n_{r1},...,n_{rk}\}$ be a name set containing a unique name for every domain rule in L. Given a domain rule r, we write n_r to indicate its name. A priority atom is an atom of the form $n_r < n_u$, where $\{n_r,n_u\} \in N$. $n_r < n_u$ means that rule r is preferred to rule u. We assume that names in N do not include "<" itself. A priority rule in L is a rule of the form:

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

where $n_r < n_u$ is a priority atom and every L_i $(1 \leq i \leq t)$ is a domain literal or a priority literal.

We use the following convention. Given a rule r of the form $L_0 \leftarrow L_1,...,L_t$, we write $H(r)$ to indicate L_0, $B(r)$ to indicate the conjunction $L_1,...,L_t$. We write $B^+(r)$ to indicate the conjunction of all positive literals in $B(r)$, and $B^-(r)$ to indicate the conjunction of all negated literals in $B(r)$.When $t = 0$ we write the rule r simply as L_0.

A (logic) program P over L is a finite set of domain rules and priority rules. Every program P has associated a set of *abducibles* A_P, without rules in P. Abducibles may be thought of as hypotheses that can be used to extend the current theory of the agent, in order to provide hypothetical solutions

1 ACORDA is a self-evolving prospective logic programming system which uses abduction to generate its own possible future scenarios, using a theory of preferences among these hypotheses to guide the process of finding the best explanations for present observations. For more details concerning its implementation see [12].

or possible explanations for given queries.

A 2-valued interpretation M of L is any set of literals from L that satisfies the condition that, for any atom A, precisely one of the literals A or *not* A belongs to M. We say that an interpretation M satisfies a conjunction of literals $L_1,...,L_t$, if every literal L_i in the conjunction belongs to M.

4.2. Declarative Semantics

In the remainder of this section we let P be a program over L, A_P the set of abducibles of P, and M an interpretation of L. We write *least*(P) to indicate the least model of P. We adopt the following two definitions from [9], and Definitions 4 and 5 from [2], and refer the reader to those sources for a detailed exposition.

Definition 1. *The set of default assumptions of P with respect to M is:*

$$Default(P,M) = \{not\ A : \neg\exists r \in P \text{ such that } H(r) = A \text{ and } M \models \mathrm{B(r)}\}$$

Definition 2. *M is a stable model of P iff* $M = least(P \cup Default(P,M))$.

Definition 3. *Let* $\Delta \subseteq A_P$. *M is an abductive stable model with hypotheses* Δ *of P iff:*

$$M = least(P^+ \cup Default(P^+,M)), \text{ where } P^+ = P \cup \Delta.$$

Note that the abducibles in A_P are defined false by default whenever they are not abduced. Given a program P, to compute which of its abductive stable models are preferred according to the priority relation $<$, we remove (from the program) all the unsupported rules together with the less preferred rules defeated by the head of some more preferred one, in a priority rule. Unsupported rules are those whose head is true in the model but whose body is defeated by the model, i.e. some of its default negated atoms are false in it.

Definition 4. *The set of unsupported rules of P and M is:*

$$Unsup(P,M) = \{r \in P : M \models H(r),\ M \models B^+(r) \text{ and } M \not\models B^-(r)\}.$$

Definition 5. $Unpref(P,M)$ *is a set of unpreferred rules of P and M iff:*

$$Unpref(P,M) = least(Unsup(P,M) \cup Q), \text{ where}$$

$$Q = \{r \in P : \exists u \in (P - Unpref(P,M)) \text{ such that } M \models n_u < n_r,\ M \models B^+(u), \text{ and } [\ not\ H(u) \in B^-(r) \text{ or } (not\ H(r) \in B^-(u),\ M \models B(r))\ ]\ \}.$$

A rule r is unpreferred if it is unsupported or there exists a more preferred rule u (which is not itself unpreferred) such that the positive literals in B(u) hold, and r is defeated by u or r attacks (i.e., attempts to defeat) u. Note that only domain rules can be unpreferred since it is required that M $\models$ $n_u < n_r$ holds, where n_r and n_u are names of domain rules.

The following definition introduces the notion of preferred abductive stable model. Given a program P and a set Δ of hypotheses, a preferred abductive stable model with hypotheses Δ of P is a stable model of the program that contains all the hypotheses in Δ, and all those rules in P that are not unpreferred.

Definition 6. *Let* $\Delta \subseteq A_P$ *and M an abductive stable model with hypotheses* Δ *of P. M is a preferred*

abductive stable model with hypotheses Δ *of P iff:*

1. *if* $M \models n_{r1} < n_{r2}$, *then* $M \not\models n_{r2} < n_{r1}$
2. *if* $M \models n_{r1} < n_{r2}$ *and* $M \models n_{r2} < n_{r3}$, *then* $M \models n_{r1} < n_{r3}$
3. $M = least(P^+ - Unpref(P^+, M) \cup Default(P^+, M))$, *with* $P^+ = P \cup \Delta$.

Conditions 1 and 2 state that the preference relation < is required to be a strict partial order. When the language contains only domain rules and priority rules (that is, there are no abducibles), the semantics reduces to the Preferential semantics of Brewka and Eiter [5]. If integrity constraints are introduced, this semantics generalizes to the Updates and Preferences semantics of Alferes and Pereira [2], which extends updatable logic programs with updatable preferences. Our semantics takes the latter, without formally addressing updating, and complements it with modifiable abducibles.

Definition 7. *An abductive explanation for a query G is any set* $\Delta \subseteq A_P$ *of hypotheses such that there exists a preferred abductive stable model M with hypotheses* Δ *of P for which* $M \models G$.

A program may have several abductive explanations for a query *G*.

5. Preferring Abducibles

The evaluation of alternative explanations is a central problem in abduction, because of the combinatorial explosion of possible explanations to handle. Thus, it is important to generate only the explanations that are relevant for the problem at hand. Several approaches have thus far been proposed, often based on some global criteria, which has the drawback of generally being domain independent and computationally expensive. An alternative to global criteria for competing alternative assumptions is to allow the theory to contain rules encoding domain specific information about the likelihood that a particular assumption is 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 one of the rules 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 there is not an expectation to the contrary (expect_not).

To express preference criteria among abducibles, we introduce the language L^*. A relevance atom is one of the form $a \lhd b$, where a and b are abducibles. $a \lhd b$ means that the abducible a is more relevant than the abducible b. A relevance rule is a rule of the form:

$$a \lhd b \leftarrow L_1,...,L_t \ (t \geq 0)$$

where $a \lhd b$ is a relevance atom and every L_i $(1 \leq i \leq t)$ is a domain literal or a relevance literal. Let L^* be a language consisting of domain rules and relevance rules.

Example 1. Consider a situation where an agent Claire drinks either tea or coffee (but not both). Suppose that Claire prefers coffee over tea when sleepy. This situation can be represented by a program Q over L^* with set of abducibles $A_Q = \{tea, coffee\}$:

drink ← tea
drink ← coffee
expect(tea)

expect(coffee)
expect_not(coffee) ← blood_pressure_high
coffee ◁ tea ← sleepy

Having the notion of expectation allows one to express the preconditions for an expectation or otherwise about an assumption *a*, and express which possible expectations are *confirmed* (or go through) in a given situation. If the preconditions do not hold, then expectation *a* cannot be confirmed, and therefore *a* will never be assumed. By means of *expect_not* one can express situations where one does not expect something. In this case, when blood pressure is high, coffee will not be confirmed or go through because of the contrary expectation arising as well (and therefore tea will be assumed).

The following definition exploits the relevancy relation ◁ of a program Q to distinguish which of its abductive stable models are relevant.[2]

Definition 8. *Let Q be a program over L^* with set of abducibles A_Q and M an interpretation of L^*. Let $\Delta \subseteq A_Q$. M is a relevant abductive stable model of Q with hypothesis Δ iff:*

1. *for every $x,y \in A_Q$, if $M \models x \lhd y$ then $M \not\models y \lhd x$*
2. *for every $x,y,z \in A_Q$, if $M \models x \lhd y$ and $M \models y \lhd z$, then $M \models x \lhd z$*
3. *$M = least(Q^+ \cup Default(Q^+, M))$, with $Q^+ = Q \cup \Delta$*
4. *Δ is the empty set or a singleton $\Delta = \{a\}$, for some $a \in A_Q$*
5. *if $\Delta = \{a\}$, then $M \models expect(a)$ and $M \not\models expect\_not(a)$*
6. *if $\Delta = \{a\}$, then there exists no relevance rule r in Q such that:*
   - *$H(r)$ is $x \lhd a$*
   - *$M \models H(r)$*
   - *$M \models expect(x)$ and $M \not\models expect\_not(x)$*

By allowing Δ to be the empty set, or a singleton (condition 4 above) we are stating that the abducibles in A_Q are mutually exclusive. This notion can be generalized to sets of abducibles, by an adequate adaptation of the preference order to one among such sets. It is possible, however, to consider that a set of literals is, in itself, an abducible, so we could even express different interesting combinations of multiple literals for a given problem domain.

Example 2. Let Q be the program of Example 1. Q has two alternative explanations $\Delta_1 = \{coffee\}$ and $\Delta_2 = \{tea\}$ for the query *drink*. In fact, Q has two relevant abductive stable models:

$M_1 = \{expect(tea), expect(coffee), coffee, drink\}$ with hypotheses Δ_1

$M_2 = \{expect(tea), expect(coffee), tea, drink\}$ with hypotheses Δ_2

for which $M_1 \models drink$ and $M_2 \models drink$. The number of models reduces to one if we add *sleepy* to Q. In this case, *coffee* being an abducible more relevant than *tea* and consequently the only relevant model of Q is $M_1 \cup \{sleepy\}$.

In order to obtain a proof procedure for the language L^*, a syntactical transformation Σ that maps programs over L^* into programs over L has been given in [6], and its soundness demonstrated. The next example illustrates its use.

---

2 See [6] for a more detailed exposition and implementation of the abductive stable models semantics in logic programming.

Example 3. Let Q be the program of Example 1. The transformation Σ maps Q into the program P with abducibles $A_P = \{abduce\}$:

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*

The role of the abducible *abduce* is to enact the assumption of one of the alternative assumptions *tea* or *coffee* needed to prove *drink*. The rules (1) and (2) code the alternative assumptions *tea* and *coffee* into cycles over negation. Rule (1) says that *coffee* can be assumed if *abduce* has been abduced, *tea* is not assumed, and coffee is confirmed. The last rule in P is a priority rule stating that rule (1) is preferable to rule (2) if *sleepy* holds. P has two preferred abductive stable models with hypotheses Δ = {abduce}:

M_1 = {*abduce*, *confirm*(*tea*), *confirm*(*coffee*), *expect*(*tea*), *expect*(*coffee*), *coffee*, *drink*}

M_2 = {*abduce*, *confirm*(*tea*), *confirm*(*coffee*), *expect*(*tea*), *expect*(*coffee*), *tea*, *drink*}

The number of preferred abductive stable models reduces to one if *sleepy* holds. In that case, the unique preferred abductive stable model would be:

M_3 = {*abduce, confirm*(*tea*), *confirm*(*coffee*), *expect*(*tea*), *expect*(*coffee*), *coffee*, *drink*, *1 < 2*}

6. Exploratory Data Analysis

Another application of expressing preferences over abducibles is that of *exploratory data analysis*, which 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 that Bob has three possible hypotheses (abducibles) *a*, *b*, *c*, capable of explaining it. In exploratory data analysis, after observing some new facts, we abduce explanations and explore them to check predicted values against observations. Though there may be more than one convincing explanation, we abduce only the more plausible of them. The next example illustrates exploratory data analysis.

Example 4. Let the program Q over L^*, with abducibles $A_Q = \{a,b,c\}$, be the theory of agent Bob:

$x \leftarrow a$ $x \leftarrow b$ $x \leftarrow c$ $expect(a)$ $expect(b)$ $expect(c)$ $a \lhd c \leftarrow not\ e$ $b \lhd c \leftarrow not\ e$ $b \lhd a \leftarrow d$	meaning: x - the car does not start a - the battery has problems b- the ignition is damaged c - there is no gasoline in the car d - the car's radio works e - Bob's wife has used the car exp - test if the car's radio works.

Q has two relevant abductive stable models capable of explaining observation x:

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

$M_2 = \{expect(a), expect(b), expect(c), a \lhd c, b \lhd c, b, x\}$ with hypothesis $\Delta_2 = \{b\}$

In this example, we have only a partial relevancy theory over abducibles. Thus, we cannot select exactly one abducible (i.e., one model), as it were the case had we a complete relevancy relation over all abducibles in A_Q. To prefer between a and b, one can perform some experiment exp to obtain confirmation (by observing the environment) about the most plausible hypothesis. To do so, we can employ active rules that are rules of the form:

$$L_1,...,L_t \Rightarrow \alpha : A$$

where $L_1,...,L_t$ are domain literals, and $\alpha : A$ is an action literal. This rule states to update the theory of an agent α with A if its body $L_1,...,L_t$ is satisfied in all relevant abductive stable models. For example, we can add the following rules (where *env* plays the role of the environment) to the theory Q of Bob:

$choose \leftarrow a$
$choose \leftarrow b$

$a \Rightarrow Bob : chosen$
$b \Rightarrow Bob : chosen$
$choose \Rightarrow Bob : (not\ \ chosen \Rightarrow env : exp)$

Initially *Bob* has two hypotheses, a and b, that are capable of explaining the observed phenomena x. Hence, *Bob* must discover the correct one. *Bob* chooses some hypothesis if a or b hold. In this case, *Bob* still has two relevant abductive stable models: $M_3 = M_1 \cup \{choose\}$ and $M_4 = M_2 \cup \{choose\}$. As *choose* holds in both models, the last active rule is triggerable. When triggered, it will add (at the next state) the active rule $not\ \ chosen \Rightarrow env : exp$ to the theory of *Bob*, and, if $not\ \ chosen$ holds, *Bob* will perform the experiment *exp*. The first two active rules are needed to prevent *Bob* from performing *exp* when *Bob* has chosen one of the abducibles.

7. Modelling Observations

In order to allow specification of model-based diagnosis using the proposed preferential theory framework, it is useful to provide a means to encode observations in the system, both external and

internal, respectively, those experiments that the system can perform on the environment, and the observations for which it wants to find a reasonable explanation. The notion of *observable* is used to represent knowledge about such observations.

An observable is a quaternary relation amongst the *observer*, i.e. that which is performing the observation; the *observed*, that which is the target of the observation; the result of the *observation* itself; and the *truth value* associated with the observation. This relation is adequate both to express program based self-triggering of internal queries, as well as observations requested from an exterior oracle to the program, and from the program directly to the environment. For example, the observable

observable(prog,prog,Query,true)

represents an observation in which the observer is the program, that which is observed is also the program, the observation is the specified Query and the positive truth value means the observation must be proven true. In this case, we expect that such an observable, by becoming active via special on_observable rules, triggers the self-evaluation of Query in the current knowledge state, in the process resorting to any relevant and expected abducibles that can account for the observation.

Under certain conditions, these observations can represent not only experiments performed on the environment, but also questions to external agents to acquire additional information with which to prefer among relevant hypotheses. These agents are called *oracles*, and they can be just about anything that is able to interact with the system in order to provide additional information, be it facts or rules. This may represent an instance of learning by being told, with the agent being informed of new pieces of knowledge which allow it to make better choices. The oracle-agent relation can be further complicated in order to include an encoding of levels of trust, or even limited time-spans for the validity of the obtained information, after which further checks will be performed in order to ascertain whether (or to what extent) the knowledge has proven itself useful.

For now, information derived from oracles can be regarded as an *event*, following the concept introduced in the language for updates LUPS [4]. In the more declarative semantics of updates, EVOLP [3], we can consider an event as an update of a literal or rule which is immediately followed by a deletion of that very same literal or rule.

8. Differential Diagnosis in Dentistry: A Use Case

The process of generating possible explanations to given observations and committing to the resulting choices can be integrated on a larger computational reasoning system, as the one described in Figure 2. Starting from some state of an evolving knowledge base, we begin by selecting the observations that we would like to explain. We are then able to come up with possible abductive hypotheses that allow us to explain those observations. A priori and a posteriori preferences are used to filter out the most relevant explanatory scenarios, allowing once again the realization of experiments to disambiguate among abducibles.

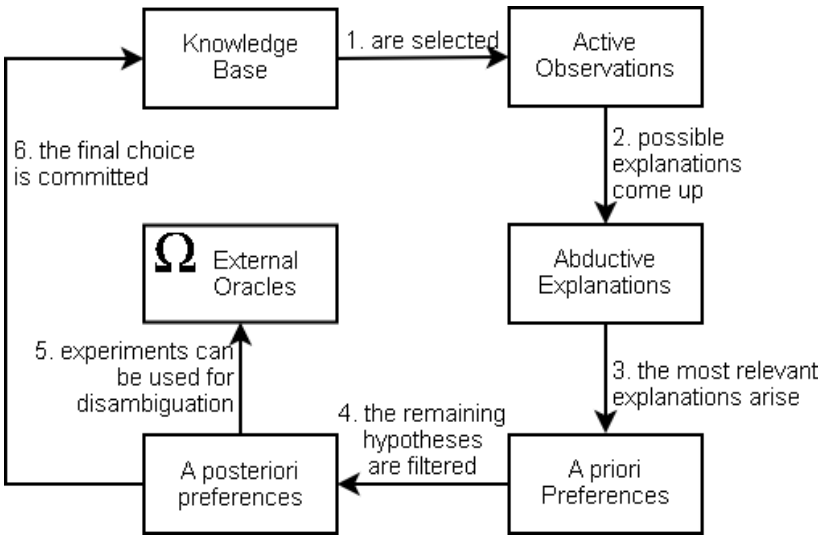

Figure 2: An evolving abductive knowledge base.

Considering such an evolving logic system, we are now ready to tackle the process of encoding a real-life situation of medical differential diagnosis using the proposed framework. Consider the following setting from the field of dentistry.

Example 5. A patient shows up at the dentist with signs of pain upon teeth percussion. The expected causes for the observed signs are:

- Periapical lesion (endodontic or periodontal source)
- Horizontal Fracture of the root and/or crown
- Vertical Fracture of the root and/or crown

Several additional examinations can be conducted to determine the exact cause, namely:

- X-Ray for determination of radiolucency or fracture traces
- X-Ray for determination of periapical lesion source
- Check for tooth mobility
- Measurement of gingival pockets

Aside from presenting multiple hypotheses for diagnosis, the knowledge exhibited by the practitioner must necessarily evolve in time, as he or she performs relevant examinations which will attempt to disqualify all but one of the possible explanations. Current examinations depend on knowledge acquired in the past, which, in turn, will end up influencing the observations and inferences which will be drawn in the future. Developing a system that is capable of modelling the evolution of a program in order to draw such inferences is a demanding but obviously useful challenge.

This setup can be intuitively modelled on top of an abductive logic program:

Initial Signs

$$\text{on\_observable(prog,prog,percussion\_pain\_cause)} \leftarrow \text{percussion\_pain.} \qquad \text{(A)}$$

First Phase of Differential Diagnosis

$$\text{percussion\_pain\_cause} \leftarrow \text{periapical\_lesion.}$$
$$\text{percussion\_pain\_cause} \leftarrow \text{vertical\_fracture.}$$

percussion_pain_cause ← horizontal_fracture.

periapical_lesion ← confirm(periapical_lesion).
expect(periapical_lesion) ← profound_caries.
expect(periapical_lesion) ← on_observable(prog,prog,percussion_pain_cause). (B)
expect_not(periapical_lesion) ← fracture_traces, not radiolucency.

vertical_fracture ← confirm(vertical_fracture).
expect(vertical_fracture) ← on_observable(prog,prog,percussion_pain_cause).
expect_not(vertical_fracture) ← radiolucency, not fracture_traces.

horizontal_fracture ← confirm(horizontal_fracture).
expect(horizontal_fracture) ← on_observable(prog,prog,percussion_pain_cause).
expect_not(horizontal_fracture) ← radiolucency, not fracture_traces.

periapical_lesion ◁ horizontal_fracture ← profound_caries, not percussion_pain.
periapical_lesion ◁ vertical_fracture ← profound_caries, not percussion_pain.

horizontal_fracture ◁ vertical_fracture ← low_mobility.
vertical_fracture ◁ horizontal_fracture ← high_mobility.

Second Phase of Differential Diagnosis

on_observable(prog,prog,periapical_lesion_source) ← periapical_lesion. (C)

periapical_lesion_source ← endodontic_lesion.
periapical_lesion_source ← periodontal_lesion.

endodontic_lesion ← confirm(endodontic_lesion).
expect(endodontic_lesion) ← on_observable(prog,prog,periapical_lesion_source).
expect(endodontic_lesion) ← devitalization.
expect(endodontic_lesion) ← not gingival_pockets.
expect_not(endodontic_lesion) ← gingival_pockets, not devitalization. (D)

periodontal_lesion ← confirm(periodontal_lesion).
expect(periodontal_lesion) ← on_observable(prog,prog,periapical_lesion_source).
expect(periodontal_lesion) ← devitalization.
expect(periodontal_lesion) ← gingival_pockets.
expect_not(periodontal_lesion) ← neg_gingival_pockets.

periodontal_lesion ◁ endodontic_lesion ← gingival_pockets.

Available Experiments

radiolucency ← observable(prog,xray,radiolucency).
fracture_traces ← observable(prog,xray,fracture_traces). (E)
high_mobility ← observable(prog,mobility_check,high_mobility).
low_mobility ← observable(prog,mobility_check,low_mobility).
gingival_pockets ← observable(prog,pockets_check,gingival_pockets).
neg_gingival_pockets ← observable(prog,pockets_check,gingival_pockets,false).
devitalization ← observable(prog,periapical_xray,devitalization).

Available Oracles

observable(prog,xray,Q,S) ← oracle,prolog((oracleQuery(xray(Q),T),S = T)).
observable(prog,mobility_check,Q,S) ←
oracle,prolog((oracleQuery(mobility_check(Q),T),S = T)).
observable(prog,pockets_check,Q,S) ←
oracle,prolog((oracleQuery(pockets_check(Q),T),S = T)).
observable(prog,periapical_xray,Q,S) ←
oracle,prolog((oracleQuery(periapical_xray(Q),T),S = T)).

General Confirmation Rule

confirm(X) ← expect(X), not expect_not(X).

8.1. Interaction Results

Using the above program as the initial knowledge base, we can provide the patient's signs (i.e. *percussion_pain*) as external updates to the system. Running an abduction cycle over the ACORDA system produces the reasoning steps detailed below. The question marks are used to indicate queries to an external oracle, followed by the answer instances that the oracle generated. Next to each evaluation step we present the number codes relating to the phases from Figure 2.

1. About to launch new introspection on selected active observables:
   [on_observable(prog,prog,percussion_pain_cause)]

2. Relevant abducibles for current introspection:
   [horizontal_fracture,periapical_lesion,vertical_fracture]

3. Partial models remaining after a priori preferences:
   [[horizontal_fracture],[periapical_lesion],[vertical_fracture]]

4. About to launch new introspection on selected active observables:
   [on_observable(prog,prog,percussion_pain_cause)]

5. Confirm observation: xray(fracture_traces) (true, false or unknown)? false.

5. Confirm observation: xray(radiolucency) (true, false or unknown)? true.

5. Relevant abducibles for current introspection: [periapical_lesion]

5. Partial models remaining after a priori preferences: [[periapical_lesion]]

The event *percussion_pain* triggers the active observable *percussion_pain_cause* (cf. rule A) which is a program-to-program generated observation, that is, it will cause a top-down query to be launched in order to attempt an explanation of the observable. The system goes down the rule derivation tree for *percussion_pain_cause*, encountering the different possible alternatives derived from the expectation and counter-expectation rules, i.e. the literals that are output in 2. In this case, the expectations are triggered only in the context of the currently active observation (cf. rule B).

Afterwards, the a priori preferences concerning the relevant abducibles come into play in order to filter out less preferred hypothesis. These preferences can themselves be supported by abducibles or further observations. The abductive stable models of the program are then computed, resulting in at

least a model for each abducible that could not be defeated by the contextual preference rules.

In our current case, no relevant preferences are present in the model, so the resulting stable models correspond to all the confirmed abducibles: *periapical_lesion*, *horizontal_fracture* and *vertical_fracture*. Since the system was not able to abduce just one model from a priori preferences, it means that the current information is insufficient to provide an adequate informed choice. The prospective search is then relaunched in order to acquire new information from the available oracles, in order to search for ways to defeat some of the models.

This means that the oracle mechanisms are activated, and for the new observations the system decrees that experiments can now be performed. A new top-down query is launched, and as the derivation tree is traversed for new confirmation of relevant abducibles, opportunities to perform experiments are encountered. This is the case for the satisfaction of *expect_not*(*periapical_lesion*) which depends on *fracture_traces* and *not radiolucency*.

In an attempt to satisfy *fracture_traces*, the ACORDA encounters an observable clause (cf. rule E). This clause depends on fracture traces being found on X-Ray imaging, so it probes the environment for just such an exam. No fracture traces are found, so periapical lesion can be confirmed under support from the oracle experiment. However, the subsequent experiment for *radiolucency* reveals positive results which can guarantee defeat of the other two abducibles, meaning that *periapical_lesion* is now the only expected and confirmed abducible. ACORDA can now successfully commit to a single diagnosis.

In the cases where the system cannot guarantee a single abductive stable model after exhausting all the introspections, it can query the user to perform the final choice over the list of surviving abducibles, or to provide additional information or experiments to perform in order to continue the diagnosis process. This behaviour could even be refined to launching separate simulations assuming each of the remaining solutions, and performing extra levels of prospective lookahead for additional ways to defeat further models. This addition is considered in future work.

After the abduction of periapical lesion as the most likely cause for the sign of percussion pain on the patient's tooth, the attention of the system turns to satisfy a new observation for the source of such lesion. As such, a new cycle of introspection on top of these results can produce a more detailed diagnosis, as depicted below:

1. About to launch new introspection on selected active observables:
   [on_observable(prog,prog,periapical_lesion_source)]

2. Relevant abducibles for current introspection: [endodontic_lesion,periodontal_lesion]

3. Partial models remaining after a priori preferences:
   [[endodontic_lesion],[periodontal_lesion]]

4. About to launch new introspection on selected active observables:
   [on_observable(prog,prog,periapical_lesion_source)]

5. Confirm observation: pockets_check(gingival_pockets) (true, false or unknown)? true.

5. Confirm observation: periapical_xray(devitalization) (true, false or unknown)? true.

5. Relevant abducibles for current introspection: [endodontic_lesion,periodontal_lesion]

5. Partial models remaining after a priori preferences: [[periodontal_lesion]]

This time, the commitment to the abduction of *periapical_lesion* triggers the activation of program-to-program observable *periapical_lesion_source* (cf. rule C). The relevant expected abducibles that satisfy this observation are *endodontic_lesion* and *periodontal_lesion*, both being confirmed under the current knowledge state. Again, no active preferences hold a priori, that is, there are no preference rules relevant to the current abducibles which do not depend on any external observations. As a result, the Abductive Stable Models of the program correspond once again to all available abducibles.

The choice mechanisms are activated and the top goal is relaunched in an attempt to acquire additional information, activating the oracles for external environment probing. The attempt to defeat *endodontic_lesion* calls for a gingival pockets measurement (cf. rule D), which reveals the existence of pockets in the vicinity of the patient's tooth. On the other hand, a periapical X-Ray is needed to confirm that no endodontic therapy was performed on that tooth. This experiment, however, reveals just that, and so ACORDA is unable to defeat the *endodontic_lesion* abducible using this *expect_not* clause.

Attempting to defeat *periodontal_lesion* by means of *expect_not* clauses proves impossible as well, due once again to the ambiguous results from the experiments. Both abducibles are again confirmed. However, a preference for *periodontal_lesion* is now in place, given the existence of gingival pockets, so a unique Abductive Stable Model emerges, yielding the final diagnosis of *periodontal_lesion*.

The system could now be extended to handle different treatment hypotheses according to the result of the diagnosis. It would just be a matter of adding new triggers for additional observations that would represent the adequate treatment. Abducibles in this case would be the expected treatments and the preference model could be extended to include the patient's own preferences.

9. Revising Relevancy Relations

Relevancy relations are subject to be modified when new information is brought to the knowledge of an individual, or 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, via abductive diagnosis.

Example 6. Consider the boolean composition of two relevancy relations, that is, $\lhd = \lhd_1 \cup \lhd_2$. The relevancy relation $\lhd$ may not be a strict partial order, since antisymmetry and transitivity are not necessarily preserved. Consider the following program Q over L^* with abducibles $A_Q = \{a,b,c\}$:

$x \leftarrow a$
$x \leftarrow b$
$x \leftarrow c$
$expect(a)$
$expect(b)$
$expect(c)$

$\underline{u} \lhd \underline{v} \leftarrow \underline{u} \lhd_1 \underline{v}$
$\underline{u} \lhd \underline{v} \leftarrow \underline{u} \lhd_2 \underline{v}$
$a \lhd_1 b$
$b \lhd_1 c$

$b \lhd_2 a$

where $\underline{u}$ and $\underline{v}$ are variables ranging over the abducibles in A_Q. The program Q does not have any relevant abductive stable model since $\lhd$ is not a strict partial order and therefore conditions 1 and 2 of Definition 8 are not met.

In order to account revising relevancy relations, we introduce the language L^+ extending L^* to contain integrity constraints.

An integrity constraint is a rule of the form:

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

where $\bot$ is a domain atom denoting contradiction, and $L_1,...,L_t$ are domain or relevance literals. The language L^+ consists of domain rules, relevance rules, and integrity constraints. In L^+ there are no abducibles, and therefore its meaning is characterized in terms of stable models. Given a program T over L^+ and a literal L, we write $T \models L$ if L is true in every stable model of T. The program T is contradictory if $T \models \bot$.

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 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 $<U, I>$ where $U \cap I = \emptyset$, $U \subseteq C$ and $I \subseteq T_c$. We say that D is a diagnosis for T iff $(T - I) \cup U \models \bot$. $D = <U, I>$ is a minimal diagnosis if there exists no diagnosis $D_2 = <U_2, I_2>$ for T such that $(U_2 \cup I_2) \subset (U \cup I)$.

Example 7. Consider the following extension of the previous program Q, now expressed over L^+ with the introduction of integrity constraints:

$x \leftarrow a$
$x \leftarrow b$
$x \leftarrow c$
$expect(a)$
$expect(b)$
$expect(c)$

$\underline{u} \lhd \underline{v} \leftarrow \underline{u} \lhd_1 \underline{v}$
$\underline{u} \lhd \underline{v} \leftarrow \underline{u} \lhd_2 \underline{v}$
$a \lhd_1 b$
$b \lhd_1 c$
$b \lhd_2 a$

$\bot \leftarrow \underline{u} \lhd \underline{u}$
$\bot \leftarrow \underline{u} \lhd \underline{v}, \underline{v} \lhd \underline{u}$
$\bot \leftarrow \underline{u} \lhd \underline{v}, \underline{v} \lhd \underline{z}, not \ \underline{u} \lhd \underline{z}$

In this case, it holds that $T \models \bot$. Let $C = \{C_1, C_2\}$. T admits three minimal diagnoses:

$D_1 = <\{\}, \{a \lhd_1 b\}>$

$D_2 = <\{\}, \{b \lhd_1 c, b \lhd_2 a\}>$
$D_3 = <\{a \lhd_1 c\}, \{b \lhd_2 a\}>$

To compute the minimal diagnoses of a contradictory program T, we employ the contradiction removal method presented in [14], adapted here to handle relevancy relations. The contradiction removal method is based on the idea of revising (to false) some of the default atoms *not A*. A default atom *not A* can be revised to false by simply adding A to T. According to [14] the default literals *not A* that are allowed to change their truth value are those for which there exists no rule in T defining A. Such literals are called revisable.

Definition 9. *Let T be a program over L^+. An atom $A \neq \bot$ is a revisable of T iff there is no rule defining A in T.*

Definition 10. *Let T be a program over L^+ and V a set of revisables of T. A set $Z \subseteq V$ is a revision of T iff $T \cup Z \not\models \bot$.*

In order to apply the algorithm in [14], we need to transform our original program to one equivalent program which is suitable for contradiction removal.

Definition 11. *Let T be a program over L^+ and C a set of predicate symbols in L^+. The transformation Γ that maps T into a program T' is obtained by applying to T the following two operations:*

- *Add not incorrect($A \leftarrow Body$) to the body of each rule $A \leftarrow Body$ in T_c.*
- *Add the rule $p(x_1,...,x_n) \leftarrow uncovered(p(x_1,...,x_n))$ for each predicate symbol p with arity n in C, where $x_1,...,x_n$ are variables.*

We assume the predicate symbols *incorrect* and *uncovered* do not occur in T.[3]

Example 8. Let T be the program of Example 7. Then, the program $\Gamma(T)$ is:

$x \leftarrow a$
$x \leftarrow b$
$x \leftarrow c$
$expect(a)$
$expect(b)$
$expect(c)$

$\underline{u} \lhd \underline{v} \leftarrow \underline{u} \lhd_1 \underline{v}$
$\underline{u} \lhd \underline{v} \leftarrow \underline{u} \lhd_2 \underline{v}$

$\bot \leftarrow \underline{u} \lhd \underline{u}$
$\bot \leftarrow \underline{u} \lhd \underline{v}, \underline{v} \lhd \underline{u}$
$\bot \leftarrow \underline{u} \lhd \underline{v}, \underline{v} \lhd \underline{z}, not\ \underline{u} \lhd \underline{z}$

$a \lhd_1 b \leftarrow not\ incorrect(a \lhd_1 b)$
$b \lhd_1 c \leftarrow not\ incorrect(b \lhd_1 c)$
$b \lhd_2 a \leftarrow not\ incorrect(b \lhd_2 a)$

3 The proof procedure for the correctness of the transformation Γ is detailed in [6].

$\underline{u} \lhd_1 \underline{v} \leftarrow uncovered(\underline{u} \lhd_1 \underline{v})$
$\underline{u} \lhd_2 \underline{v} \leftarrow uncovered(\underline{u} \lhd_2 \underline{v})$

$\Gamma(T)$ admits three minimal revisions with respect to the revisables of the form *incorrect*(.) and *uncovered*(.):

$Z_1 = \{incorrect(a \lhd_1 b)\}$
$Z_2 = \{incorrect(b \lhd_1 c), incorrect(b \lhd_2 a)\}$
$Z_3 = \{uncovered(\text{a} \lhd_1 c), incorrect(b \lhd_2 a)\}$

The following result relates the minimal diagnoses of a program T with the minimal revisions of $\Gamma(T)$.

Theorem 1. *Let T be a program over* L^+. *The pair* $D = <U, I>$ *is a diagnosis for T iff*

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

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

To compute the minimal diagnosis of a program T we consider the transformed program $\Gamma(T)$ and compute its minimal revisions. An algorithm for computing minimal revisions in such logic programs is given in [14].

10. Concluding Remarks

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 with updates. These programs are executable by means of publicly available state-of-the-art systems, using available transformations provided here and elsewhere [14]. We have furthermore shown how preferences about knowledge extensions 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.

Although we have based our approach on the Stable Model semantics, we could just as easily have used the 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* [15]. Indeed, when there are odd loops over default negation in a program, Stable Model semantics does not afford the program with semantics.

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.

The abductive process and the system of a priori preferences can be improved as well, in order to allow for the abduction of a set of abducibles. It will be necessary to specify new ways in which we can express preferences over these sets, but work is already well underway in this regard.

Abduction of multiple literals is, however, already possible in the current system, by making single abducibles themselves stand for sets of "abducible" literals. One just needs to carefully model all the relevant combinations of literals that one would be inclined to expect.

Concerning model-based diagnosis, the integration of actions as possible abductions is also a must in order to tackle even more complex problems and applications, especially in diagnosis and prospective logic programming, as we must necessarily deal with pre- and post-conditions. Namely, the pre-conditions for an action to be abduced must be evaluated before the update for the action actually takes place, but the post-conditions must also be taken in consideration during the simulation and before the real action is executed.

Preferences over observables will also be desirable, since not every observation costs the same for the agent. Performing an X-Ray costs more than checking for tooth mobility or gingival pockets, and these differences should be modelled directly in the system. It would also be interesting to study more general ways of selecting the most interesting internal observations to pay attention to at each diagnosis step. Furthermore, abductive reasoning can be used to generate hypotheses of observations of events possibly occurring in the future along the lines of [1]. This ability will allow us to foresee the occurrence of new events which may or may not occur within some time interval, and thereby confirm or disprove an abduced explanation.

In [17], 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. [7] 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. 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, all the while being the very vehicle that enables testing its specification, when not outright its very implementation [16]. Consequently, it merits sustained attention from the community of researchers addressing the issues we have considered and outlined.

11. Acknowledgements

We thank Gregory Wheeler for extended discussion and comment on the philosophical material in this paper. We thank Joana Nogueira for preparing and supplying scientific information for the case study on differential medical diagnosis, and also a number of colleagues at DEIS, U. Bologna, for prior stimulating discussions.

References

1. M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Abduction with Hypotheses Confirmation. Poster paper. *Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI-05)*, pages 1545-1546, 2005.
2. J. J. Alferes and L. M. Pereira. Updates plus preferences. In M. O. Aciego, I. P. de Guzmán, G. Brewka, and L. M. Pereira, editors, *Logics in AI, Procs. JELIA'00*, LNAI 1919, pages 345–360, Springer 2000..
3. J. J. Alferes, A. Brogi, J. A. Leite, L. M. Pereira. Evolving logic programs. In S. Flesca, S. Greco, N. Leone, G. Ianni, editors, *Procs. of the 8th European Conf. on Logics in Artificial Intelligence (JELIA'02)*, LNCS 2424, pages 50–61, Springer, 2002.
4. J. J. Alferes, L. M. Pereira, H. Przymusinska and T. C. Przymusinski. LUPS – a language for updating logic programs. *Artificial Intelligence*, 138(1–2), 2002.
5. G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109:297–356, 1999.
6. P. Dell'Acqua and L. M. Pereira. Preferential theory revision. *Journal of Applied Logic.* Forthcoming, 2007.
7. Jon Doyle. Prospects for preferences. *Computational Intelligence*, 20(2):111–136, 2004.
8. Pierre Duhem. *The Aim and Structure of Physical Theory*. Princeton University Press, 2nd edition, 1954.
9. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. A. Bowen, editors, *ICLP'88*, pages 1070–1080. MIT Press, 1988.
10. Sven Hansson. Ten philosophical problems in belief revision. *Journal of Logic and Computation*, 13:37–49, 2003.
11. Isaac Levi. *Mild Contraction*. Clarendon Press, Oxford, 2004.
12. G. Lopes and L. M. Pereira. Prospective Logic Programming with ACORDA. In *Empirically Successful Computerized Reasoning (ESCoR'06)* workshop at the 3rd International Joint Conference on Automated Reasoning (IJCAR'06), August 2006.
13. Nayak. Iterated belief change based on epistemic entrenchment. *Erkenntnis*, 41:353–390, 1994.
14. L. M. Pereira, C. Damásio and J. J. Alferes. Debugging by Diagnosing Assumptions. In P. Fritzson, editor, *1st Int. Ws. on Automatic Algorithmic Debugging, AADEBUG'93*, LNCS 749, pages 58–74, Springer, 1993.
15. L. M. Pereira and A. M. Pinto. Revised Stable Models - a Semantics for Logic Programs. *12th Portuguese Intl. Conf. on Artificial Intelligence (EPIA'05)*, LNAI 3808, pages 29–42, Springer, 2005.
16. L. M. Pereira, Philosophical Incidence of Logical Programming, in: Handbook of the Logic of Argument and Inference, D. Gabbay et al. (eds.), pages 425–448, Studies in Logic and Practical Reasoning series, volume 1, Elsevier Science 2002.
17. Hans Rott. *Change, Choice and Inference*. Oxford University Press, Oxford, 2001.
18. Wolfgang Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In William L. Harper and Brian Skyrms, editors, *Causation in Decision, Belief Change and Statistics*, volume 2, pages 105–134, Reidel, 1987.