Abduction with Hypotheses Confirmation*

Marco Alberti[1], **Marco Gavanelli**[1], **Evelina Lamma**[1], **Paola Mello**[2], **and Paolo Torroni** [2]
[1] ENDIF - University of Ferrara - Via Saragat, 1 - 44100 Ferrara, Italy.
{malberti|m gavanelli|elamma}@ing.unife.it
[2] DEIS - University of Bologna - Viale Risorgimento, 2 - 40136 Bologna, Italy.
{pmello|ptorroni}@deis.unibo.it

1 Introduction

Abductive Logic Programming (ALP) is an extension of Logic Programming to formalise hypothetical reasoning. It typically distinguishes between facts, defined within a static theory and known to be true or false, and potentially true, abducible atoms (hypotheses).

However, in most applications, this distinction is not adequate to capture the dynamics of knowledge, as reasoning is confronted with an environment in evolution. It may turn out that some hypotheses gain or lose strength, as events happen in the world. For example, in the medical domain, where a kind of hypothetical reasoning is the diagnostic inference, we can model symptoms as observations and diseases as (not directly measurable) hypotheses: test results and new possibly upcoming symptoms, instead, are none of these, and should be interpreted as hypotheses confirmable by events.

An abductive derivation verifies a goal by using deduction as in logic programming, but also by possibly assuming that some abducibles are true. In order to have this process converge to a meaningful explanation, an abductive theory normally comes together with a set of *integrity constraints* IC, and it is required that hypotheses be consistent with IC.

In our extended framework, we distinguish between two classes of abducible literals: *hypotheses*, as classically understood, and *expectations* about events. Expectations can be "positive" (to be confirmed by certain events occurring), or "negative" (to be confirmed by certain events not occurring).

We propose a new language to define abductive logic programs with expectations, inspired to the IFF proof procedure [Fung and Kowalski, 1997], and whose semantics extends those of classical ALP. The language permits to express abducible hypotheses and expectations with *variables* and *constraints*. Within this new framework we can model and reason about a number of concrete application scenaria. Our framework permits, e.g., to reason about deadlines, and to express and correctly handle expectations with universal quantification: this typically happens with negative expectations ("The patient is expected *not* to show symptom Q *at all times*").

*This work is partially funded by the Information Society Technologies programme of the European Commission under the IST-2001-32530 SOCS Project, and by the MIUR COFIN 2003 projects *Sviluppo e verifica di sistemi multiagente basati sulla logica*, and *La Gestione e la negoziazione automatica dei diritti sulle opere dell'ingegno digitali: aspetti giuridici e informatici*.

The operational semantics is an abductive proof-procedure which builds on the IFF and is proven sound for a relevant class of programs. It has been implemented using *Constraint Handling Rules* and integrated in a Java-based system for hypothetical reasoning [Alberti *et al.*, 2005].

2 Dynamic ALP

An ALP [Kakas *et al.*, 1992] is a tuple $\langle KB, IC, \mathcal{A}\rangle$ where KB is a logic program, $\mathcal{A}$ is a set of predicates called *abducibles* not defined in KB, and IC is a set of formulae called *Integrity Constraints*. An abductive explanation for a goal G is a set $\Delta \subseteq \mathcal{A}$ s.t. $KB \cup \Delta \models G$ and $KB \cup \Delta \models IC$.

New dynamically upcoming events are encoded into atoms $\mathbf{H}(Descr[, Time])$ where $Descr$ is a ground term representing the event and $Time$ is an integer representing the time at which the event happened. Such events are recorded into a set (called history, or $\mathbf{HAP}$) containing $\mathbf{H}$ atoms. A Dynamic Abductive Logic Program (DALP) is a sequence of ALPs, each grounded on a given history. We will write $DALP_{\mathbf{HAP}}$ to indicate the abductive logic program obtained by grounding the DALP with the history $\mathbf{HAP}$. The history dynamically grows during the computation, as new events happen.[1]

An instance $DALP_{\mathbf{HAP}}$ of this framework can be queried with a *goal* G, that may contain both predicates defined in KB and abducibles. The abductive computation produces a set Δ of hypotheses, partitioned in two sets: general hypotheses (ΔA) and *expectations* ($\mathbf{EXP}$), containing *positive* expectations (in the form of $\mathbf{E}(Descr[, Time])$ atoms), and *negative* expectations ($\mathbf{EN}((Descr[, Time]))$ atoms).

Typically, expectations will contain variables, over which CLP constraints can be imposed. Variable quantification is *existential* in $\mathbf{E}$ expectations, and *universal* in $\mathbf{EN}$ expectations (unless the same variable is used outside of such expectation). Explicit negation can also be applied to expectations.[2] Constraints on universally quantified variables will be considered as *quantifier restrictions*. For instance, $\mathbf{EN}(p(X))$, $X > 0$ has the semantics $\forall_{X>0}\ \mathbf{EN}(p(X))$.

The declarative semantics for $DALP_{\mathbf{HAP}}$ is based on its ground version, and considers CLP-like constraints as defined

[1] The source of events is not modelled, but can be imagined as a queue.

[2] For each abducible predicate $A \in \{\mathbf{E}, \mathbf{EN}\}$, the abducible predicate $\neg A$ is implicitly defined, to represent the negation of A, together with the integrity constraint $(\forall X)\neg A(X), A(X) \rightarrow \mathit{false}$.

predicates. First, an abductive explanation should entail the goal and satisfy the integrity constraints:

$$Comp(KB \cup \Delta A \cup \mathbf{EXP} \cup \mathbf{HAP}) \models G \quad (1)$$

$$Comp(KB \cup \Delta A \cup \mathbf{EXP} \cup \mathbf{HAP}) \models IC \quad (2)$$

where, as in the IFF proof procedure, the symbol $\models$ stands for three valued entailment and $Comp$ stands for completion.

Among the sets of expectations of an instance $DALP_{\mathbf{HAP}}$, we select the ones that are consistent with respect to expectations (i.e., the same event should not be both expected to happen and not to happen), and that are *confirmed*:

Definition 1 *A set* $\mathbf{EXP}$ *is* E-consistent *iff for each (ground) term* p*:*

$$\{\mathbf{E}(p), \mathbf{EN}(p)\} \not\subseteq \mathbf{EXP}. \quad (3)$$

Given a history $\mathbf{HAP}$*, a set of expectations* $\mathbf{EXP}$ *is* confirmed *if and only if for each (ground) term* p*:*

$$Comp(\mathbf{HAP} \cup \mathbf{EXP}) \cup \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{EN}(p) \rightarrow \neg\mathbf{H}(p)\} \not\models false \quad (4)$$

We write $DALP_{\mathbf{HAP}} \not\approx_{\Delta A \cup \mathbf{EXP}} G$ if equations (1-4) hold.

The operational semantics is an extension of the IFF. Each state is defined by a tuple defining confirmed, disconfirmed, and pending expectations, along with the resolvent, the set of abduced literals that are not expectations, the constraint store, a set of *partially solved integrity constraints*, and $\mathbf{HAP}$.

A derivation D is a sequence of nodes T_j, where the initial node T_0 contains the goal G as the initial resolvent, and the other nodes $T_j, j > 0$, are obtained by applying one among a set of transitions, until quiescence.

Definition 2 *Starting with an instance* $DALP_{\mathbf{HAP}^i}$ *there exists a* successful derivation *for a goal* G *iff the proof tree with root node* T_0 *has at least one consistent leaf node* T_n *(i.e., there exists for* T_n *a ground variable assignment such that all the constraints are satisfied). In that case, we write:*

$$DALP_{\mathbf{HAP}^i} \not\sim^{\mathbf{HAP}^f}_{\Delta A \cup \mathbf{EXP}} G$$

The transitions are those of the IFF, enlarged with those of CLP, and with specific transitions accommodating the concepts of hypotheses confirmation and evolving history.

The CLP(FD) solver has been extended for dealing with universally quantified variables and *quantifier restrictions*. For instance, given two expectations $\forall_{X<10}\mathbf{EN}(p(X))$ and $\exists_{Y>5}\mathbf{E}(p(Y))$, the solver is able to infer $\exists_{Y\geq 10}\mathbf{E}(p(Y))$. To the best of our knowledge, this is the only proof-procedure able to abduce atoms containing universally quantified variables; moreover, it also handles constraints *à la* CLP on universally quantified variables.

We proved soundness for *allowed* DALPs:[3]

Theorem 1 *Given* $DALP_{\mathbf{HAP}^i}$ *and a ground goal* G*, if* $DALP_{\mathbf{HAP}^i} \not\sim^{\mathbf{HAP}}_{\Delta} G$ *then* $ALP_{\mathbf{HAP}} \not\approx_{\Delta} G$*.*

3 Case study

We conclude with an example from the medical domain to show two main contributions of our work: the dynamic detection of new facts, and the confirmation of hypotheses by events. Suppose a symptom s can be caused by one of three types of diseases. Let disease d_1 be an acceptable explanation for s if the patient is not also affected by d_3, and in such a case the patient's temperature cannot go below 37^oC. s may alternatively be caused by disease d_2, and in this case red spots are expected to appear on the patient's skin within 4 days. Finally, d_3 may be the cause of s, provided that an exam r gives a positive result:

$$symptom(s, T_1) \leftarrow \mathbf{disease}(d_1, T_1) \wedge not\ \mathbf{disease}(d_3, T_1) \wedge \mathbf{EN}(tem(T), T_1) \wedge T < 37.$$
$$symptom(s, T_1) \leftarrow \mathbf{disease}(d_2, T_1) \wedge \mathbf{E}(red\_spots, T_2) \wedge T_1 < T_2 \leq T_1 + 4.$$
$$symptom(s, T_1) \leftarrow \mathbf{disease}(d_3, T_1) \wedge \mathbf{E}(exam(r, +), T_1).$$

The initial goal can be the observation $symptom(s, 1)$. We model $\mathbf{disease}$ as a classical abducible, whereas *expectations* are used to corroborate the explanations.

Notice the twofold use of expectations: both in the second and third clause, the expectation defines a further event that can support the diagnosis. But while $\mathbf{E}(red\_spots, T_2)$ simply defines the expected course of illness (in order for the diagnosis to be corroborated), $\mathbf{E}(exam(r, +), T_1)$ can also be intended as a *suggestion* to the physician for a further exam to be done, or as a *request* of further information.

The combinations of abducible literals can be refined by means of ICs. For example, if the result of some exam r is positive, then we can assume that the patient is not affected by disease d_1: $\mathbf{H}(exam(r, +), T_1) \rightarrow not\ \mathbf{disease}(d_1, T_1)$. The dynamic occurrence of events can drive the generation and selection of abductive explanations. If the query is, e.g., $symptom(s, 1)$, there can be three alternative explanations:

$\{\mathbf{disease}(d_1, 1), \forall_{T>37}\mathbf{EN}(tem(T), 1)\}$,
$\{\mathbf{disease}(d_2, 1), \exists_{1<T_2\leq 5}\mathbf{E}(red\_spots, T_2)\}$, and
$\{\mathbf{disease}(d_3, 1), \mathbf{E}(exam(r, +), 1)\}$.

If event $\mathbf{H}(tem(36), 1)$ happens, the first set contains a disconfirmed expectation: $\forall_{T>37}\ \mathbf{EN}(tem(T), 1)$, so it can be ruled out. If, within the deadline $T_2 \leq 5$, the event red_spots does not happen, the second set is excluded as well, and only the third remains acceptable.

Finally, integrity constraints could suggest possible cures, or warn about consequences of not taking certain drugs:

$$\mathbf{disease}(d_3, T_1) \rightarrow \mathbf{E}(aspirin, T_1) \vee \mathbf{E}(tem(T), T_2) \wedge T > 40 \wedge T_2 < T_1 + 2.$$

4 Conclusions

We presented *S*CIFF, an abductive proof-procedure able, beside proposing explanations, to infer *expectations* about the happening of events. Expectations are abducibles, but more expressive: they can contain universally quantified variables, possibly with CLP constraints. They can represent requests for information, or the expected evolution of a system. *S*CIFF is able to process dynamically incoming events to confirm the expectations, providing corroboration to abduced hypotheses.

References

[Alberti *et al.*, 2005] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Compliance verification of agent interaction: a logic-based software tool. *Applied Artificial Intelligence*, 2005. To appear.

[Fung and Kowalski, 1997] T. H. Fung and R. A. Kowalski. The IFF proof procedure for abductive logic programming. *Journal of Logic Programming*, 33(2):151–165, November 1997.

[Kakas *et al.*, 1992] A.C. Kakas, R.A. Kowalski, and F. Toni. Abductive logic programming. *J. Log. Comput*, 2(6):719–770, 1992.

[3] The proof is based on a correspondence drawn between *S*CIFF and IFF transitions, and exploits the soundness results of the IFF.