$\mathcal{E}$-RES: Reasoning about Actions, Events and Observations

Antonis C. Kakas
Dep. of Computer Science
University of Cyprus
B.O.Box 20537
CY-1678 Nicosia, Cyprus
antonis@ucy.ac.cy

Rob Miller
School of Library, Archive
and Information Studies
University College London
London WC1E 6BT, U.K.
rsm@ucl.ac.uk

Francesca Toni
Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ, U.K.
ft@doc.ic.ac.uk

Abstract

The language $\mathcal{E}$ for reasoning about actions and change can be translated into an argumentation framework. In this paper, we extend this translation of the basic language and show how it can, together with methods from abduction, form the basis for a principled implementation of $\mathcal{E}$. The extension we have considered concerns the addition of new type of sentences in the language as well as allowing theories where the narrative of events given is incomplete.

A system, called $\mathcal{E}$-RES, is developed that can support directly a variety of modes of common sense reasoning such as: default persistence in credulous or skeptical form, assimilation of observations and their diagnosis possibly under incomplete information, as well as combinations of these. Application problems that require these modes of reasoning can then be addressed in a natural way directly from their high-level specification that the language $\mathcal{E}$ can provide. To improve the efficiency of the system we have considered, together with a notion of relevancy of constrains, its interface with a SAT solver within the process of validating the time universal constrains imposed by ramification statements.

1 Introduction

General formalisms of action and change can provide a natural framework for a variety of AI problems such as diagnosis, planning and cognitive robotics. They can offer a high level of expressivity and a basis for the development of a computational framework to solve these problems. In this paper we study how one such formalism, the Language $\mathcal{E}$ [Kakas and Miller, 1997], can be developed into a framework capable of supporting a variety of basic reasoning modes needed to address this type of AI applications.

The computational foundation of this framework and its associated system, called $\mathcal{E}$-RES, is a re-formulation of the Language $\mathcal{E}$ in terms of argumentation [Bondarenko *et al.*, 1997; Dimopoulos and Kakas, 1995] together with a synthesis of methods from abductive reasoning [Kakas *et al.*, 1998]. This argumentation re-formulation formalizes the intuitive notion that information from later events is stronger than information from earlier events while properly accounting for the possibility of incomplete information in the given theory and the constrains imposed by observations and ramification statements in the theory. Argumentation then gives a natural computational model for the language and allows a principled implementation of the $\mathcal{E}$-RES system in a way that separates issues of expressiveness and efficiency. Within this computational model it is also possible to examine how we can use in a modular way "external" solvers, e.g. a SAT solver [Gent *et al.*, 2000] together with a notion of relevancy of part of the theory to the goal at hand, for improving the computational behaviour of the framework.

2 The Language E and its Model Semantics

The vocabulary of the Language $\mathcal{E}$ consists of a set Φ of *fluent constants*, a set of *action constants*, and a partially ordered set $\langle \Pi, \preceq \rangle$ of *time-points*. This vocabulary depends each time on the domain being modeled. A *fluent literal* is either a fluent constant F or its negation $\neg F$. In the current implementation of the $\mathcal{E}$-RES system the only time structure that is supported is that of the natural numbers, so we restrict our attention here to domains of this type using the standard ordering relation $\leq$ in all examples.

Domain descriptions in the Language $\mathcal{E}$ are collections of the following kinds of statements (where A is an action constant, T is a time-point, F is a fluent constant, L is a fluent literal and C is a set of fluent literals):

- *t-propositions* of the form: L `holds-at` T
- *h-propositions* of the form: A `happens-at` T
- *c-propositions* of the form: A `initiates` F `when` C or, A `terminates` F `when` C
- *r-propositions* of the form: L `whenever` C
- *p-propositions* of the form: A `needs` C.

T-propositions are used to record observations that particular fluents hold or do not hold at particular time-points, and h-propositions are used to state that particular actions occur at particular time-points. C-propositions state general "action laws" – the intended meaning of "A `initiates` F `when` C" is "C is a minimally sufficient set of conditions for an occurrence of A to have an initiating effect on F". R-propositions serve a dual role in that they describe both static constrains between fluents and ways in which fluents may be indirectly affected by action occurrences. The intended meaning of "L `whenever` C" is "at every time-point that C holds, L holds, and hence every action occurrence that brings about C also brings about L". P-propositions state necessary conditions for an action to occur.

The semantics of $\mathcal{E}$ is based on a notion of a *model* of a domain D. A map, $H : \Phi \times \Pi \mapsto \{true, false\}$, is an interpretation of D. Given a time point T and a fluent constant F we first define the notion of an *initiation-point* (*termination-point* resp.) *for F in H relative to D* as follows. Consider first the case where D contains no r-propositions. Then T is an **initiation-point** (**termination-point** resp.) **for** F **in** H **relative to** D iff there is an action constant A such that (i) D contains both an h-proposition A `happens-at` T and a c-proposition A `initiates` (`terminates`, resp.) F `when` C, and (ii) H satisfies C at T (i.e for each $F \in C$, $H(F,T) = true$, and for each F' such that $\neg F' \in C$, $H(F',T) = false$).

When the domain D contains r-propositions this definition has to be extended to allow for initiation or termination points that are generated recursively through such these r-propositions.

Definition 1 *(Initiation/termination point) Let H be an interpretation of $\mathcal{E}$, and D be a domain description. Let $\mathcal{W}$ be the set $2^{\Phi\times\Pi} \times 2^{\Phi\times\Pi}$ and let the operator $\mathcal{F} : \mathcal{W} \mapsto \mathcal{W}$ be defined as follows. For each, $\langle \mathcal{In}, \mathcal{Te} \rangle \in \mathcal{W}$ denote $\mathcal{F}(\langle \mathcal{In}, \mathcal{Te} \rangle)$ by $\langle \mathcal{In}', \mathcal{Te}' \rangle$. Then for any $F \in \Phi$ and $T \in \Pi$, (F,T) is in $\mathcal{In}'$ (respectively in $\mathcal{Te}'$) iff one of the following two conditions holds.*

1. *There is an $A \in \Delta$ such that (i) there is both an h-proposition in D of the form "A `happens-at` T" and a c-proposition in D of the form "A `initiates` F `when` C" (respectively "A `terminates` F `when` C") and (ii) H satisfies C at T.*
2. *There is an r-proposition in D of the form "F `whenever` C" (respectively "$\neg F$ `whenever` C") and a partition $\{C_1, C_2\}$ of C such that (i) C_1 is non-empty, for each fluent constant $F' \in C_1$, $(F',T) \in \mathcal{In}$, and for each fluent literal $\neg F' \in C_1$, $(F',T) \in \mathcal{Te}$, and (ii) there is some $T_2 \in \Pi$, $T \prec T_2$, such that for all T_1, $T \preceq T_1 \preceq T_2$, H satisfies C_2 at T_1.*

Let $\langle \mathcal{In}^f, \mathcal{Te}^f \rangle$ be the least fixed point of the (monotonic) operator $\mathcal{F}$ starting from the empty tuple $\langle \emptyset, \emptyset \rangle$. T is an initiation-point (respectively termination-point) for F in H relative to D iff $(F,T) \in \mathcal{In}^f$ (respectively $(F,T) \in \mathcal{Te}^f$).

It is useful to note that any initiation or termination point at some time T relative to D, defined in this way, must refer to at least one known h-proposition at T in the domain D.

Given this notion of an initiation and termination point then an interpretation H is a **model** of D iff, for every fluent constant F and time-points $T_1 \prec T_3$:

1. If there is no initiation- or termination-point T_2 for F in H such that $T_1 \preceq T_2 \prec T_3$, then $H(F,T_1) = H(F,T_3)$. i.e. fluents change their truth values only via occurrences of initiating or terminating actions;
2. If T_1 is an initiation-point for F in H, and there is no termination-point T_2 for F in H such that $T_1 \prec T_2 \prec T_3$, then $H(F,T_3) = true$. i.e. initiating a fluent establishes its truth value as $true$;
3. If T_1 is a termination-point for F in H, and there is no initiation-point T_2 for F in H such that $T_1 \prec T_2 \prec T_3$, then $H(F,T_3) = false$. i.e. terminating a fluent establishes its truth value as $false$.
4. H satisfies the following constrains:
   - For all F `holds-at` T in D, $H(F,T) = true$, and for all "$\neg F$ `holds-at` T'" in D, $H(F,T') = false$.
   - For all A `needs` C in D and A `happens-at` T in D, H *satisfies* C *at* T.
   - For all L `whenever` C in D, and time-points T, if H satisfies C at T then H satisfies $\{L\}$ at T.

A domain D is *consistent* iff it has a model. A domain D **entails** (written $\models$) the t-proposition F `holds-at` T ($\neg F$ `holds-at` T, resp.), iff for every model H of D, $H(F,T) = true$ ($H(F,T) = false$, resp.).

The first three conditions for a model encapsulate a notion of *default persistence* for fluents and they remain fixed as we extend the expressiveness of the language. The fourth condition imposes *constrains* on the model from explicit information about the fluents given in D. As we increase the expressiveness of the language with new types of sentences new constrains are added in this condition. This separation allows a modular extension of the language and, as we will see, facilitates the development of a proof theory for it.

As an example, consider the following simple "bulb domain" D_b, with action constants *SwitchOn*, *SwitchOff* and *Break* and fluents *Light* and *Normal*:

Example 1 *(Bulb Domain: D_b)*

SwitchOn `initiates` *Light* `when` $\{$*Normal*$\}$	(D_b1)
SwitchOff `terminates` *Light*	(D_b2)
Break `terminates` *Normal*	(D_b3)
$\neg$*Light* `whenever` $\{\neg$*Normal*$\}$	(D_b4)
SwitchOn `needs` $\{\neg$*Light*$\}$	(D_b5)
SwitchOn `happens-at` 2	(D_b6)
Normal `holds-at` 0	(D_b7)

In this example, D_b entails *Light* `holds-at` 4 but not when D_b is extended with *Break* `happens-at` 3.

The above semantics assumes that no events occur other than those explicitly given in D. This is not always a valid assumption and it is possible to have domains where some action types are **open**, e.g. in the example above *Break* could be considered as open. Following work on abduction [Kakas *et al.*, 1998], we define a notion of **generalized model** of D as any model of $D \cup Ab$, where Ab is any set of h-propositions over the open action types in D. A corresponding entailment is then defined in terms of these generalized models.

3 An Argumentation Proof Theory for $\mathcal{E}$

The basic subset of the language $\mathcal{E}$, comprising only of h- and c-propositions, has been re-formulated into the argumentation framework of *LPwNF* in [Kakas *et al.*, 1999]. In this section, we give a brief review of the argumentation formulation of $\mathcal{E}$ and show how it can be extended when we extend the syntax of the basic language or when we allow open action types in a domain description. This results in a proof theory for $\mathcal{E}$ which in turn will form the basis of the $\mathcal{E}$-RES system implementing the language.

The argumentation re-formulation of $\mathcal{E}$ translates a domain D, over the basic subset of the language, into an **argumentation program** $P_{\mathcal{E}}(D) = (B(D), \mathcal{A}_{\mathcal{E}}, <_{\mathcal{E}})$ in *LPwNF*. This is defined in a logic programming argumentation framework over the following background monotonic logic $(\mathcal{L}, \vdash)$:

- $\mathcal{L}$ consists of all sentences $\lambda_0 \leftarrow \lambda_1, \ldots, \lambda_n$ $(n \geq 0)$, with λ_i, $0 \leq i \leq n$, positive or negative (via a negation or complement operator, $\neg$,) literals, and all variables implicitly universally quantified from the outside, and
- $\vdash$ is obtained by repeatedly applying the classical modus ponens inference rule $\dfrac{X \leftarrow Y, \quad Y}{X}$ with $X \leftarrow Y$ any ground instance of a sentence in $\mathcal{L}$.

$B(D)$, called the **background theory**, is given by:

If A `happens-at` $T \in D$, then $Happens(A, T) \in B(D)$.

If A `initiates (terminates, resp.)` F `when` $\{L_1, \ldots, L_n\} \in D$, then $B(D)$ contains a rule:
$Initiation(F, t)(Termination(F, t)\ resp.) \leftarrow Happens(A, t),$
$HoldsAt(L_1, t), \ldots, HoldsAt(L_n, t).$
Here and below $HoldsAt(\neg F_i, t)$ stands for $\neg HoldsAt(F_i, t)$.

The rest of $P_{\mathcal{E}}(D)$ is independent of any given domain D. $\mathcal{A}_{\mathcal{E}}$, called the **argumentation theory** consists of:

Generation rules:
$HoldsAt(f, t_2) \leftarrow Initiation(f, t_1), t_1 \prec t_2$ $\quad (PG[f, t_2; t_1])$
$\neg HoldsAt(f, t_2) \leftarrow Termination(f, t_1), t_1 \prec t_2$ $\quad (NG[f, t_2; t_1])$

Persistence rules:
$HoldsAt(f, t_2) \leftarrow HoldsAt(f, t_1), t_1 \prec t_2$ $\quad (PP[f, t_2; t_1])$
$\neg HoldsAt(f, t_2) \leftarrow \neg HoldsAt(f, t_1), t_1 \prec t_2$ $\quad (NP[f, t_2; t_1])$

Assumption rules:
$HoldsAt(f, t)$ $\quad (PA[f, t])$
$\neg HoldsAt(f, t)$ $\quad (NA[f, t])$

Also, $<_{\mathcal{E}}$ is a **priority relation** defined over $\mathcal{A}_{\mathcal{E}}$ by:
$NP[f, t; t_1] <_{\mathcal{E}} PG[f, t; t_2]$ iff $t_1 \preceq t_2$,
$NG[f, t, t_1] <_{\mathcal{E}} PG[f, t; t_2]$ iff $t_1 \prec t_2$,
$PA[f, t] <_{\mathcal{E}} NG[f, t; t']$, $PA[f, t] <_{\mathcal{E}} NP[f, t; t']$,
together with the corresponding cases where positive rules are replaced by negative rules and vice versa.

The essential element of this translation is that it formalizes that the effects of later events take priority over the effects of earlier events. The argumentation semantics of $P_{\mathcal{E}}(D)$ is given via the *admissible* extensions of $B(D)$. These are subsets, S, of argument rules from $\mathcal{A}_{\mathcal{E}}{}' \subset \mathcal{A}_{\mathcal{E}}$ consisting only of generation or assumption rules, which are added to $B(D)$. An extension S is **admissible** iff:

- it is consistent i.e. *non-self-attacking*, and
- *(counter-)attacks* any set of arguments *attacking* it.

A set of argument rules, A, **attacks** another such set, B, if the two sets are in conflict, by deriving monotonicaly (in $\vdash$), together with $B(D)$, complimentary literals λ and $\neg\lambda$, respectively, and A is *not of lower priority* than B. A set A is of lower priority than B if it has a rule of lower priority than some rule in B and does not contain any rule of higher priority than some rule in B. The exact definition of an attack, is given in [Kakas *et al.*, 1999].

Given this translation it can be shown, under some quite general restrictions on D, that the models of D correspond exactly to the maximally (w.r.t. set inclusion) admissible extensions of $P_{\mathcal{E}}(D)$. We can then use this translation of the language $\mathcal{E}$ to develop an argumentation-based proof theory for $\mathcal{E}$ by specializing that of *LPwNF* [Dimopoulos and Kakas, 1995] to the argumentation programs $P_{\mathcal{E}}(D)$. This proof theory is a realization of the abstract, argumentation-based computational framework in [Kakas and Toni, 1999]. It is defined in terms of *derivations of trees*, whose nodes are sets of arguments in $\mathcal{A}_{\mathcal{E}}$ attacking the arguments in their parent nodes.

Let S_0 be a (non-self-attacking) set of arguments in $\mathcal{A}_{\mathcal{E}}{}'$ such that $B(D) \cup S_0 \vdash (\neg)HoldsAt(F, T)$ for some literal $(\neg)$ F `holds-at` T that we want to prove to be entailed by D. Then, two kinds of derivations are defined:

- *Successful derivations* $S_0, ..., S$, building, from a tree consisting only of the root S_0, a tree whose root S is an admissible subset of $\mathcal{A}_{\mathcal{E}}{}'$ such that $S \supseteq S_0$.
- *Finitely failed derivations*, guaranteeing the absence of any admissible set of arguments containing S_0. (some attacks are not counterattacked in the final tree in the derivation).

Then, the given literal, $L = (\neg)HoldsAt(F, T)$, we say that L is a **skeptical** consequence of D iff (i) there exists a successful derivation starting with S_0 and, (ii) for every set S_0' of argument rules in $\mathcal{A}_{\mathcal{E}}{}'$ such that $B(D) \cup S_0'$ derives (in $\vdash$) the complement of L, every derivation for S_0' is finitely failed. If only the first condition, (i), holds we say that L is a **credulous** consequence of D.

The formal details of the derivations are not needed for this paper (see [Kakas and Toni, 1999; Dimopoulos and Kakas, 1995]). Informally, both kinds of derivation incrementally consider all attacks against the root and, whenever this does not counter-attack one of its attacks, a new set of arguments that can do this is added to the root. Then, the process is repeated, until every attack has been counter-attacked successfully (successful derivation) by the extended root or until some attack cannot be counter-attacked by any extension of the root (finitely failed derivations). During this process, the counterattacks are chosen in such a way that they do not attack the root. Examples of this proof theory will be presented in the next section.

An important feature of the argumentation re-formulation of $\mathcal{E}$ is the fact that this is modular with respect to the addition of new type of sentences in the language. This follows primarily from the fact that the translation is faithful at the level of the models of the language and so it can reflect the modular separation of the model definition into two parts: conditions (1-3) encapsulating default persistence and condition 4 for extra constrains. When we add r-propositions we only need to extend the background definitions of $Initiation$ and $Termination$ in $B(D)$ without changing the type of arguments in $P_{\mathcal{E}}(D)$. For each, L `whenever` C a fact $Whenever(L, C)$, is added to $B(D)$, and the definitions of *Initiation* and *Termination* are augmented with:

$$Start(l,t) \leftarrow Whenever(l,c), Select(c, l1, \{l_2, \ldots, l_n\}), Start(l1,t), HoldsAt(l_2, t_+), \ldots, HoldsAt(l_n, t_+)$$

where for a positive literal $l = F$, $Start(l,t)$, is to be read as *Initiation*(F,t) and for a negative literal, $l = \neg F$, as *Termination*(F,t), and t_+ is the next immediate time after t. Hence every event that brings about any literal, l_1, of C while the rest of this, $\{l_2, \ldots, l_n\}$, continues to hold also brings about, through the r-proposition, L. This means that the type of arguments remain exactly the same but now we have additional possibilities for generation arguments through the ramification statements.

In turn, the only extension required to the proof theory is to add, for any r-proposition L `whenever` $\{L_1, \ldots, L_k\}$, to the root S of any derivation a set of arguments, S_r, so that that $B(D) \cup S \cup S_r \vdash \phi$ where ϕ is the (classical) formula $HoldsAt(L,t) \leftarrow HoldsAt(L_1,t), \ldots, HoldsAt(L_k,t)$. Similarly, when we extend the language with t and p-propositions we need to extend the proof theory by adding to the root of a derivation a set of arguments, S_t and S_p, so that they can derive (with $B(D)$) the *HoldsAt* literals corresponding to these sentences. The proof theory continues then as before but now with the extra attacks against S_r, S_t and S_p to be considered.

Finally, when we have *open* action types in the domain D the proof theory is extended to allow the *abduction* of a new set of events, H, and hence new generation arguments can be added to the root. Derivations are now defined in terms of tuples, $< H, S >$ where H is a set of h-propositions (not in D) and S is, as before, a set of arguments at the root of the derivation tree. In this extended proof theory it is possible for new attacks to be generated during the derivation due to the new events abduced. The proof theory therefore now includes **suspended** attacks which can become actual attacks when H grows. If this happens then these attacks need to be counter-attacked as usual, otherwise, suspended attacks that remain so until the end of the derivation are ignored. Given these extensions we can prove the following theorem.

Theorem 1 *(Soundness of Extended Proof Theory)*
Let D[1] be a description domain possibly with open action types and S_0 a consistent set of arguments. If there exists a successful extended derivation from $< \emptyset, S_0 >$ to $< H, S >$ then there exists a generalized model, M, of D such that (i) M is also a model of $D \cup H$ and (ii) M satisfies L for any literal L s.t. $B(D \cup H) \cup S \vdash L$.
Also, if every extended derivation from $< \emptyset, S_0 >$ is finitely failed, then there exists no generalized model, M, of D such that M satisfies L for every literal L s.t. $B(D \cup H) \cup S_0 \vdash L$.

[1]All results in this paper refer to domains with discrete linear time, a finite number of h-propositions and a restriction that limits the possibility for events to simultaneously initiate and terminate the same fluent.

4 Reasoning with the Language $\mathcal{E}$

The language $\mathcal{E}$ can support in a natural way a variety of modes of reasoning with actions and observations. The argumentation-based computational model for E, described in the previous section, allows a principled implementation of these forms of reasoning. In this section we present some of these forms of reasoning and explain briefly how they are mapped into argumentation.

4.1 Default Persistence

The argumentation translation of the language $\mathcal{E}$ maps the basic reasoning of default persistence captured by the model theoretic semantics of $\mathcal{E}$ into an argumentation reasoning. Consider the following example where vaccine A provides protection only for people with blood type O, and vaccine B for people with blood type other than O.

Example 2 *(Vaccinations - No open actions)*

InjectA `initiates` *Protected* `when` $\{$*TypeO*$\}$	(D_v1)
InjectB `initiates` *Protected* `when` $\{\neg$*TypeO*$\}$	(D_v2)
InjectA `happens-at` 2	(D_v3)
InjectB `happens-at` 3	(D_v4)

Given this domain, we can show that at any time T_f after 3 $G = \{$*Protected* `holds-at` $T_f\}$ is a skeptical consequence of D_v whereas for times less or equal to 3 $Protected$ is only a credulous consequence. An argument S_0 for G is given by $\Delta_0 = \{PG[Protected, T_f; 2], PA[Type0, 2]\}$ i.e. a generation argument based on the event of *InjectA* at time 2 together with an assumption argument for $Type0$ at time 2. All attacking arguments against this can be counter-attacked (defended) by S_0 itself. This gives a successful derivation for S_0 and thus G is a credulous consequence. To show that it is a skeptical consequence we consider the opposite goal $\neg G$.

The only way to derive this is through the argument $R_1 = \{NA[Protected, T_f]\}$ (there are no generation arguments for $\neg G$). This is attacked by S_0 given above, which can be counterattacked (only) if R_1 is extended to R_2 with the assumption argument $NA[Type0, 2]$. But R_1 and thus also R_2 are also attacked by $\{PG[Protected, T_f; 3], NA[Type0, 3]\}$ via the event *InjectB* at time 3. To defend against this it is now necessary to add $PA[Type0, 3]$ to R_2 to give $R_3 = \{NA[Protected, T_f], NA[Type0, 2], PA[Type0, 3]\}$. But then we have a new attack against R_3 given by $\{NP[Type0, 3; 2],\ NA[Type0, 2]\}$ through a persistence argument from time 2 to time 3. This attack can only be counterattacked via a generation argument for $HoldsAt(Type0, 3)$. But no such arguments exist in D_v and hence the derivation for $\neg G$ finitely fails, as required.

This example shows how the argumentation reasoning deals correctly with default persistence under incomplete information. For a more complex example consider the same goal G in the domain below where the fluents *TypeO* and *Strong* are incompletely specified.

Example 3 *(Vaccinations Cnt.)*

InjectB `initiates` *Protected* `when` $\{\neg$*TypeO*$\}$	(D_r1)
InjectC `initiates` *Protected* `when` $\{$*Strong*$\}$	(D_r2)
Strong `whenever` $\{$*TypeO*$\}$	(D_r3)
InjectB `happens-at` 2	(D_r4)
InjectC `happens-at` 3	(D_r5)

As above, derivations for $\neg G$ finitely fail. The two attacks against $\neg G$, via the two injection events, can only be counterattacked by $\{PA[Type0, 2]\}$ and $\{NA[Strong, 3]\}$. But then the satisfaction of the ramification statement requires $\{PA[Strong, 2]\}$ to be added. In turn this gives a new persistence attack of *Strong* from time 2 to 3 which can not be counterattacked as there is no generation rule for $\neg$*Strong*.

4.2 Assimilating Observations and Diagnosis

A domain description in the language $\mathcal{E}$ may contain observations (t-propositions) about some of its fluents. The observations can refer either to some initial time or any other time point. An argument for a default conclusion to be valid must also be extensible to an admissible superset that is able to confirm these observations. This extra requirement gives a form of reasoning from effects to causes both forward and backward in time.

Example 4 *Infections - No open actions*

Expose `initiates` *Infected* `when` $\{$*TypeA*$\}$	(D_i1)
Expose `initiates` *Infected* `when` $\{$*TypeB*$\}$	(D_i2)
Allergic `whenever` $\{$*TypeA*, *Infected*$\}$	(D_i3)
Allergic `whenever` $\{$*TypeB*$\}$	(D_i4)
Expose `happens-at` 3	(D_i5)
$\neg$*Infected* `holds-at` 1	(D_i6)
Infected `holds-at` 6	(D_i7)

The observation at time 6 requires that a generation rule argument for *Infected* at 6 is added to the root of any derivation. The weaker assumption argument $PA[Infected; 6]$ cannot defend against its persistence attack starting from time 1 where the observation of $\neg$*Infected* is given. The only possibility for such a generation rule is the one based on the event of $Expose$ at time 3 with either *TypeA* or *TypeB* assumed at time 3, and consequently at any other time before or after 3. Then the satisfaction of the two r-propositions imply that *Allergic* holds skeptically from 6 onwards.

Effectively, these observations are explained in terms of missing information on incomplete fluents. When a domain contains open action types this gives us a form of *diagnosis* of the observations in terms of assumptions both on incomplete fluents and on unknown (in D) events.

Definition 2 *(Diagnosis in $\mathcal{E}$)*
Let D be a given domain description[2] *and O a set of observations. A* **(strong) diagnosis** *for O* **in** *D is a set H of h-propositions s.t. $D \cup H$ is consistent and $D \cup H \models O$.*

[2]For simplicity of presentation we will assume that the domain does not contain any p-propositions, P. If this is the case then we have an extra requirement on the diagnosis that $D \cup H \models P$.

A weaker form of diagnosis useful when we have incomplete information on fluents whose truth cannot be affected by any action (e.g at some initial time point), is as follows.

Definition 3 *(Conditional Diagnosis in $\mathcal{E}$)*
Let D be a given domain description and O a set of observations. Then a **weak diagnosis** *for O* **in** *D is a set H of h-propositions such that there exists a model M of $D \cup H$ where $M \models O$. We say that H is* conditional on a set of assumptions *Δ iff Δ is a set of t-propositions ($\Delta \cap O = \emptyset$) such that H is a (strong) diagnosis for O in $D \cup \Delta$. The tuple, $< H, \Delta >$, is called a* **conditional diagnosis** *for O in D.*

Note that the assumptions Δ in a conditional plan can refer to any time point not necessarily to an initial time point only. In the previous example the empty set $H = \emptyset$ is a weak diagnosis for the observation *Infected* `holds-at` 6 in the domain D_i given by the sentences D_i1-D_i5. Two conditional diagnoses are $< \emptyset$, *TypeA* `holds-at` 3 $>$ and $< \emptyset$, *TypeB* `holds-at` 3 $>$. Note that $< \emptyset$, *TypeA* `holds-at` 1 $>$ is also a conditional diagnosis. The assumption that *TypeA* holds at some time point (e.g. an initial time point 1) implies that it also holds at any other time point as no action in D_i can affect the value of this fluent. Typically, if we have incomplete information on fluents that cannot be affected by any action or the information is incomplete at some initial time point before which actions can not occur then a conditional diagnosis is appropriate.

The argumentation based proof theory extended with abduction and suspended attacks to allow for the incremental construction of a set of h-propositions, as described in section 2, computes correctly conditional diagnoses.

Theorem 2 *Let D be a given domain description and O a set of observations. Let also $S_0 \subseteq \mathcal{A}_{\mathcal{E}}$ be a set of arguments and H_0 a set of action facts such that $B(D \cup H_0) \cup S_0 \vdash O$. If there exists a successful extended derivation in D from $< H_0, S_0 >$ to $< H, S >$ then $< H, \Delta >$ is a conditional diagnosis for O in D, where $\Delta = \{F$ `holds-at` $T | PA[F, T] \in S\} \cup \{\neg F$ `holds-at` $T | NA[F, T] \in S\}$.*

To illustrate this computation of conditional diagnoses let us consider again example 1 where D_b7 is absent and *Break* is an open action type. Suppose we are given the observations: *Light* `holds-at` 4 and $\neg$*Light* `holds-at` 6.

To assimilate the first observation we can use a generation argument, $S_0 = \{PG[Light, 4; 2], PA[Normal, 2]\}$, based on the given event of *SwitchOn*. This can defend itself against all its attacks except possibly an attack via a generation argument based on an event of *Break* at a time after 2 and before 4. As we have no such event in our computed diagnosis this remains suspended. Also, because the p-proposition in D requires that $\neg$*Light* holds at 2, S_0 will be extended to S_1 with $NA[Light, 2]$. Note that this is also the reason that the other possible proof for the first observation given by $S'_0 = \{PA[Light, 4]\}$ (i.e. the light was true from the start) is excluded. We then have the only one conditional diagnosis for the first observation given by $< \emptyset, \{PA[Normal, 2], NA[Light, 2]\} >$.

To assimilate the second observation the only way we can extend, S_1, is via a generation argument based on an event of *Break* at a time before 6. The assumption argument $NA[Light, 6]$ is rejected as it is attacked by S_1 and hence cannot be defended by S_1 and no generation arguments based on an event of *SwitchOff* are possible as no such events are given in the domain description. Hence S_1 is extended to $S_2 = S_1 \cup \{NG[Light, 4; T]\}$ and H_0 to $H_1 = \{$*Break* `happens-at` $T\}$ for $T < 6$. Note that this generation of $\neg$*Light* is *indirect* through the ramification statement in the theory.

Adding this new event results in the re-examination of the suspended attack from before. In general, there are two ways to deal with this situation. One way is to constrain the time of the new event so that it does not lead to an actual attack. The other is to counter-attack this attack in the usual way. In this case, the second option is not available as we cannot assume *SwitchOn* events. Hence we are forced to set $T \geq 4$ so that the *Break* event cannot form an attack on the arguments we have in S_1 for the first observation. The computation then concludes successfully with $< H_1, S_2 >$ giving the (a set of) conditional diagnosis (one for each T in [4,6)) $< \{$*Break* `happens-at` $T\}, \{$*Normal* `holds-at` 2, $\neg$*Light* `holds-at` $2\} >$.

A computed conditional diagnosis $< H, \Delta >$ in D can easily be tested to see if this is a strong diagnosis by checking whether the assumptions Δ are a skeptical consequence of $D \cup H$. In the previous example, $\{$*Normal* `holds-at` $2\}$ is not a skeptical consequence (the domain is incomplete on this fluent) and hence the diagnosis needs this condition. On the other hand, we can see that $\{$*Normal* `holds-at` $2\}$ holds in every possible conditional diagnosis of the observations.

In general, we can combine together this diagnostic reasoning with the default reasoning of $\mathcal{E}$ by asking whether a goal holds credulously or skeptically under one or all possible diagnoses of the observations respectively. Again in the example above the goal $G_1 =$ *Normal* `holds-at` T holds skeptically for any time T less or equal 4 and $G_2 = \neg$*Normal* `holds-at` T holds skeptically for any time T from 6 onwards. At time 5 these are only credulously true. If we allow the action *SwitchOff* also to be open then G_2 is no longer a skeptical consequence as in at least one possible diagnosis through a new *SwitchOff* event this does not hold.

5 The $\mathcal{E}$-RES System: Implementing $\mathcal{E}$

The argumentation based proof theory described in section 3 forms the basis of a principled implementation of the language $\mathcal{E}$ into a system, called $\mathcal{E}$-RES. The computational effectiveness of this system depends on two main factors:

- Reducing the number of attacks considered for the goal at hand by restricting only to attacks that are necessary.
- Improving the efficiency of the satisfaction of the global constrains imposed by the t,p and r-propositions.

A major optimization that we can apply with respect to the first factor concerns the consideration of persistence attacks. It is easy to show that it is not necessary to consider chains of persistence attacks between two time points provided that we consider the direct persistence attacks between any two time points. The next lemma shows that it is sufficient to consider only a *restricted* set of attacks.

Definition 4 *A* **restricted attack** *against a set S is a minimal attack on S which does not contain any persistence rule $PP[F, T'; T]$ (resp. $NP[F, T'; T]$) unless S contains the assumption rule $NA[F, T]$ (resp. $PA[F, T]$) and $B(D) \cup S \vdash HoldsAt(F, T')$ (resp. $B(D) \cup S \vdash \neg HoldsAt(F, T')$).*

Lemma 1 *Let D be a domain and S a set of argument rules that is consistent and attacks all the restricted attacks against it. Then there exists a superset of S which is admissible.*

This means that we only need to consider those persistence attacks against a set S that start from assumptions that we are making in S to ensure that these assumptions are self-consistent. In the implementation of $\mathcal{E}$-RES we exploit this lemma by considering a notion of *suspended* persistence attacks on an assumption which are activated whenever the contrary assumption (at another time point) is added to S.

5.1 Satisfiability of Constrains in $\mathcal{E}$-RES

The global constrains imposed by the t,p and r-propositions can be computationally demanding. Although most of these constrains refer to a single time point, those imposed by the ramification statements need to be satisfied at every time point and hence could be a major source of inefficiency. The following lemma allows us to reduce this computation by confining the task of satisfiability to a specific set of time points.

Lemma 2 *Let D be a domain and T_1, T_2 ($T_1 < T_2$) be time points such that there is no h-proposition in D at any time T in (T_1, T_2). Suppose also that there exists a partial model M_p of D defined over the whole time line minus the interval $(T_1, T_2]$, except at times points in $(T_1, T_2]$ where t-propositions are given in D where M_p satisfies the conditions imposed by these. M_p also satisfies any p-propositions at T_2. Then if there exists a time point T in $(T_1, T_2]$ such that M_p can be extended to a partial model covering also T then M_p can be extended to a full model of D.*

Hence when D contains only a finite number of h-propositions we can split the (linear) time line to a finite number of time intervals and satisfy the ramification constrains only at one time point in each of these intervals. In addition, we must also ensure that the satisfaction of the ramification constrains at one time point does not invalidate their satisfaction at some other time point in another interval. $\mathcal{E}$-RES implements an interleaved process of (a) satisfiability of the ramification statements as classical implications at these time points and (b) cross-check of the assumptions required in (a) under the language $\mathcal{E}$ default persistence.

As the number of (ground) ramification constrains at each time point can be large we can employ a SAT solver [Gent *et al.*, 2000] within the $\mathcal{E}$-RES system to carry out this process (a) of generating a classical model for these. Furthermore, we have considered a notion of *relevancy* of ramifications to the query at hand which, assuming that D is consistent, selects at each time point only a subset of ramification constrains. Therefore within the global constraint satisfaction we now have an iterative (over the finite number of time points) process of interleaving between: (i) projecting the assumptions, that we have added to S so far, to the current time point and selecting the relevant ramification constrains based on these,

(ii) generating a classical model of these constrains using a SAT solver given the partial instantiation generated in (i), and (iii) ensuring the compatibility of this model with the arguments S computed so far at the previous time points. Note that the output of steps (ii) and (iii) could affect the set of relevant ramifications computed in (i) and hence we need to repeat the whole process before going to the next time point.

Initial experiments with this iterative method indicate a significant reduction in the computation. Note however that we are still left with the problem of deciding which t-propositions are relevant to the query/goal at hand. Currently, we assume that these are selected externally to the system.

5.2 Functionality of the $\mathcal{E}$-RES system

The $\mathcal{E}$-RES system is currently implemented in Prolog (Eclipse 4.2). An interface allows the user to define directly in the syntax of the language $\mathcal{E}$ (see figure 1) the domain description. The system also supports some extra forms of auxiliary information, e.g. that a fluent is *constant* and so does not change over time. Open action types are specified (see figure 2) together with their associated p-propositions and priority information amongst them that might exist. In addition, although $\mathcal{E}$ is defined as a propositional language the $\mathcal{E}$-RES system allows domain descriptions to be given in non-propositional form under some restrictions. An early version of the system is available from http://www.ucl.ac.uk/ uczcrsm/LanguageE/. New versions of the system will be added to this web site in the near future.

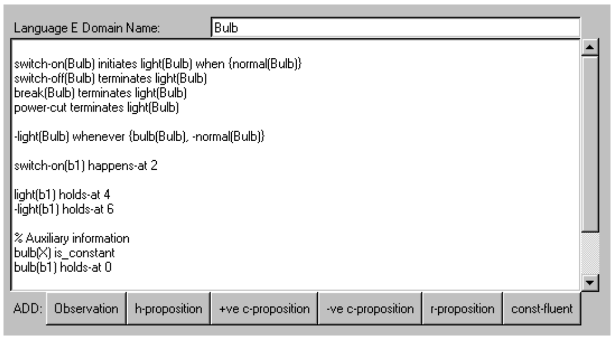

Figure 1: The Domain Entry Window of $\mathcal{E}$-RES

The $\mathcal{E}$-RES system supports directly a number of queries which can be used to perform the forms of reasoning described above in section 4. Given a domain description D and a goal `Goal` as a conjunction of t-propositions, the type of queries available by the system are the following.

- `skeptical(Goal)`: if this succeeds (resp. finitely fails) then `Goal` is (resp. is not) a skeptical consequence of D.
- `credulous(Goal)`: if this succeeds (resp. finitely fails) then `Goal` is (resp. is not) a credulous consequence of D.
- `credulous(Goal,X)`: if this succeeds then `Goal` is a credulous consequence of D, and the set of arguments returned in `X` provides a corresponding admissible extension of D.

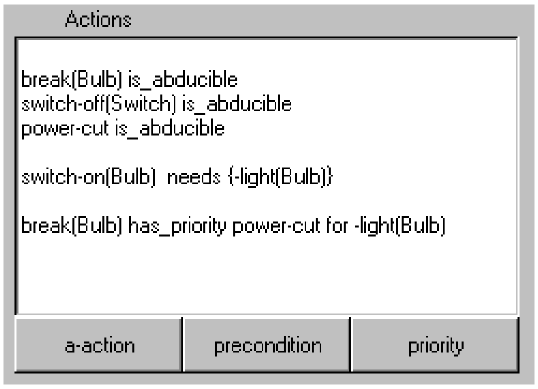

Figure 2: The Action Entry Window of $\mathcal{E}$-RES

- `assimilate(Diagnosis)`: if this succeeds then the sets of events and assumptions in `Diagnosis` form a conditional diagnosis for all the observations in D.

Hence the skeptical and credulous default reasoning of $\mathcal{E}$ is done simply with the use of the corresponding queries in $\mathcal{E}$-RES. Note that if in the definition of D we specify that some actions are open then these queries refer to the truth or not of the $Goal$ in the generalized models of D, thus implementing default reasoning under the possible diagnoses of the observations given in D. If we are interested only in finding out how the observations can be explained then we use the $assimilate$ query which gives the conditional diagnoses for these in D.

The $\mathcal{E}$-RES system can also be used to do *planning*. The effects of the actions together with the preconditions for these effects to hold are described in D through c-propositions. Ramification statements can be used to specify additional domain knowledge. The initial state is described with a set of observations at the initial time 0. Actions are specified as open together with any p-propositions for their preconditions. We can also give other information e.g. incompatibility statements between actions. These are treated as integrity constrains in abduction. Then the query *planning(Goal, Plan)* where $Goal$ is the final state, returns in $Plan$ a *conditional*[3] *plan* under the possibly incomplete initial state.

6 Related Work and Conclusions

Recently there has been a wide interest in developing specialized action languages [Gelfond and Lifschitz, 1998]. These efforts have concentrated on the formal semantics of such languages and how they can be applied to specific problems. Examples of these are the language $Golog$ [Levesque *et al.*, 1997], based on the early framework of the situation calculus, for cognitive robotics, a circumscriptive Event Calculus [Shanahan, 2000] and the language $\mathcal{C}$ [Giuchiglia and Lifschitz, 1998] for planning, and the language $\mathcal{L}$ together with others related to it [Baral *et al.*, 2000; Gelfond and Galloway, 2001] for the problem of diagnosis. Our work focuses on the

[3]A conditional plan is defined in a completely analogous way to that of a conditional diagnosis in definition 3.

general computational aspects of such languages using argumentation and abduction as a basis for a principled implementation of the language $\mathcal{E}$. A system, called Causal Calculator [McCain, 1997], which is based on the language $\mathcal{C}$ translates the whole representation into a propositional theory and then uses a SAT solver to find a solution to its query. A systematic comparison of the $\mathcal{E}$-RES system with these systems would be useful.

An interesting feature of our approach is the possibility it opens of synthesizing, in the implementation of these languages, the resolution based computation of argumentation and abduction with the propositional satisfiability methods of SAT solvers. A SAT-based procedure has also been used recently in $\mathcal{C}$ [Giuchiglia, 2000] for planning. This hybrid computational model, that could also include other methods e.g. constraint solving, is an important topic of future work. We are also investigating further notions of relevancy of parts of the theory, especially of t-propositions, to the query at hand in order to focus the computation and improve its efficiency.

References

[Baral *et al.*, 2000] C. Baral, S. McIlraith, and T. Son. Formulating diagnostic problem solving using an action language with narratives and sensing. In *KR-00*, 2000.

[Bondarenko *et al.*, 1997] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic framework for default reasoning. *Journal of Artificial Inelligence*, 93(1-2):63–101, 1997.

[Dimopoulos and Kakas, 1995] D. Dimopoulos and A.C. Kakas. Logic programming without negation as failure. In *Proc. of ILPS'95*, volume pp. 369-383, 1995.

[Gelfond and Galloway, 2001] M. Gelfond and J. Galloway. Diagnosing dynamic systems in a-prolog. In *Answer Set Programming, AAAI Symposium (to appear)*, 2001.

[Gelfond and Lifschitz, 1998] M. Gelfond and V. Lifschitz. Action languages. In *ETAI*, volume 3(16), 1998.

[Gent *et al.*, 2000] I. Gent, H. van Maaren, and T. Walsh. *Highlights of Satisfiability Research in the Year 2000*. IOS Press, 2000.

[Giuchiglia and Lifschitz, 1998] E. Giuchiglia and V. Lifschitz. An action language based on casual explanation. In *AAAI-98*, pages 623–630, 1998.

[Giuchiglia, 2000] E. Giuchiglia. Planning as satisfiability with expressive action languages. In *KR-00*, 2000.

[Kakas and Miller, 1997] A.C. Kakas and R.S. Miller. A simple declarative language for describing narratives with actions. In *JLP 31(1-3), pp. 157-200*, 1997.

[Kakas and Toni, 1999] A.C. Kakas and F. Toni. Computing argumentation in logic programming. In *JLC 9(4), 515–562, O.U.P*, 1999.

[Kakas *et al.*, 1998] A.C. Kakas, R.A. Kowalski, and F. Toni. The role of abduction in logic programming. In *Handbook of Logic in AI and Logic Programming*, volume 5, pages 235–324. OU Press, 1998.

[Kakas *et al.*, 1999] A.C. Kakas, R.S. Miller, and F. Toni. An argumentation framework for reasoning about actions and change. In *LPNMR'99, 78–91, Springer Verlag*, 1999.

[Levesque *et al.*, 1997] H. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. Golog: A logic programming language for dynamic domains. *Logic Programming*, 31:59–84, 1997.

[McCain, 1997] N. McCain. *Causality in Commonsense Reasoning about Actions*. PhD thesis, University of Texas at Austin, 1997.

[Shanahan, 2000] M. Shanahan. An abductive event calculus planner. *Logic Programming*, 44(1-3):207–239, 2000.