Using Constrained Resolution for Abductive Temporal Reasoning

Nicolas Chleq
INRIA Sophia-Antipolis
BP 93 – 06902 Sophia Antipolis Cedex – France
chleq@sophia.inria.fr

Abstract

We describe in this article an abductive procedure based on a constrained resolution principle. The choice of constrained resolution is motivated by the whish to gain full advantage of using reified temporal logics. For this purpose, it is interesting to deal efficiently with temporal ordering and equality relation between instants. The constrained resolution principle described here is a solution to this point. It is an instance of the more general constrained resolution principle of H.J. Bürckert. It also relies on the work done in the area of temporal constraint propagation.

For the purpose of temporal reasoning it is also necessary to cope with temporal persistency of known and deduced facts. This point is solved by handling persistency in an abductive fashion.

1 Introduction

This article describes a resolution-based abductive procedure. This procedure is based on works done in the area of abductive logic programming and uses a constrained resolution principle. Such a resolution principle is necessary in order to be able to deal with any reified temporal logic based on instants. For the purpose of this paper, we use it on a simple temporal logic based on a simple and naïve ontology. It is also sufficiently expressive for practical use, but this gain on expressiveness is due to an increased complexity of the language itself (more axioms), hence a more complex reasoning task.

The use of abduction in temporal reasoning has been motivated by [12]. This reasoning method is complementary to prediction as it allows to deal with persistency and produce explanation, while retaining a very intuitive form for causal rules *effect if causes*. Abduction has also been applied to planning with temporal formalisms such as the Event Calculus [5, 10]. Most abductive procedure are based on resolution, and practical use of abduction relies on the feasibility of resolution based reasoning for temporal reasoning.

Though it is not necessary to argue again on the usefulness of a resolution principle, practical use of this method is not straightforward. In particular, some features of the formulæ on which this method is applied can suffer from great efficiency problems. Equality relation and self-resolving clauses are some examples of these problematic features. Most of these problems are to be solved by adapted strategies and specialized inference rules. A lot of work has been done on the combination of the resolution principle with some particular "algorithmic" theories: for example the *Theory Resolution* of Stickel [14], and the *Constrained Resolution* of Bürckert [2].

In the next section, we begin by an informal presentation of a temporal logic. This logic is used throughout this paper, and its features are representative of most reified temporal logics. It also illustrates the need of a specialized resolution principle, which I describe as an instance of Bürckert's one. The rest of the paper is devoted to abductive reasoning, and focuses on the abductive procedure we developed as an extension of the abductive logic programming procedure described in [7].

2 Reified Temporal Logics

Reified temporal logics are sorted predicate calculi: one of the sorts is used for time points or time intervals. A formal description of these logics is given in [13]. They are usually based on two primitive entities: instants as in McDermott's logic [9], or intervals as in Allen's one [1]. The basic construct $\langle \phi, t \rangle$ of these logics associates a formula ϕ with a temporal entity t, this last one being either an instant or an interval depending on the kind of logic. The intuitive meaning of this expression is that the formula ϕ is true at the instant denoted by the term t, or true throughout the interval denoted by t. These languages can express the truth of such-and-such proposition over time, they are hence good candidates for the expression of temporal knowledge.

2.1 A temporal logic

The logic we use is a two-sorted predicate calculus, where *time* is the sort of all expressions denoting time instants, and *proposition* is the sort of all terms associated with temporal entities. It is based on in-

$$\{]t_1,t_2],p\} \rightarrow t_1 < t_2 \qquad (\mathcal{A}_1)$$
$$\{]t_1,t_2],p\} \rightarrow begin(t_1,p) \qquad (\mathcal{A}_2)$$
$$\{]t_1,t_2],p\} \rightarrow end(t_2,p) \qquad (\mathcal{A}_3)$$
$$\{]t_1,t_2],p\} \rightarrow persist(t_1,t_2,p) \qquad (\mathcal{A}_4)$$
$$persist(t_1,t_4,p) \wedge (t_1 \leq t_2 < t_3 \leq t_4) \rightarrow persist(t_2,t_3,p) \qquad (\mathcal{A}_5)$$
$$persist(t_1,t_3,p) \wedge (t_1 < t_2 \leq t_3) \rightarrow true(t_2,p) \qquad (\mathcal{A}_6)$$
$$persist(t_1,t_2,p) \wedge begin(t_3,p) \rightarrow (t_3 \leq t_1) \vee (t_2 < t_3) \qquad (\mathcal{A}_7)$$
$$persist(t_1,t_2,p) \wedge end(t_3,p) \rightarrow (t_3 < t_1) \vee (t_2 \leq t_3) \qquad (\mathcal{A}_8)$$
$$begin(t_1,p) \rightarrow (\forall t_2 > t_1 \;\; persist(t_1,t_2,p) \vee (\exists t_3 \;\; (t_1 < t_3 < t_2) \wedge end(t_3,p))) \qquad (\mathcal{A}_p)$$
$$begin(t_1,p) \wedge persist(t_1,t_2,p) \wedge end(t_2,p) \rightarrow \{]t_1,t_2],p\} \qquad (\mathcal{A}_9)$$

Figure 1: Some axioms of the temporal logic.

stants as the primitive temporal entity, and intervals are written with two instants as their lower and upper bounds.

The simplest expression associates a proposition with an instant or an interval: $\{t,P\}$ means that P holds at time t, and $\{]t,t'],P\}$ means that P holds throughout the interval $]t,t']$. Instants are taken from a set that we want to be dense, so that we are able to speak of an instant between any two other instants: the set of rational numbers suits our need.

We divide the set of propositions into the *ephemeral* ones, used to describe "instantaneous" phenomena, and *durable* ones. Propositions of the first class are always associated with instants by expressions of the form $\{t,p\}$, while propositions of the second type are associated with intervals by expressions of the form $\{]t,t'],p\}$.

The set of durable propositions can be refined similarly to the classification established by Shoham [13]. For our purpose, we are only interested in the *liquid* propositions in this taxonomy, or *homogeneous* in ETL [11]: we call these propositions *stable* and this means that their truth throughout an interval implies their truth at all instants within the interval. The truth of p at one instant t, as a consequence of the truth of p over an interval comprising t, is expressed by $true(t,p)$. The set of stable propositions is a subset of the set of durable ones.

An instant can be represented by five means: a number, a variable, a symbolic constant, an arithmetic expression such as $t+n$ where t is an instant and n a number, and a functional term $f(t_1,\ldots,t_n)$.

We use two relation symbols for the ordering of instants: $<$ and $\leq$. Between these ordering relations and expressions of the form $\{I,P\}$, we propose axiom $\mathcal{A}_1$ in figure 1.

Another useful feature for flexibility of a temporal logic as a knowledge expression language, is the ability to give partial information about truth periods. For this purpose, we introduce the following expressions:

begin(*t*, *p*) means that one of the truth period of proposition p begins at the instant t. When the beginning of the interval is known by this way, the term $e(t,p)$ refers to the end (the upper bound) of this interval. The axiom $\mathcal{A}_2$ establishes the relationship with the expression $\{I,p\}$ where I is an interval.

end(*t*, *p*) means that one of the truth period of p ends at the instant t. In the same way as above, the term $b(t,p)$ refers to the lower bound of this interval. This expression is formally defined by axiom $\mathcal{A}_3$.

persist(t_1, t_2, *p*) means that the interval $]t_1,t_2]$ is included in one of the truth period of p. This expression is defined by the two axioms $\mathcal{A}_4$ and $\mathcal{A}_5$.

The maximality of truth period expressed by formulae like $\{]t,t'],p\}$ entails that the lower bound of these intervals are really the time when the proposition becomes true, and that the upper bounds are the instants when the proposition ceases to be true. This entails that overlapping truth periods of the same proposition lead to a contradiction between the truth inside one of the intervals, and the non-truth outside the other one. We suggest axioms $\mathcal{A}_7$ and $\mathcal{A}_8$ to express that overlapping of distinct truth periods is not allowed for a durable proposition.

Axiom $\mathcal{A}_9$ completes the definition of the logic and enables to deduce a complete truth period from partial information.

2.2 Example

We propose to illustrate the use of this logic by the well known "Yale Shooting Problem" which is written in figure 2.

We can deduce from this example that the gun will become unloaded at time 4 because of the firing. We express it by $end(4,loaded)$ and the proof of it involves reasoning about the temporal persistency of *loaded*. The truth period of this proposition begins at time T_1 because of the *loading* action on the gun. Then, thanks to axiom $\mathcal{A}_p$, it will last as long as needed, provided it is not interrupted. At this moment, the upper bound of the persistence can not be fixed, but we can assume it to be at least equal to

$\{T_1, loading\}$; $true(2, alive)$; $\{4, pull{-}trigger\}$; $T_1 < 2$;
$R_1 : \forall t\ \{t, loading\} \rightarrow begin(t, loaded)$;
$R_2 : \forall t\ \{t, unloading\} \wedge true(t, loaded) \rightarrow end(t, loaded)$;
$R_3 : \forall t\ \{t, pull{-}trigger\} \wedge true(t, loaded) \rightarrow end(t, loaded)$;
$R_4 : \forall t\ \{t, pull{-}trigger\} \wedge true(t, loaded) \wedge true(t, alive) \rightarrow end(t, alive)$

Figure 2: The Yale Shooting Problem. *loading* and *pull-trigger* are ephemeral propositions, and *loaded* and *alive* are stable.

4. It can not be greater than 4 because of axiom $\mathcal{A}_7$. Thus, the only solution is that this upper bound is equal to 4.

3 The constrained resolution principle

In this section, we focus on the ability to do resolution-based reasoning with some temporal logic similar to the one described in the previous section. The main problem of these languages comes from the use of equality and ordering relation symbols. The usual axiomatization of the ordering relation $\leq$, which describes the transitivity, reflexivity and anti-symmetry involves some *self-resolving* clauses. This feature entails that for some queries the resolution process may spend a lot of time with repeated use of these clauses, without any way to know whether or not these inferences are relevant for the original query.

Our solution is an instance of the more general resolution principle proposed by H.J. Bürckert [2]. This constrained resolution principle is introduced in the framework of a particular logic, called *logic with restricted quantifiers*. In this logic, quantifiers are associated with formulæ interpreted as restriction on the variables of the overall formula. Clausal formulæ with restriction are noted $C \parallel R$, which means $\forall_X R \rightarrow C$, where X is the vector of variables in C. $\mathcal{T}$ denotes the theory of restriction formulæ: it is given in such a way that (un)satisfiability and validity of these formulæ can be decided by an algorithmic mean. Bürckert simply assumes given a class of models for the restriction theory $\mathcal{T}$.

The *RQ-resolution* principle is given by:

$$\frac{\begin{array}{l}\{P(x_1, \cdots, x_n)\} \cup C \parallel R \\ \{\neg P(y_1, \cdots, y_n)\} \cup D \parallel S \\ R \wedge S \wedge \Gamma \text{ is } \mathcal{T}\text{-satisfiable}\end{array}}{C \cup D \parallel R \wedge S \wedge \Gamma} \quad (1)$$

where Γ is the conjunction of equations $x_1 = y_1, \ldots, x_n = y_n$. One of the completeness result from [2] says that given an unsatisfiable set of clauses, it is possible to derive by RQ-resolution an empty clause $\square \parallel R$ such that $\mathcal{T} \models \exists(R)$.

For the purpose of temporal reasoning, we consider that the restriction theory $\mathcal{T}$ is the theory where $\leq$ is interpreted as an ordering relation between time instants and $=$ means that two instants are at the same position on the time line.

To enable the use of constrained resolution, one first needs to have a constrained clausal form of the input formulæ. For example, axiom $\mathcal{A}_1$ in Figure 1 produces the following constrained clause:

$$\leftarrow \{]t_1, t_2], p\} \parallel \neg(t_1 < t_2)$$

while axiom $\mathcal{A}_6$ is transformed in:

$$true(t_2, p) \leftarrow \{]t_1, t_3], p\} \parallel (t_1 < t_2 \leq t_3)$$

The constrained forms of the axioms have a restriction which is a conjunction of temporal constraints. However, some of these constraints are negative literals. To simplify the use of constrained resolution, we choose to assume that $\leq$ is a total ordering relation. This allows to use rewriting rules such as $\neg(t < t') \rightarrow t' \leq t$ to eliminate negative constraints. We also split clauses with disjunctive restriction:

$$C \parallel R_1 \vee R_2 \rightarrow \{\, C \parallel R_1 \,,\, C \parallel R_2 \,\}$$

3.1 Deciding satisfiability

Provided that the restrictions of clauses are conjunction of positive litterals, it is possible to use temporal constraint propagation techniques to decide satisfiability. Thus, the satisfiability of a conjunction of temporal expressions is equivalent to the global consistency of the constraints network built from these expressions.

For our problem, we are interested in both the symbolic and numeric relationships between instants. We choose to rely on the formalism of *Simple Temporal Problem* (STP) studied by Dechter [3]. In this formalism, a constraint between two instants is represented by an edge between two nodes representing the instants, the label of the edge being a numeric interval. Such a constraint is written $x : [a, b] : y$ where x and y are two instants, a and b are two numbers belonginq to $\mathbb{R} \cup \{-\infty, +\infty\}$. This constraint means that $a \leq y - x \leq b$. A set of these constraints gives a network of binary constraints, such that an $O(n^3)$ path consistency algorithm is a complete decision procedure for the global consistency of the constraint set. All expressions comparing instants denoted with numbers, variables, constants and arithmétic terms can be expressed within this constraint formalism.

Some simplification rules for unification [8], especially the ones for decomposition of functional terms,

are used inside the constraint solver when an equality is encountered. The purpose is to: (1) handle non-arithmetic functional terms involved in equations by simplifying these equations; (2) identify equations that involve variables so that they are used to instantiate the resolvent clause. This keeps the set of constraints as small as possible.

Thus, given the set of constraints $R \wedge S \wedge \Gamma$ of rule (1), the satisfiability test produces a pair $\langle \sigma, C \rangle$ where σ is a substitution and C is a set of constraints such that $\sigma(R \wedge S \wedge \Gamma) \equiv C$. Then, the resolution principle is formulated as a variant of Bürckert's one. This gives:

$$\frac{\begin{array}{l}\{P(x_1, \cdots, x_n)\} \cup C \parallel R \\ \{\neg P(y_1, \cdots, y_n)\} \cup D \parallel S \\ R \wedge S \wedge \Gamma \text{ is satisfiable}\end{array}}{\sigma(C \cup D) \parallel \Gamma'} \quad (2)$$

where Γ is the set $\{x_1 = y_1, \ldots, x_n = y_n\}$, and $\langle \sigma, \Gamma' \rangle$ is the pair resulting from the satisfiability test of $R \wedge S \wedge \Gamma$.

4 Abductive temporal reasoning

This section describes an extension of the abductive logic programming procedure described by Kakas et Mancarella in [7]. This extension handles constrained resolution and, contrary to the original one, can handle non ground abducible litterals. The original abductive procedure is an extension of SLD-resolution, and is inspired from the first one described by Eshghi and Kowalski in [4] to handle negation as failure in a abductive fashion.

The definition assumes a logic program P (a set of clauses of the form $C \leftarrow L_1 \ldots L_n$, where C is the *head* and $L_1 \ldots L_n$ the *body*), a set H of predicate symbols called *abducible predicate*, and a set IC of integrity constraints (clauses with empty head). The purpose of the original procedure is to find, for a query Q, a set Δ of hypotheses (ground instances of abducible predicates) such that there exists a stable model M [6] of $P \cup \Delta$ such that $M \models Q$ and $M \models IC$.

4.1 Definition of the procedure

The particular features of the procedure are the following:

- we use the ability for a refutation using constrained resolution to produce "conditional answers", as it is done in Constraint Logic Programming. For this purpose, we consider that, at the end of an abductive refutation, the ground temporal constraints in the restriction of the derived empty clause represent, if they are not satisfied, some additional ordering hypotheses which can be assumed if they are consistent;
- in the same way, it is possible to force a failure in a derivation by assuming some additional constraints. When an empty clause is derived, the constraint part of this empty clause can be made unsatisfiable. The simplest possibility is to add a new temporal constraint to the current set of hypothesis such that the constraints set of the empty clause becomes inconsistent;

The procedure builds interleaved sequences of states. The first sequence form is called an *abductive refutation* where each state has the form $\langle G_i, \Delta_i^t, \Delta_i, \Theta_i, I_i \rangle$. At the beginning, G_0 is the original query. G_i is a goal clause, Δ_i is a set of constraints, and Δ_i is the current set of hypotheses. The set I_i is initialized with the integrity constraints in IC and is used to collect new integrity constraints from the failure in the consistency check part of the procedure.

Definition 1 *Let G be a goal clause of the form $\leftarrow B$. An* abductive refutation *of G is a finite sequence of tuple:*

$$\langle G_1, \Delta_1^t, \Delta_1, \Theta_1, I_1 \rangle \ldots \langle G_n, \Delta_n^t, \Delta_n, \Theta_n, I_n \rangle$$

where G_i is a goal clause, Δ_i^t is a set of ground constraints, Δ_i is a set of ground literals, Θ_i is a substitution, I_i is a set of integrity constraints,

$$\begin{array}{lll} G_1 = G & \Theta_1 = \epsilon & I_1 = IC \\ G_n = \ \Box \parallel R & & \end{array}$$

such that either $\mathcal{T}, \Delta_n^t \models R$ or $\Delta_n^t \wedge R$ is $\mathcal{T}$-satisfiable; and for each $i = 1, \ldots, n$, G_i has the form $\leftarrow L, L' \parallel R$ where L is the selected literal, and the next state $\langle G_{i+1}, \Delta_{i+1}^t, \Delta_{i+1}, \Theta_{i+1}, I_{i+1} \rangle$ is obtained according to one of the following rules:

(A_1) L is positive, C is the resolvent of G_i and of a variant of some clause in P on the selected literal L with the pair $\langle \sigma, \Gamma \rangle$, then:

$$\begin{array}{ll} G_{i+1} = C & \\ \Delta_{i+1}^t = \Delta_i^t & \Delta_{i+1} = \Delta_i \\ \Theta_{i+1} = \sigma & I_{i+1} = I_i \end{array}$$

(A_2) L is either positive and abducible or negative, L unifies with an element of Δ_i with the pair $\langle \sigma, \Gamma \rangle$, $\sigma(R \wedge \Gamma)$ is $\mathcal{T}$-satisfiable, then:

$$\begin{array}{ll} G_{i+1} = \ \leftarrow \sigma L' \parallel \sigma(R \wedge \Gamma) & \\ \Delta_{i+1}^t = \Delta_i^t & \Delta_{i+1} = \Delta_i \\ \Theta_{i+1} = \sigma & I_{i+1} = I_i \end{array}$$

(A_3) L is either positive and abducible or negative, neither L nor its negation unifies with an element of Δ_i, and there exists a consistency derivation from $\langle F_0, \Delta_i^t, \Delta_i \cup \{\sigma L\}, I_i \rangle$ to $\langle \{\}, \Delta_t', \Delta', I' \rangle$ then:

$$\begin{array}{ll} G_{i+1} = \ \leftarrow \sigma L' \parallel \sigma R & \\ \Delta_{i+1}^t = \Delta_t' & \Delta_{i+1} = \Delta' \\ \Theta_{i+1} = \sigma & I_{i+1} = I' \end{array}$$

where σ is a substitution which maps each variable of L to a new skolem constant, and F_0 is the set of all resolvents of the clause $\sigma L \leftarrow$ with clauses of I_i.

A *consistency derivation* implements the test of consistency of an hypothesis. It is very similar in essence to a negation as failure call in logic programming. The aim is to check whether an assumption is consistent with the program P and the current set of hypotheses Δ and of temporal constraints Δ_t. A consistency derivation is a sequence of states of the form $\langle F_i, D_i^t, D_i, I_i \rangle$, where F_i is a set of goal clauses, D_i^t is the set of temporal constraints, and D_i the set of current hypotheses. The set I_i collects the failed goals during the test, so that they will be used with further hypotheses.

Definition 2 *A* consistency derivation *is a finite sequence of tuple*

$$\langle F_1, D_1^t, D_1, I_1 \rangle \ldots \langle F_m, D_m^t, D_m, I_m \rangle$$

such that for each $i \in [1, m]$, F_i *is a set of goal clauses and has the form* $\{ \leftarrow L, L' \parallel R \} \cup F_i'$, F_m *is the empty set,* D_i^t *is a set of ground temporal constraints,* D_i *is a set of ground literals, and* I_i *is a set of integrity constraints, and* L *is selected in the body of* $\leftarrow L, L' \parallel R$. $\langle F_{i+1}, D_{i+1}^t, D_{i+1}, I_{i+1} \rangle$ *is obtained according to one of the following rules:*

(C_1*)* *there exists in* F_i *an empty clause* $C = \square \parallel R'$, *then:*

$$F_{i+1} = F_i - \{C\} \qquad D_{i+1} = D_i$$
$$D_{i+1}^t = D_i^t \cup \{D'\}$$
$$I_{i+1} = I_i$$

where D' *is a ground constraint such that* $R' \wedge D'$ *is inconsistent, and* D_{i+1}^t *is* $\mathcal{T}$*-satisfiable.*

(C_2*)* L *is positive,* $\mathcal{C}$ *is the set of all resolvents of clauses in* P *with the clause* $\leftarrow L, L' \parallel R$ *on the literal* L, *then:*

$$F_{i+1} = \mathcal{C} \cup F_i'$$
$$D_{i+1}^t = D_i^t \qquad D_{i+1} = D_i$$
$$I_{i+1} = \begin{cases} I_i & \text{if } \mathcal{C} \neq \emptyset \\ I_i \cup \{ \leftarrow L, L' \parallel R \} & \text{if } \mathcal{C} = \emptyset \end{cases}$$

(C_3*)* L *is either positive and abducible or negative,* $\mathcal{C}$ *is the set of all resolvents of* $\leftarrow L, L' \parallel R$ *with elements of* D_i *on the literal* L, *then:*

$$F_{i+1} = \mathcal{C} \cup F_i'$$
$$D_{i+1}^t = D_i^t \qquad D_{i+1} = D_i$$
$$I_{i+1} = I_i \cup \{ \leftarrow L, L' \parallel R \}$$

(C_4*)* L *is either positive and abducible or negative,* L *is ground, the opposite of* L *is in* D_i, *then:*

$$F_{i+1} = F_i' \qquad D_{i+1}^t = D_t'$$
$$D_{i+1} = D' \qquad I_{i+1} = I_i$$

(C_5*)* L *is either positive and abducible or negative,* L *is ground, and there exists an abductive refutation from* $\langle \leftarrow \neg L, D_i^t, D_i, \epsilon, I_i \rangle$ *to the state* $\langle \square \parallel R', D_t', D', \Theta', I' \rangle$ *then:*

$$F_{i+1} = F_i' \qquad D_{i+1} = D'$$
$$I_{i+1} = I'$$

and either $D_{i+1}^t = D_t'$ *if* $\mathcal{T}, D_t' \models R'$, *or* $D_{i+1}^t = D_t' \wedge R'$ *if* $\mathcal{T}, D_t' \not\models R'$ *and* $D_t' \wedge R'$ *is* $\mathcal{T}$*-satisfiable.*

For our purpose, the logic program P is made of the constrained clause form of the axioms of Figure 1 together with rules describing domain relationships, such as the clauses of Figure 2 for the YSP example. Axioms $\mathcal{A}_1$, $\mathcal{A}_7$, and $\mathcal{A}_8$ are integrity constraints in the set IC. The predicate symbol *persist* is abducible: this means that whenever a new persistency hypothesis is needed, the consistency check will try to refute goals of the form $begin(t, p)$ and $end(t, p)$ where t falls within the period of the persistency assumption. Of course, the aim is that these refutations fail so that the assumption does not violate integrity constraints.

4.2 Example

Recall the YSP example of Figure 2. The query $Q = \leftarrow end(t, p)$, means that we are interested in finding when the gun will cease to be loaded. Rule R_3 yields the goal $\leftarrow true(4, loaded)$. Axiom $\mathcal{A}_6$ produces the goal $\leftarrow persist(t_1, t_2, loaded) \parallel t_1 < 4 \leq t_2$ where the literal $persist(t_1, t_2, loaded)$ is abducible. We begin a consistency derivation with a set of temporal ordering assumptions $\Delta_t = \{S_1 < 4 \leq S_2\}$, and a set of hypotheses $\Delta = \{persist(S_1, S_2, loaded)\}$, where S_1 and S_2 are new temporal constants. The set of goals that we want to fail is:

$$F_1 = \left\{ \begin{array}{l} \leftarrow \neg persist(S_1, S_2, loaded) \parallel \top, \\ \square \parallel S_2 \leq S_1, \\ \leftarrow end(t, loaded) \parallel S_1 \leq t < S_2, \\ \leftarrow begin(t, loaded) \parallel S_1 < t \leq S_2 \end{array} \right\}$$

The first clause in F_1 disappears because the opposite of the literal $\neg persist(S_1, S_2, loaded)$ is in Δ, and the second clause also disappears when we add the ordering constraints $S_1 < S_2$ to the set Δ_t. At the end of the consistency derivation the set of clauses is empty, and the temporal ordering constraints in Δ_t force S_2 to be equal to 4 and S_1 to T_1. The primary abductive refutation ends and yields the substitution $\{t \mapsto 4\}$ as an answer to the query Q.

One limitation of the procedure lies in the ability to handle repeated events: although the logic is able to describe those situations, the refutation procedure must be protected for infinite queries by a bound on the depth of the refutation. It should be noted that those queries do not lead to subsumption between the current goal and one of its ancestors: it appears to be a "translation" on the time line. At this moment, we do not have any mean to identify this relationship, nor can we characterize "translated" goals with respect to their utility in the refutation process.

5 Conclusions

In this paper, we describe an abductive procedure using a constrained resolution principle. Such a resolution principle is very useful in the area of

temporal reasoning. Constrained resolution allows an important gain on efficiency by reducing non-determinism, which is otherwise too much a trouble in pure resolution-based reasoning methods. The abductive procedure is based on work done by kakas and Mancarella on abductive logic programming. This procedure can be used to handle persistency as an assumption, and for planning problems where the set of computed hypotheses and temporal constraints describes a plan to achieve the requested goal.

References

[1] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[2] Hans-Jürgen Bürckert. *A Resolution Principle for a Logic with Restricted Quantifiers*, volume 568 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin Heidelberg, 1991.

[3] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraints networks. *Artificial Intelligence*, 49:61–95, 1991.

[4] K. Eshghi and R. A. Kowalski. Abduction compared with negation by failure. In G. Levi and M. Martelli, editors, *Logic Programming: Proc. of the Sixth International Conference*, pages 234–254. MIT Press, Cambridge, MA, 1989.

[5] Kave Eshghi. Abductive planning with event calculus. In *Proc. of the 5^{th} Int. Conf. on Logic Programming*, pages 562–579, 1988.

[6] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proc. of ICLP'88*, pages 1070–1080, 1988.

[7] A. C. Kakas and P. Mancarella. On the relation between truth maintenance and abduction. In *Proc. of PRICAI'90*, pages 438–443, 1990.

[8] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Trans. Programming Languages and Systems*, 4(2):258–282, 1982.

[9] Drew V. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

[10] Lode Missiaen. *Localized Abductive Planning with the Event Calculus*. PhD thesis, K.U. Leuven, September 1991.

[11] Erik Sandewall. Non-monotonic entailment for reasoning about time and action Part I : Sequential actions. Research Report LiTH-IDA-R-88-27, Linköping University, September 1988.

[12] Murray Shanahan. Prediction is deduction but explanation is abduction. In *Proc. of the 11^{th} Int. Joint Conference on Artificial Intelligence (IJCAI)*, pages 1055–1060, 1989.

[13] Yoav Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, 33:89–104, 1987.

[14] Mark E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.