About the Use of Time on Argumentative Systems[*]

María Laura Cobo **Guillermo R. Simari**

Department of Computer Science and Engineering – Universidad Nacional del Sur
Laboratory of investigation and Development on Artificial Intelligence (L.I.D.I.A)
Institute of Computer Science and Engineering
Av. Alem 1253 – B8000CPB Bahía Blanca, ARGENTINA
Email: [mlc,grs]@cs.uns.edu.ar

Key Words: Knowledge representation, defeasible reasoning, argumentative systems, temporal reasoning

Abstract

There are many areas in Computer Science where time plays an important role. Artificial Intelligence is one of them. In this particular area *Defeasible Logic Programming* (DeLP) was developed to cope with incomplete and potentially inconsistent information. This formalism combines results from Logic Programming and Defeasible Argumentation. DeLP in particular provides the possibility of representing defeasible information in a declarative way and a defeasible argumentation inference mechanism to warrant conclusions. Although this formalism and Defeasible logic programming in general are very useful has left apart an issue that is crucial on several kind of problems, namely *time*. There are also many developments that face temporal reasoning, in particular *Event Calculus*, but non of them consider defeasible information or what to do if we have incomplete or not completely reliable information. In this work we try to attempt an exploration of a possible combination of these two reasoning areas, temporal and defeasible.

1 Introduction

Time plays a fundamental role in computer science, it is critical in real-time systems, useful in distributed systems and assignment of resources, very helpful in databases, and also in specifying properties of programs and proof of correctness. Temporal references are needed on artificial intelligence particularly in any complex deductive process. In the last decades a lot of formalisms has been proposed to provide temporal reasoning capabilities, some of them are developed for particular issues, such as dealing with events and their consequences, planning activities for agents, etc.

Argumentation systems [CML00] provide a way to formalize and implement defeasible reasoning, allowing reasoning about a changing world where available information is usually incomplete or not completely reliable. They also give us abilities to change conclusions according with the information we count on. Thus the conclusions obtained by the system are *justified* by *arguments* supporting their consideration. These *arguments* can be seen as a 'defeasible proofs' for their respective conclusions.

These two reasoning areas were in general analyzed separately, except for the investigations of Augusto [AS01]. Although there are several places where both aspects are needed, as for example

[*]VI Workshop de Agentes y Sistemas Inteligentes, XI Congreso Argentino de Ciencia de la Computacin, Concordia - Entre Ríos.

e-commerce [PC01, KB01] and aspects of semantic web [SCZ04, BC02]. In this paper we are going to set an initial bound between them, in particular through a particular argumentative system, DeLP [GS04], and a well known temporal logic language, Event Calculus [KS86].

In general, an argumentative system counts with five elements, at least in the abstract layer:

1. *Underlying logical language:* in this particular case we need a temporal-logic language, we choose *Event Calculus*.
2. *Argument definition*
3. *Conflict and rebuttal among arguments*
4. *Argument evaluation*
5. *Notion of defeasible logic consequence:* again in this case it must be *defeasible temporal-logic consequence*

In many cases, the five above mentioned elements are not explicitly defined because they are clearly not independent. In fact dependencies among them allow the identification of four fundamental layers [PV98] in argumentative systems.

- *Logical Layer:* It comprises language definition, inference rules and argument construction.
- *Dialectic Layer:* This layer both involves the definition of conflict between arguments and formalizes the way of solving those possible conflicts.
- *Procedural Layer:* Defines arguments interchange.
- *Strategic Layer:* Present heuristics for argument selection during a debate based on maximizing success possibilities.

In this work we deal with logical and dialectic layers in order to obtain the basis for a temporal argumentative system. Besides we deal with the first three elements, i.e. underlying logical language, argument definition and conflict-rebuttal among arguments.

The rest of the paper is structured as follows. Section 2 summarizes *Event Calculus* language. Next, section 3 reviews the main aspects of DeLP. Section 4 introduces an analysis of the aspects to reach a temporal defeasible system. Finally section 5 concludes the paper.

2 Event Calculus: a new underlying logical language

The selection of representation language is very important because there is a close relationship between arguments and representation language. In this paper we need to count with a language that can express and deal with time and change notions. In this first approximation, we are going to use *Event Calculus*. Although there are a wide set of dialects of this calculus [Sha90, Mue02, MS04], in this case we are going to use a simplified version of it.

Event Calculus was introduced in the eighties by Kowalski and Sergot as a logic programming formalism to represent events and their effects [KS86]. Many dialects have been developed since then, e.g. [Sha90, Mue02, MS04]. In the original language events initiate time periods during which properties hold. Since a property or “fluent” is initiated it holds, unless it is terminated by the occurrence of an event. In Kowalski and Sergot version, a discrete time ontology was chosen to indicate changes.

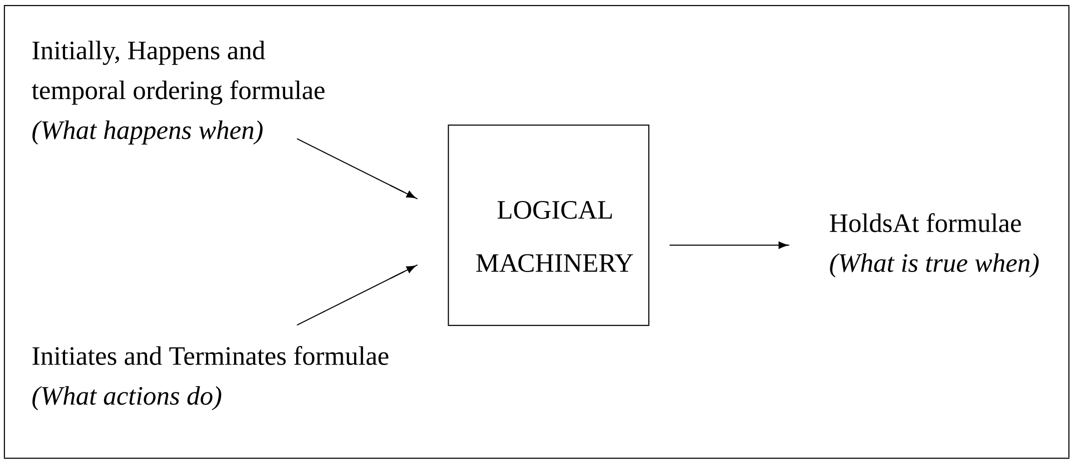

Figure 1: How Event Calculus functions

A particular extension of the language is required in order to represent continuous characteristics. Most known extensions of this calculus were developed by Shanahan [Sha90].

Event Calculus(EC) functions as shown in figure 1. In general it is a logical mechanism capable of making inferences to determine *what is true when* from *what happens when* (knowledge about the state of the world) and *what actions do* (effect of an action on the world). The logical machinery includes arithmetic to set a relation between time references. The kind of arithmetic involved depends on the selected temporal ontology. The basic ontology of the calculus are *actions* or *events, fluent* and *time points*. A *fluent* is anything whose value is subject to change over time. It could be a quantity such as "temperature in a room" or "amount of liquid in a bottle" whose numerical value is subject to variation, or a proposition such as "it is sunny" whose truth value change from time to time. The predicate deals basically with propositional fluents although the other ones are allowed in some dialects. Another important issue in the choice of the ontology is the choice of the predicates. The main predicates used on a simple version of *Event Calculus*, SEC, are:

$happens(E, T)$:	Event E takes place on time moment T.
$holdsAt(F, T)$:	Fluent F holds at time moment T.
$initiates(E, F, T)$:	Fluent F starts to hold after action E at time moment T, and is not freed on $T + 1$.
$terminates(E, F, T)$:	Fluent F ceases to hold after action E at time moment T.
$releases(E, F, T)$:	Fluent F is not subject to inertia after action E at time moment T
$trajectory(F_1, T_1, F_2, T_2)$:	If fluent F_1 is initiated by an even that takes place on T_1, and T_2 is greater than zero, then fluent F_2 holds at $T_1 + T_2$.
$antiTrajectory(F_1, T_1, F_2, T_2)$:	If F_1 is finished by an event that takes place on T_1, and T_2 is greater than zero, then F_2 holds on $T_1 + T_2$.
$clipped(T_1, F, T_2)$:	Fluent F is terminated between times T_1 and T_2.
$initiallyP(F)$:	Fluent F holds from time zero.

Calculus complete axiomatization depends on time ontology. For example if we consider a discrete ontology, we can use ontology presented by Mueller [Mue02] or more completely from Miller and Shanahan research [MS04]. The idea of using a discrete version came along to make easy reasoning problems resolution through satisfaction. Axioms make a bound between predicates, which is extremely important at reasoning time. For the sake of example we can consider the following axioms that are part of SEC axiomatization.

$$\begin{array}{rl} initially(F) \leftarrow & initiallyP(F), \\ & \neg clipped(0, F, T) \\ holdsAt(F, T_2) \leftarrow & happens(A, T_1), \\ & initiates(A, F, T_1), T_1 < T_2, \\ & \neg clipped(T_1, F, T_2) \\ clipped(T_1, F, T_2) \leftarrow & \exists A, T[Happens(A, T), T_1 < T < T_2, terminates(A, F, T_2)] \end{array}$$

In section 4.1 we present some formulas of *Event Calculus* to make an illustration of its expressive power.

3 Reviewing aspects of DeLP

DeLP [GS04] is a language developed in term of three disjoint sets: a set of *facts*, a set of *strict rules* and finally one of *defeasible rules*, where

- A *fact* is a literal, i.e. a ground atom, o a negated ground atom.
- A *strict rule* is an order pair, denoted as "$Head \leftarrow Body$", whose first member is a literal and the second one, $Body$, es finite set of literals. A strict rule can also be written as:

  $$L_0 \leftarrow L_1, \ldots, L_n(n > 0$$

  where L_0 is rule's $Head$ and each $L_i, i \geq 0$ is a literal.
- A *defeasible rule* is also an order pair, noted as $L_0 \prec L_1, \ldots, L_n$. Again L_i is a literal and $i \geq 0$

Notice that strict negation may affect any literal, in particular may affect L_0, i.e. any rules *Head*. At simple sight the only difference between strict and defeasible rules is the way they are denoted, although their meaning is clearly different. In the first kind there are no doubts about the conclusion expressed on the rule, while in the other ones we only assure that we have a "good feeling" about the conclusion but we can not be completely sure about it.

DEFINITION **1** from [GS04]

A Defeasible Logic Program, $\mathcal{P}$, is a possible infinite set of facts, strict rules and defeasible rules. In a program $\mathcal{P}$, we will distinguish the subset Π of facts and rules, and the subset Δ of defeasible rules.

□

Strict and defeasible rules are ground. However we are going to use "schematic rules", i.e. rules with variables. This rules can be grounded through predicate $Ground(\mathcal{R})$ that represents the set of all ground instances of rules in set $\mathcal{R}$. In DeLP there are four possible answers for a query: `YES` , `NO`, `UNDECIDED` and `UNKNOWN`. We refer the interested reader to [GS04].

To keep going on with the basis of DeLP system we still require some definitions as *Defeasible derivation* and of course *Argument Structure*.

DEFINITION **2** *Defeasible Derivation*

A defeasible derivation for a ground literal, l, is a finite sequence of ground literals, each literal is in the sequence because:

1. Literal l is a fact in set Π of the program.
2. There is a rule (strict or defeasible) in the program, whose head is literal l and all the literals in the *Body* appears in the sequence before.

$\square$

Now we can define argument's notion:

DEFINITION **3** *Argument Structure*
Let l be a literal and $\mathcal{P} = (\Pi, \Delta)$ a DeLP program. We say that $\langle \mathcal{A}, l \rangle$ is an argument structure for l, if $\mathcal{A}$ is a set of defeasible rules for Δ, such that:

1. there is a defeasible derivation for l from $\Pi \cup \mathcal{A}$,
2. the set $\Pi \cup \mathcal{A}$ is non-contradictory, and
3. $\mathcal{A}$ is minimal (there is not a subset of $\mathcal{A}$ such that satisfies the previous conditions).

$\square$

EXAMPLE **1** extracted from [GS04]
Consider the following DeLP program $\mathcal{P} = (\Pi, \Delta)$ where

$$\Pi = \left\{ \begin{array}{l} bird(X) \leftarrow chicken(X) \\ bird(X) \leftarrow penguin(X) \\ \neg flies(X) \leftarrow penguin(X) \\ chicken(tina) \\ penguin(tweety) \\ scared(tina) \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{l} flies(X) \prec bird(X) \\ \neg flies(X) \prec chicken(X) \\ flies(X) \prec chicken(X), scared(X) \\ nest\_in\_trees \prec flies(X) \end{array} \right\}$$

The sequence

$$\{chicken(tina), bird(tina), flies(tina)\}$$

is a defeasible derivation for $flies(tina)$, obtained for the following set of rules:

$$\{bird(tina) \leftarrow chicken(tina), flies(tina) \prec bird(tina)\}$$

Note that there is also a derivation for $\neg flies(tina)$ from the sequence $chicken(tina), \neg flies(tina)$ obtained from this other set of rules:

$$\neg flies(tina) \prec chicken(tina)$$

Then there is an argument $\langle \mathcal{A}_1, flies(tina) \rangle$ and there is an argument $\langle \mathcal{A}_2, \neg flies(tina) \rangle$ where:

$$\mathcal{A}_1 = \{flies(tina) \prec bird(tina)\}$$
$$\mathcal{A}_1 = \{\neg flies(tina) \prec chicken(tina)\}$$

$\bullet$

4 Towards a temporal version of DeLP

To add time we need to change the basic language of *DeLP* that implies a change in the three sets previously presented. On the temporal version, literals are going to be predicates from SEC. The others definitions presented are analogous. For that purpose we extend argument notion like this:

DEFINITION **4** *Temporal Argument Structure*
Let $lit = (l, T)$ be a temporal literal and $\mathcal{P} = (\Pi, \Delta)$ a DeLP program. We say that $\mathcal{A}^T = \langle \mathcal{A}, lit \rangle$ is an argument structure for lit, if $\mathcal{A}$ is a set of defeasible rules for Δ, such that verifies definition 3 $\square$

Now lets see some application examples.

4.1 ***Yale Shooting Problem*** example

The *Yale Shooting Problem, YSP* was introduced by Steve Hanks and Drew McDermott [HD87]. It was presented to illustrate the *frame problem,* i.e., the problem of knowing what remains unchanged after an event occurrence, and is largely known in temporal literature. It is about a person who at any point in time is either alive or dead, and about a gun that can be either loaded or unloaded. The gun becomes loaded any time a load action is executed. The person becomes dead any time he is shot with a loaded gun. Assume that the person is initially alive. The gun is loaded, then he waits for a while, and then he is shot with the gun. What can we say, given these assumptions, about the values of fluent involved – alive and loaded – at various points in time?

The description of the domain above does not say whether the load action is considered executable when the gun is already loaded. Let's decide that the answer is yes: when the gun is loaded, that action can be executed but will have no effect.

Note that the assumptions of the Yale Shooting Problem do not determine the initial state completely: the fluent loaded can be either true or false in the initial state. But once the initial value of this fluent is selected, all future changes in the values of fluent are uniquely defined.

The problem then has the following aspects involved:
constants:

- loaded, alive: inertial fluent;
- load, shoot: exogenous actions.

behavior:
- load causes loaded.
- shoot causes not loaded and
- load-shoot are nonexecutable together
- shoot causes not alive.
- shoot is nonexecutable if not loaded.

In the original problem the after sequence $load - shoot$ the person involved in the scenario will be dead. In *Event Calculus* part of available knowledge can be described like this:

$$\begin{aligned}
inititates(load, loaded, T) & \\
teminates(shoot, loaded, T) & \\
initiates(shoot, dead.T) \leftarrow & \quad holdsAt(loaded, T) \\
terminates(shoot, alive, T) \leftarrow & \quad holdsAt(loaded, T)
\end{aligned}$$

$$\begin{aligned}
initiallyP(alive) \\
happens(load, 0) \\
happens(shoot, 2)
\end{aligned}$$

On the other hand, legend has it that assassins have been known to use ice instead of lead in their bullets – it's claimed that after a successful hit, the ice bullet melts without a trace. Are ice bullets possible? If we believe it is possible then we need to rearrange our knowledge. This knowledge must deal with the fact that a person may be alive after a shooting because the ice bullet was melted before shooting, or not in case the bullet was still frozen at shooting time. This kind of knowledge seems to be defeasible, because at simple sight its impossible to determine if a person, let us call her Joe, is alive o not after shooting.

In this new scenario another fluents are needed

- *frozen*: determines if ice-bullet is still frozen or not. Is an inertial fluent except something in the environment changes its condition.
- *lowTemperature*: shows if environment temperature is low enough to keep the bullet frozen. In a way is the fluent that may change the inertial condition, o not, of *frozen* fluent.

and off course now a shoot not always causes a change in alive fluent. We are going to present a formalization of this new scenario, extension of original *YSP* in SEC enriched with defeasible rules.

Let consider the following program $\mathcal{P}_2 = (\Pi, \Delta)$, where:

$$\Pi = \left\{ \begin{array}{l} initiallyP(alive) \\ happens(load, 0) \\ happens(shoot, 2) \\ \\ initiates(load, loaded, T) \\ terminates(shoot, loaded, T) \\ \\ \neg holdsAt(lowTemperature, 0) \\ holdsAt(lowTemperature, 1) \\ holdsAt(frozen, 0) \\ holdsAt(alive, T) \leftarrow holdsAt(alive, T_1), \neg happens(shoot, T_1), T = T_1 + 1 \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{rl} holdsAt(alive, T) \prec & happens(load, T_1), \neg holdsAt(lowTemperature, T_1), \\ & T > T_1 \\ \neg holdsAt(alive, T) \prec & holdsAt(alive, T_1), holdsAt(loaded, T_1), \\ & holdsAt(frozen, T), happens(shoot, T_1), T = T_1 + 1 \\ holdsAt(alive, T) \prec & holdsAt(alive, T_1), holdsAt(loaded, T_1), \\ & \neg holdsAt(frozen, T), happens(shoot, T_1), T = T_1 + 1 \\ \\ holdsAt(frozen, T) \prec & holdsAt(lowTemperature, T_1), \\ & holdsAt(frozen, T_1), \\ & T = T_1 + 1 \\ \neg holdsAt(frozen, T) \prec & \neg holdsAt(lowTemperature, T_1), T = T_1 + 1 \\ holdsAt(frozen, T) \prec & holdsAt(frozen, T_1), \\ & T = T_1 + 1 \\ \neg holdsAt(frozen, T) \prec & \neg holdsAt(frozen, T_1), \\ & T = T_1 + 1 \end{array} \right\}$$

Consider the following sequence

$$holdsAt(frozen, 0), holdsAt(frozen, 1)$$

it is a defeasible derivation for the literal $holdsAt(frozen, 1)$, obtained from the set of rules

$$\mathcal{P}_1 = \left\{\ holdsAt(frozen, 1) \prec holdsAt(frozen, 0), 1 = 0 + 1\ \right\}^{1}$$

At the same time, from program above we have the sequence

$$holdsAt(frozen, 0), \neg holdsAt(lowTemperature, 0), \neg holdsAt(frozen, 1)$$

This sequence is obtained from this set of rules:

$$\mathcal{P}_2 = \left\{\ \neg holdsAt(frozen, 1) \prec \neg holdsAt(lowTemperature, 0), 1 = 0 + 1\ \right\}$$

So we have the following argument:

$$\langle \{holdsAt(frozen, 1) \prec holdsAt(frozen, 0), 1 = 0 + 1\}, holdsAt(frozen, 1)\rangle$$
$$\langle \{\neg holdsAt(frozen, 1) \prec \neg holdsAt(lowTempreature, 0), 1 = 0 + 1\}, \neg holdsAt(frozen, 1)\rangle$$

So we have set of rules that assure opposite values for the same fluent in the same moment of time. The preference of one of them changes the value of another important fluent, the one that sets if Joe is alive or not after a shooting in, let say, moment 2.

Then if we consider fluent $holdsAt(alive, 2)$ it may be considered argument $\mathcal{A}_1^2 = \langle Arg_1, holdsAt(alive, 2)\rangle$ where Arg_1 is the following set of defeasible rules:

$$Arg_1 = \left\{ \begin{array}{rl} holdsAt(alive, 2) \prec & holdsAt(alive, 1), \\ & holdsAt(loaded, 1), \\ & \neg holdsAt(frozen, 2), \\ & happens(shoot, 1), 2 = 1 + 1 \\ \neg holdsAt(frozen, 2) \prec & \neg holdsAt(frozen, 1), 2 = 1 + 1 \\ \neg holdsAt(frozen, 1) \prec & \neg holdsAt(frozen, 0), 1 = 0 + 1 \\ \neg holdsAt(frozen, 0) \prec & \neg holdsAt(lowTemperature, 0), 1 = 0 + 1 \end{array} \right\}$$

and $\mathcal{A}_2^2 = \langle Arg_2, \neg holdsAt(alive, 2)\rangle$ supporting for $\neg holdsAt(alive, 2)$ o against the previous one:

$$Arg_2 = \left\{ \begin{array}{rl} \neg holdsAt(alive, 2) \prec & holdsAt(alive, 1), \\ & holdsAt(loaded, 1), \\ & holdsAt(frozen, 2), \\ & happens(shoot, 1), 2 = 1 + 1 \\ holdsAt(frozen, 2) \prec & holdsAt(lowTemperature, 1), \\ & holdsAt(frozen, 1), 2 = 1 + 1 \\ \neg holdsAt(frozen, 1) \prec & holdsAt(frozen, 0), 1 = 0 + 1 \end{array} \right\}$$

4.2 Amphibians Breathing example

We have an scenario with incomplete information and we can not determine with accuracy what kind of breathing an amphibian has at certain moment. In particular if we are trying to infer this from the fact that the animal is amphibian and growing conditions, for example we may know when birth took place, or behavioral stuff such as if they swim only or if they also jump. Other aspects that can be considered is if metamorphosis of the animal has taken place or not. A good reason to believe that metamorphosis has happened is when an animal has lost its tail unless, we are speaking about some particular amphibian species. In the example $matt$ is an amphibian that never loses his tail because, for example, he could be an iguana.

We consider the following fluent and actions:

[1] Note that this is a grounded version of the rule $holdsAt(frozen, T) \prec holdsAt(frozen, T_1), T = T_1 + 1$

- frog(X), amphibian(X), has_tail(X): inertial fluent.
- branchial(X), pulmonary(X): inertial fluent.
- swim(X), jump(X): exogenous actions.
- metamorphosis(X): action.

The programm is represented through the usual sets, so this is the available information:

$$\Pi = \left\{ \begin{array}{rl} initiatesP(\neg transformed(X)) & \\ initiates(metamorphosis(X), transformed(X), T) & \\ terminates(metamorphosis(X), \neg transformed(X), T) & \\ & \\ holdsAt(frog(renee), T) & \\ holdsAt(amphibian(matt), T) & \\ holdsAt(has\_tail(matt), T) & \\ & \\ happens(swim(renee), 0) & \\ happens(jump(renee), 2) & \\ happens(swim(renee), 1) & \\ happens(born(matt), 0) & \\ & \\ holdsAt(amphibian(X), T) \leftarrow & holdsAt(frog(X), T) \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{rl} holdsAt(branquial(X), T) \prec & holdsAt(amphibian(X), T), \\ & \neg happens(metamorphosis(X), T_1), T_1 < T \\ holdsAt(pulmonar(X), T) \prec & holdsAt(amphibian(X), T) \\ holdsAt(pulmonar(X), T) \prec & holdsAt(amphibian(X), T), \\ & happens(jump(X), T) \\ holdsAt(branquial(X), T) \prec & holdsAt(amphibian(X), T), \\ & happens(swim(X), T), \\ & \neg happens(jump(X), T_1), T_1 < T \\ & \\ happens(metamorphosis(X), T) \prec & \neg happens(metamorphosis(X), T_1), T_1 < T, \\ & happens(swim(X), T_1), \\ & happens(jump(X), T_2), \\ & T_1 < T_2, T = T_2 - 1 \\ happens(metamorphosis(X), T) \prec & \neg happens(metamorphosis(X), T_1), \\ & T_1 < T, \neg holdsAt(has\_tail(X), T) \\ happens(metamorphosis(X), T) \prec & holdsAt(born(X), T_1), T_1 = T + 3 \\ \neg happens(metamorphosis(X), T) \prec & holdsAt(has\_tail(X), T) \end{array} \right\}$$

Let consider what kind of breathing has $matt$ at time point 3. Lets find the argument for fluent $holdsAt(pulmonar(matt), 3)$ and one for $holdsAt(branquial(renee), 1)$. Argument $\mathcal{A}_3^3$ is formed like this:

$$\langle \{holdsAt(pulmonar(matt), 3) \prec holdsAt(amphibian(matt), 3)\}, holdsAt(pulmonar(matt), 3) \rangle$$

$$\mathcal{A}_4^1 = \langle Arg_4, holdsAt(branquial(renee), 1)\rangle$$

where

$$Arg_4 = \left\{ \begin{array}{ll} holdsAt(branquial(renee), 1) \prec & holdsAt(amphibian(renee), 1), \\ & happens(swim(renee), 1), \\ & \neg happens(jump(renee), 0), \\ & 0 < 1 \end{array} \right\}$$

At the same time if we consider if $matt$ suffers metamorphosis or not we can reach to a situation like this one:

- we know that $matt$ born on moment 0 so by defeasible rule

  $$happens(metamorphosis(matt), 3) \prec holdsAt(born(matt), 0), 3 = 0 + 3$$

  we can conclude that $matt$ suffers metamorphosis at time 3.

- we also know that $matt$ has a tail always, so by defeasible rule

  $$\neg happens(metamorphosis(matt), 4) \prec holdsAt(has\_tail(matt), 4)$$

Then, we have the arguments shown on table 4.2.

$$\begin{array}{c} \mathcal{A}_5^3 = \langle Arg_5, holdsAt(transformed(matt), 3)\rangle \\ \mathcal{A}_6^4 = \langle Arg_6, \neg holdsAt(transformed(matt), 4)\rangle \end{array}$$

$$\begin{array}{l} Arg_5 = \left\{ \begin{array}{ll} happens(metamorphosis(matt), 3) \prec & holdsAt(born(matt), 0), \\ & 3 = 0 + 3 \end{array} \right\} \\ Arg_6 = \{\neg happens(metamorphosis(matt), 4) \prec holdsAt(has\_tail(matt), 4)\} \end{array}$$

4.3 Arguments Disagreement

In DeLP answers to queries are supported by arguments. Since arguments are built over defeasible information it may have different arguments some supportive for our query and some not. So we need a way of comparing arguments. Basically this comparing process is based on *disagreement* notion.

DEFINITION **5** *(Disagreement)*
Two literals a and b *disagree* in a program $\mathcal{P} = (\Pi, \Delta)$ if and only if $\Pi \cup \{a, b\}$ is contradictory. □

In general in a non temporal environment *disagreement* is enough to determine if there is a conflict between arguments, there will be arguments or sub-arguments that are in *disagreement*. Recall $\langle \mathcal{B}, q\rangle$ is a sub-argument of argument $\langle \mathcal{A}, p\rangle$ such as $\mathcal{B}$ is a subset of $\mathcal{A}$.

When we consider adding temporal information to the above *disagreement* definition another kind of conflicts may appear. For example if we consider the problem presented on section 4.1, a conflict exists between an argument supporting $\neg holdsAt(alive, 2)$ and one supporting $holdsAt(alive, 3)$. There is no disagreement in terms of definition 6 although there is a conflict at knowledge level that must be taken in consideration. The main problem here is that this is not a generalizable situation, because there are other scenarios or fluents where this change do not generate conflicts. In the other example a similar situation takes place. Specially if we take a look to arguments $\mathcal{A}_5^3$ and $\mathcal{A}_6^4$ in example 4.2 where $\mathcal{A}_5^3$ says that $matt$ suffers metamorphosis at time point 3 while $\mathcal{A}_6^4$ expresses that he still do not suffer it at time 4. Here there is a clear disagreement between arguments, but the disagreement is temporal because it is due to temporal information involved. In this kind of disagreement, although is not explicitly stated, on some moment of time we have conventional disagreement. In the example presented, in one line of reasoning metamorphosis took place and $matt$ is transformed. This transformation is a permanent change. In the other one metamorphosis did not happen so at moment 3 $matt$ was not transformed. Looking closer to both argument a conflict or disagreement take place at moment 4.

Formalizing this idea we reach to the following definition:

DEFINITION **6** *(Temporal Disagreement)*

Two arguments $\mathcal{A}_1^{t_1} = \langle \mathcal{A}_1, (l_1, t_1)) \rangle$, $\mathcal{A}_2^{t_2} = \langle \mathcal{A}_1, (l_2, t_2)) \rangle$ are in *temporal disagreement* of there are time points t_1, t_2 such that $\mathcal{A}_1^{t_1}$, $\mathcal{A}_2^{t_2}$ are in disagreement. Meaning that

$$\Pi \cup \{(l_1, t_1), (l_2, t_2)\} \vdash \perp$$

□

Considering again the program presented in *amphibians breathing example*, see section 4.2, and arguments presented on table 4.2 we have that:

$$\neg holdsAt(trasnformed(matt), 4) \in \Pi \cup \left\{ \begin{array}{c} holdsAt(trasnformed(matt), 3), \\ \neg holdsAt(trasnformed(matt), 4) \end{array} \right\}$$

but we also have that

$$holdsAt(trasnformed(matt), 4) \in \Pi \cup \left\{ \begin{array}{c} holdsAt(trasnformed(matt), 3), \\ \neg holdsAt(trasnformed(matt), 4) \end{array} \right\}$$

5 Conclusions and Future Work

We presented a temporal argument notion through a combination of *Event Calculus* and DeLP, as a way to gain temporal defeasibility systems. The idea is to complement DeLP with other interesting abilities such as temporal capabilities. We presented a proposal for temporal defeasibility through two examples, one widely boarded on temporal literature and a new one. They show situations where defeasibility is involved and also the argument's notion. At the same time we provide a definition for *temporal disagreement* since there are circumstances where traditional definition of disagreement is not expressive enough to find any conflicting situation. As a matter of fact if we only add the conclusion of the argument, as DeLP definition does, the conflict can be disregarded.

This is a first approximation that still requires work, we need to improve some definitions, and determine with accuracy when one temporal argument counter-argues another one, and how to make a choice between arguments. We also need to complete definition of the other layers in order to have a complete temporal argumentation system.

Another interesting way of following this line of investigation might be the consideration of other temporal languages, such as *situation calculus*, a underlying logical language. Research in these directions is currently being pursued.

References

[AS01] Juan Carlos Augusto and Guillermo R. Simari. Temporal defeasible reasoning. *Knowledge and Information Systems*, 3:287–318, 2001.

[BC02] Sabin C. Buraga and Gabriel Ciobanu. A rdf-based model for expressing spatio-temporal relations between web sites. In *WISE '02: Proceedings of the 3rd International Conference on Web Information Systems Engineering*, pages 355–361, Washington, DC, USA, 2002. IEEE Computer Society.

[CML00] Carlos I. Chesñevar, Ana G. Maguitman, and Ron Loui. Logical models of argument. *ACM Computing Surveys*, 4(32):337–383, 2000.

[GS04] Alejandro J. Garcia and Guillermo R. Simari. Defeasible logic programming: an argumentative approach. *TPL*, 4:95–138, 2004.

[HD87] Steve Hanks and Drew Mc Dermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[KB01] Ryszard Kowalczyk and Van Bui. On constraint-based reasoning in e-negotiation agents. In *Agent-Mediated Electronic Commerce III, Current Issues in Agent-Based Electronic Commerce Systems (includes revised papers from AMEC 2000 Workshop)*, pages 31–46, London, UK, 2001. Springer-Verlag.

[KS86] Robert Kowalski and Marek Sergot. A logic-based calulus of events. *New Generation Computing*, 4(1):67–895, 1986.

[MS04] R. Miller and Murray Shannahan. Some alternative formulations of the event calculus. *Computational Logic: Logic Programming and Beyond*, 14:703–730, 2004.

[Mue02] Erik T. Mueller. Event calculus reasoning through satisfiability. *Journal of Logic and Computation*, pages 452–490, 2002.

[PC01] Matteo Pradella and Marco Colombetti. A formal description of a practical agent for e-commerce. In *Agent-Mediated Electronic Commerce III, Current Issues in Agent-Based Electronic Commerce Systems (includes revised papers from AMEC 2000 Workshop)*, pages 84–95, London, UK, 2001. Springer-Verlag.

[PV98] Henry Prakken and Gerard Vreeswijk. Logics for defeasible argumentation. Kluwer Academic Publishers, 1998.

[SCZ04] Monika Solanki, Antonio Cau, and Hussein Zedan. Augmenting semantic web service descriptions with compositional specification. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 544–552, New York, NY, USA, 2004. ACM Press.

[Sha90] Murray Shanahan. Representing continuous change in the event calculus. In *Proceedings ECAI 90*, pages 598–603, 1990.