Applications of Linear Defeasible Logic: combining resource consumption and exceptions to energy management and business processes

Francesco Olivieri[1], Guido Governatori[1],
Claudio Tomazzoli[2], Matteo Cristani[2]
[1]Data61, CSIRO, Australia
[2]University of Verona, Italy

Abstract. Linear Logic and Defeasible Logic have been adopted to formalise different features of knowledge representation: consumption of resources, and non monotonic reasoning in particular to represent exceptions. Recently, a framework to combine sub-structural features, corresponding to the consumption of resources, with defeasibility aspects to handle potentially conflicting information, has been discussed in literature, by some of the authors. Two applications emerged that are very relevant: energy management and business process management. We illustrate a set of guide lines to determine how to apply linear defeasible logic to those contexts.

1 Introduction

Resource-based logical framework have been employed to accommodate varieties of methods that propose to introduce relevant limits to the reasoning process. In particular, in the idealised representation of rational agents, these agents have unlimited reasoning power, complete knowledge of the environment and their capabilities, and unlimited resources. Over the years, a few approaches (using different logics) have been advanced to overcome some of these ideal (and definitely unrealistic) assumptions.

In [15, 21, 22], the authors propose the use of Linear Logic to model the notion of resource utilisation, and to generate which plans the agent adopts to achieve its goals. In the same spirit, [13, 12] address the problem of agents being able to take decisions from partial, incomplete, and possibly inconsistent knowledge bases, using (extensions of) Defeasible Logic (a computational and proof theoretic approach) to non-monotonic reasoning and reasoning with exceptions. While these last two approaches seem very far apart, they are both based on proof theory (where the key notion is on the idea of (logical) derivation), and both logics (for different reasons and different techniques) have been used for modelling business processes [14, 28, 8, 24, 20, 18, 9].

Formally, a business process can be understood as a compact representation of a set of traces, where a trace is a sequence of tasks. A business process is hence equivalent to a set of plans with possible choices, or a partially specified plan. The idea behind the work mentioned above is to allow agents to use their deliberation phase to determine the business processes (instead of the plans) to execute.

In this paper, we discuss some applications of the combination of linear logic and a computationally oriented non-monotonic formalism known as defeasible logic. The *linear defeasible logic* has been introduced recently [19].

To provide the above mentioned framework we introduce here a notion of logic as a rule-based system. We can distinguish rules (or sequents, or instances of a consequence relation) and inference (or derivation) rules. A rule specifies that some consequences follow

from some premises, while a derivation rule provides a recipe to determine the valid steps in a proof or derivation. A classical example of a derivation rule is Modus Ponens (i.e., from '$\alpha \to \beta$' and α to derive β). A rule can be understood as a pair

$$\Gamma \vdash \Theta,$$

where Γ and Θ are collections of formulas in an underling language. In Classical Logic, Γ and Θ are sets of formulas and in Intuitionistic Logic Θ is a singleton. Thus the rules

$$\alpha, \beta \vdash \gamma, \delta \qquad \text{and} \qquad \beta, \alpha \vdash \delta, \gamma$$

are the same rule. Where "," in the antecedent Γ is understood as conjunction and disjunction in the consequent Θ. In substructural logics (e.g., Lambek Calculus, and the family of Linear Logics), Γ and Θ are assumed to have an internal structure, and they are considered as multi-sets or sequences. An interpretation of a rule is how to transform the premises in the conclusion. Thus the rule

$$\alpha, \beta, \alpha \vdash \gamma$$

can be taken to mean that we need two instances of α with an instance of β in between to produce an instance of γ. Derivation rules, on the other hand, tell us how to combine rules, to obtain new rules. For example, the derivation rule

$$\frac{\Gamma \vdash \alpha \qquad \Theta \vdash \alpha \to \beta}{\Gamma, \Theta \vdash \alpha, \alpha \to \beta, \beta}$$

establishes whether we have a derivation of α from Γ and a derivation of $\alpha \to \beta$ from Θ, then we can combine the Γ and Θ, to obtain a derivation, where we have α followed by $\alpha \to \beta$ and then β. If the formulas denote activities (or tasks) and resources, then the consequent is a sequence of tasks describing the activities to be done (and the order in which they have to be executed) to produce an outcome (and also, what resources are needed). Thus, we can use the rules to model transformation in a business processes, and derivations as the traces of the process (or the ways in which the process can be executed or the runs of system).

A formalism that properly models processes should feature some key characteristics, and one of the most important ones is to identify which resources are *consumed* after a task has finished its execution. Consider the notorious vending machine scenario by Girard, where the dollar resource is spent to *produce* a can of cola. once we get the cola, the dollar resource is no longer spendable. However, the specifications of a process may include thousands of rules to represent at their best all the various situations that may occur during the execution of the process itself: situations where the information at hand may be incomplete and, sometimes, even contradictory, and rules encoding possible exceptions. This means that we have to adopt a formalism that is able to represent and reason with exceptions, and partial information.

Defeasible Logic (DL) [16] is a non-monotonic rule based formalism, that has been used to model exceptions and processes. The starting point being that, while rules define a relation between premises and conclusion, DL takes the stance that multiple relations are possible, and it focuses on the "strength" of the relationships. Three relationships are identified: *strict rules* specifying that every time the antecedent holds the consequent hold; *defeasible rules*, when the antecedent holds then, normally, the consequent holds; and *defeaters* when the antecedent holds the opposite of the consequent might not hold. An example of rules with a baseline condition and exception is the scenario the outcome of inserting a dollar coin in a vending machine is that we get a can of cola, unless the machine is out of order, or the machine is switched off. Thus, we can represent this scenario with the

rules[1]:

$$r_1 : 1\$ \Rightarrow cola$$
$$r_2 : OutOfOrder \Rightarrow \neg cola$$
$$r_3 : Off \Rightarrow \neg cola.$$

In the rest of the paper we shall provide a (brief) introduction to the formalism of linear defeasible logic, and then discuss two specific cases in which that fornalism applies to energy management and business processes. In particular we dedicate Section 2 to the discussion of the properties introduced in the system discussed in [19], further detailed in Section 3. Section 4 is indeed devoted to the discussion of the applications of Linear Defeasible Logic. Section 5 reviews some relevant related work, and Section 6 discusses some further works and takes conclusions.

2 Desired properties

We dedicate this section to detailing which features are needed for linear defeasible logic. This section refers to the identified properties in [19]. The properties are summarised in a short way.

Ordered list of antecedents. Given the rule '$r : a, b \Rightarrow c$', the order in which we derive a and b is typically irrelevant for the derivation of c. As such, r may indistinctively assume the form '$b, a \Rightarrow c$'. Consider a login procedure requiring a username and password. Whether we insert one credential before the other does not affect a successful login.

Nonetheless, sometimes it is meaningful to consider an *ordered* sequence of atoms in the head of a rule, instead of an *unordered* set of antecedents.

Naturally, in the same set/list of antecedents, combinations of sequences and multi-sets of literals is possible. For instance, "$r : a; b; (c, d); e \Rightarrow f$" represents a situation where, in order to obtain f, we need to first obtain a, then b, then either c or d in any order, and lastly, only after both c and d are obtained, we need to obtain e.

The notation ';' is used as a separator in sequences, while ',' separates in multi-sets.

Multi-occurrence/repetitions of literals. From these ideas, it follows that some literals may appear in multiple instances, and that two rules such as "$r : a; a; b \Rightarrow c$" and "$s : a; b; a \Rightarrow c$" are semantically different. Rule r may describe a scenario where the order of a product may require two deposit payments followed by a full payment prior to delivery.

Resources consumption. Assume we have two rules, "$r : a, b \Rightarrow d$" and "$s : a, c \Rightarrow e$". If we are able to derive a, b and c, then d and e are subsequently obtained. Deducing both d and e is a typical problem of *resource consumption*.

In this paper, we assume all literals to be consumable, since the treatment/derivation of non-consumable literals is the same as in Standard DL (SDL).

In our framework, deriving a literal means to produce the corresponding resource. After it has been produced, it can be 'consumed' if it appears in the antecedent of a rule. Naturally, since we have complete knowledge of our theory, we know exactly in which rules such a literal appears as an antecedent. Not consuming a resource when such is available means that either no rule is applied, or that the rule where the literal appears in as antecedent was *blocked* by a rule for the opposite.

[1] r_i is the name of rule i, symbol $\Rightarrow$ denotes rules meant to derive *defeasible* conclusions, i.e. conclusions which may be defeated by contrary evidence. As it will be clear in Section 3, defeasible rules, defeaters, and the superiority relation represent the non-monotonic aspects of our framework.

Concurrent production. Symmetrically, we consider two rules with the same conclusion: "$r : a \Rightarrow c$ " and "$s : b \Rightarrow c$". It now seems reasonable that, if both a and b are derived, then we conclude two instances of c (whereas in classical logics we only know that c holds true). For example, consider a family where it is tradition to have pizza on Friday evening. Last Friday, the parents were unable to communicate with each other during the day, and one baked the pizza while the other bought take-away on the way home.

However, there exists consistent cases where multiple rules for the same literal produce only *one* instance of the literal (even if they all fire). For example, both a digital or handwritten signature would provide permission to proceed with a request. The same request does not require permission twice: either it is permitted, or it is not.

Resource consumption: A team defeater perspective. Sceptical logics provide a means to decide which conclusion to draw in case of contradicting information. A superiority relation is given among rules for contrary conclusions: it is possible to derive a conclusion only if there exists a *single* rule stronger than *all* the rules for the opposite literal.

DL handles conflicts differently, and the idea here is that of *team defeater*. We do not look at whether there is a single rule prevailing over all the other rules, but rather whether there exists a *team* of rules which can jointly defeat the rules for the contrary conclusion. That is, suppose rules r', r'' and r''' all conclude P, whilst s' and s'' are for $\neg P$. If $r' > s'$ and $r'' > s''$, then the team defeater made of $\{r', r''\}$ is sufficient to prove P.

The focus remains on resource consumption and production. As such, the questions we need to answer are, again, which resources are consumed, and how many instances of the conclusion are derived.

Consider the process of writing a scientific publication. If the paper is accepted, the *manuscript* resource is consumed: it cannot be submitted again. On the contrary, if the paper is rejected, the *manuscript* resource is *not* consumed since it can be submitted again.

Multiple conclusions and resource preservation. Consider internet shopping. As soon as we pay for our online order, the bank account balance decreases, the seller's account increases. Both the seller and the web site have the shipping address.

The conclusion of a rule is usually a single literal but the above example suggests that a single rule may produce more than one conclusion. We thus allow rules to have multiple conclusions, like "$r : a, b \Rightarrow c, d$".

Similar to our discussion on the ordering of antecedents, we may have any combination of ordered/unordered sequences of conclusions. In the previous example, only after we have provided the credit card credentials, our bank account decreases, whilst we can provide the shipping address before the credit card credentials, or the other way around.

The notion of multiple conclusions, along with the discussion on team defeaters, leads to another problem. Consider the two rules "$r : a \Rightarrow b; c; d$" and "$s : e \Rightarrow \neg c$", where no superiority is given. Do we conclude that b or d? What happens if now we have "$r : a \Rightarrow b, c, d$", and s is stronger then r? Do we conclude that b and d (i.e., only the derivation of c has been blocked by s), or will the production of b and d be affected also?

Loops. The importance of being able to properly handle loops is evident: loops play a fundamental role in many real life applications, from business processes to manufacturing. Back to the login procedure, if one of the credentials is wrong, the process loops back to a previous state, for instance, by asking the user to re-enter both credentials.

3 Language and logical formalisation of RSDL

The logic presented here considers consumable literals only: the formalisation of non-consumable literals is the same of that in SDL [1], and thus the process would not add any

value to this contribution. Also we will have either multi-sets or sequences of literals, but *not* combinations of those.

In linear defeasible logic there are two types of derivations: *strict* and *defeasible*. *Strict rules* derive indisputable conclusions, i.e., conclusions that which are always true. Thus, if two strict rules have opposite conclusions, then the resulting logic is inconsistent. On the contrary, defeasible rules are to derive pieces of information that can be defeated by contrary evidence. Finally, defeaters are special type of rules whose purpose is to block contrary evidence: they cannot be used to directly derive conclusions, but only to prevent other rules to conclude.

In [19] it is introduced the language of Resource-driven Substructural Defeasible Logic (RSDL). PROP is the set of propositional atoms, the set Lit = PROP $\cup \{\neg p | p \in \text{PROP}\}$ is the set of literals. The *complement* of a literal p is denoted by $\sim q$; if q is a positive literal p, then $\neg q$ is $\neg p$, and if q is a negative literal $\neg p$, then $\sim q$ is p.

Definition 1. *Let* Lab *be a set of arbitrary labels. Rules have form "$r : A(r) \hookrightarrow C(r)$":*

1. *$r \in$ Lab is a unique name.*
2. *$A(r)$ is the* antecedent*, or* body*, of the rule. $A(r)$ can either have the form $A(r) = a_1, \ldots, a_n$ to denote a multi-set, or $A(r) = a_1; \ldots; a_n$ to denote a sequence.*
3. *$\hookrightarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$ denotes a strict rule, a defeasible rule, and a defeater, respectively.*
4. *$C(r)$ is the* consequent*, or* head*, of the rule. For the head, we consider three options: (a) The head is a single literal 'p', (b) The head is a multi-set '$p_1, \ldots, p_m$', and (c) The head is a sequence '$p_1; \ldots; p_m$'.*

Given a set of rules R and a rule $r : A(r) \hookrightarrow C(r)$, we use the following abbreviations: (i) R_s is the subset of strict rules, (ii) R_{sd} is the set of strict and defeasible rules, (iii) $R[p; i]$ is the set of rules where p appears at index i in the consequent where the consequent is a sequence, (iv) $R[p, i]$ when the consequent is a multi-set containing p.

Definition 2. *A* resource-driven substructural defeasible theory *is a tuple $(F, R, \succ)$ where: (i) $F \subseteq$ Lit are pieces of information denoting the resources available at the beginning of the computation. This differs strikingly from SDL, where facts denote* always-true *statements; (ii) R is the set of rules; (iii) $\succ$, the superiority relation, is a binary relation over R.*

A theory is *finite* if the sets of facts and rules are finite. In SDL, a *proof* P of length n is a finite sequence $P(1), \ldots, P(n)$ of *tagged literals* of the type $\pm\Delta p$, $\pm\partial p$. The idea is that, at every step of the derivation, a literal is either proven or disproven.

In Linear Defeasible Logic, we must be able to derive multiple conclusions in a single derivation step, and hence we require a mechanism to determine when premises have been used to derive a conclusion. Accordingly, we modify the definition of proof to be a matrix.

Definition 3. *A proof P in RSDL is a finite matrix $P(1, 1), \ldots, P(l, c)$ of* tagged literals *of the type $\pm\Delta p$, $\pm\partial p$, $+\sigma p$, $+\Delta p^{\checkmark}$, and $+\partial p^{\checkmark}$.*

We assume that facts are simultaneously true at the beginning of the computation. Notation $+\#p^{\checkmark}$, $\# \in \{\Delta, \partial\}$ denotes that p has been consumed. The distinctive notation for when a literal is proven and when it is consumed will play a key role to determine which rules are applicable. The tagged literal $\pm\Delta p$ means that p is *strictly proved/refuted* in D, and, symmetrically, $\pm\partial p$ means that p is *defeasibly proven/refuted*, $+\sigma p$ indicates that there are applicable rules for p, but the resources for such rules have already been used.

The set of positive and negative conclusions is named *extension*. In Standard Defeasible Logic, given a set of facts, a set of rules, and a superiority relation, the extension is unique. This is not the case when resource consumption and ordered sequences are to be taken into

account: depending on the order in which the rules are applied, (rather) different extensions can be obtained. In RSDL every distinct derivation corresponds to an extension.

In SDL, derivations are based on the notions of a rule being *applicable* or *discarded*. Intuitively, a rule is applicable when every literal in the antecedent has been proven at a previous step. We report hereafter the defeasible proof tag in SDL to give the reader a better understanding of how defeasible conclusions can be drawn.

If $P(n+1) = +\partial p$ then
(1) $\exists r \in R_{sd}[p]$: r is applicable and
(2) $\forall s \in R[\sim p]$ either (2.1) s is discarded or (2.2)$\exists t \in R[p]$: t is applicable and $t \succ s$.

A literal is defeasibly proven when there exists an applicable rule for such a conclusion and all the rules of the opposite are either discarded, or defeated by stronger rules. (Strict derivations only differ in that, when a rule is applicable, we do not care about contrary evidence, and the rule will always produce its conclusion nonetheless.)

In [19] authors proceed incrementally. First, they provide definitions of derivability for multi-sets, and further, definitions for sequences. In both cases, rules may have only one *single* literal for conclusion.

Definition 4. *A rule r is* #-applicable*, $\# \in \{\Delta, \partial, \sigma\}$, at $P(l+1, c+1)$ iff $\forall\ a_i \in A(r)$ then $+\#a_i \in P[(1,1)..(l,c)]$. Moreover, we say that r is* #-consumable *iff r is #-applicable and $\exists\ l' \leq l$ such that $P(l', c) = +\#a_i$.*

A rule is consumable if it is applicable and, for every literal in its antecedent, there is an *unused* occurrence. Discardability is the strong negation of applicability.

Definition 5. *A rule r is* #-discarded*, $\# \in \{\Delta, \partial\}$, at $P(l+1, c+1)$ iff $\exists\ a_i \in A(r)$ such that $-\#a_i \in P[(1,1)..(l,c)]$. Moreover, we say that r is* #-non–consumable *iff either r is discarded, or $\forall\ l' \leq l$, $P(l', c) \neq +\partial a_i$.*

Lastly, we define the conditions describing when a literal is consumed.

Definition 6. *Given rule r, a literal $a \in A(r)$ is #-consumed, $\# \in \{\Delta, \partial\}$, at $P(l+1, c+1)$, iff 1. $\exists\ l' \leq l$ such that $P(l', c) = +\#a$, and 2. $P(l', c+1) = +\#a^{\checkmark}$.*

Naturally, two mutually exclusive extensions are possible, based on whether $+\partial b$ is used by r_1 to derive $+\partial c$, or by r_2 to derive $+\partial d$.

Proof tags for multi-sets in the antecedent and single conclusion

We begin with: (1) the antecedent is a multi-set, (2) single literal in the conclusion.

$+\Delta$: If $P(l+1, c+1) = +\Delta p$ then
(1) $p \in F$, or (2) $\exists r \in R_s[p]$ s.t. r is Δ-consumable and $\forall a_j \in A(r)$, a_j is Δ-consumed.

Literal q is definitely provable if either is a fact, or there is a strict consumable rule for p. Condition (2) actually consumes the literals by replacing $+\Delta a_j$ with $+\Delta a_j^{\checkmark}$.

$-\Delta$: If $P(l+1, c+1) = -\Delta p$ then
(1) $p \notin F$ and (2) $\forall r \in R_s[p]$, r is Δ-non-consumable.

Literal q is definitely refuted if p is not a fact, and every rule for p is non-consumable.

$+\partial$: If $P(l+1, c+1) = +\partial p$, then
(1) $+\Delta p \in P(l, c)$ or
(2) (1) $-\Delta{\sim}p \in P(l, c)$ and
(2) $\exists r \in R_{sd}[p]$ ∂-consumable and (3) $\forall s \in R[{\sim}p]$ either
(1) s is ∂-discarded, or
(2) $\exists t \in R[p]$ ∂-consumable, $t \succ s$, and
(3) if $\exists w \in R[{\sim}p]$ ∂-applicable, $t \succ w$, then
(1) $\forall a_j \in A(t)$, a_j is ∂-consumed, otherwise (2) $\forall a_k \in A(r)$, a_k is ∂-consumed.

Condition (1) is to inherit a defeasible derivation from a definite one. Condition (2.1) ensures that the logic is sound. Condition (2.2) requires that there is a rule r that is triggered by literals that: (i) have been previously proven, (ii) have not yet been consumed. Clause (2.3.1) is that, to rebut an attacking argument, either it is possible to show that some of its premises have been refuted, or is defeated by stronger, consumable rules. The final part is to determine which resources are consumed during the derivation of p. This variant assumes that only the rules in the winning team defeater consume resources, whilst the 'defeated rules' do not.

For $-\partial$ authors of [19] use the strategy similar to that used in [2] to provide proof conditions for the ambiguity propagating variant of SDL, that is, we make it easier to attack a rule (2.2.2). Also, derivations literals tagged with $-\partial$ do not consume of resources, in line with what above.

$-\partial$: If $P(l+1, c+1) = -\partial p$, then
(1) $-\Delta p \in P[l, c]$ and
(2)(1) $+\Delta{\sim}p \in P[l, c]$ or
(2) $\forall r \in R_{sd}[p]$ either
(1)r is ∂-discarded or (2) $\exists s \in R[{\sim}p]$ s.t.
(1)s is σ-applicable and (2) $\forall t \in R[p]$ either t is ∂-discarded or not $t \succ s$.

The idea behind $+\sigma$ is that there are applicable, non-defeated rules for the consequent, irrespective whether the premises have been used or not.

$+\sigma$: If $P(l+1, c+1) = +\sigma p$ then
(1) $\Delta p \in P[l, c]$ or
(2) $\exists r \in R_{sd}[p]$ s.t. (1) r is σ-applicable and (2) $\forall s \in R[{\sim}p]$ either s is ∂-discarded or $s \not\succ r$.

Assume that, at $P(4, 4)$, r_0 is taken into consideration; r_0 is consumable, but so is r_3, and no superiority is given between r_0 and r_3. Actually, r_2 is consumable and stronger than r_3. Accordingly, the team defeater allows us to prove $+\partial d$, and only b is consumed in this process. Thus a is still available, and can be used at $P(5, 5)$ to get $+\partial e$, via r_1. Note that we do not consume resource $+\Delta c$, since $r_2 \succ r_3$.

Proof tags for sequences in the antecedent and single conclusion

When considering sequences in the antecedent, definitions of applicable, discarded, and consumable must be revised. A rule is *sequence applicable* when the derivation order reflects the order in which the literals appear in the antecedent,

Definition 7. *A rule $r \in R[p]$ is* #-sequence applicable*, $\# \in \{\Delta, \partial\}$, at $P(l+1, c+1)$ iff for all $a_i \in A(r)$:*
1. *there exists $c_i \leq c$ such that $P(l_i, c_i) = +\#a_i$, $l_i \leq l$, and*
2. *for all $a_j \in A(r)$ such that $i < j$, then*
3. *for all $c_j \leq c$ such that $P(l_j, c_j) = +\#a_j$, $l_j \leq l$ then $l_i < l_j$ and $c_i < c_j$.*

We say that r is #-sequence consumable *iff is* #-applicable *and 4. $P(l_i, c) = +\#a_i$.*

A rule is sequence discarded when there exists a literal in the antecedent, which has been previously disproven, or there are two proven literals in the antecedent, say a and b, such that a appears before b, and one proof for b is before every proof for a.

Definition 8. *A rule $r \in R[p]$ is* #-sequence discarded, *$\# \in \{\Delta, \partial\}$, at $P(l+1, c+1)$ iff for there exists $a_i \in A(r)$ such that either*

1. *$-\#a_i \in P(l-1, c-1)$, or*
2. *for all $c_i \leq c$ such that $P(l_j, c_i) = +\#a_i$, $l_i \leq l$, then*
   - *there exists $a_j \in A(r)$, with $i < j$, such that*
   - *there exists $c_j \leq c$ such that $P(l_j, c_j) = +\#a_j$, $l_j \leq l$, and $c_i > c_j$, $l_i > l_j$.*

The definition of being #-consumed remains the same as before. The proof tags for strict and defeasible conclusions with (i) sequences in the antecedent, and (ii) a single conclusion can be obtained by simply replacing: (a) #-applicable with #-sequence applicable, (b) #-consumable with #-sequence consumable, and (c) #-discarded with #-sequence discarded. Assume the rules are activated in this order: first r_1, then r_2, last r_3. Thus, $P(4,4) = +\partial a$, $P(5,5) = +\partial a$, and $P(6,6) = +\partial b$. The derivation order between b and the second occurrence of a has not been complied with, and r_0 is sequence discarded. Same if the order is 'r_3, r_1, r_2', whilst 'r_2, r_3, r_1' is a legit order to let r_0 be sequence applicable.

Proof tags for sequences in both the antecedent and conclusion

Even when we consider sequences in the consequent, a literal's strict provability or refutability depends only upon whether the strict rule (where the literal occurs) is sequence consumable or not. As such, given a strict rule $r \in R_s[p; j]$, still p's strict provability/refutability depends only upon whether r is strictly sequence consumable or not. However, now we also have to verify that, if $r \in R_s[q; j-1]$, we prove p immediately after q. The resulting new formalisations of $+\Delta$ and $+\partial$ are as follows (negative proof tags are trivial and thus omitted):

$+\Delta$: If $P(l+1, c+1) = +\Delta p$ then
(1) $p \in F$, or (2) (1) $\exists r \in R_s[p; j]$ s.t. (2) r is Δ-sequence-consumable,
(3) $r \in R_s[q; j-1]$ and $P(l+i-1, c+1) = +\Delta q$, (4) $\forall a_j \in A(r)$, a_j is Δ-consumed.

$+\partial$: If $P(l+i, c+1) = +\partial p$, then
(1) $+\Delta p \in P(l, c)$ or (2) (1) $-\Delta{\sim}p \in P(l, c)$ and
(2) (1) $\exists r \in R_{sd}[p; j]$ ∂-sequence-consumable and
(2) $\exists r \in R[q; j-1]$ and (3) $P(l+i-1, c+1) = +\partial q$, and
(3) $\forall s \in R[{\sim}p]$ either
(1) s is ∂-sequence-discarded, or
(2) $\exists t \in R[p]$ ∂-sequence-consumable, $t \succ s$, and
(3) if $\exists w \in R[{\sim}p]$ ∂-sequence-applicable, $t \succ w$, then
(1) $\forall a_j \in A(t)$, a_j is ∂-consumed, otherwise (2) $\forall a_k \in A(r)$, a_k is ∂-consumed.

Proof tags for sequences in the antecedent and multi-sets in the conclusion

We now consider multi-sets in the conclusion. Strict provability does not change with respect to the one described in the previous section, and is therefore omitted. When considering a 'team defeater fight', two variants are possible. In this first variant, we draw a conclusion only if there is a winning team defeater for each literal in the conclusion.

$+\partial$: If $P(l+i, c+1) = +\partial p$, then
(1) $+\Delta p \in P(l, c)$ or (2) (1) $-\Delta{\sim}p \in P(l, c)$ and
(2) $\exists r \in R_{sd}[p, j]$ ∂-sequence-consumable and
(3) $\forall s \in R[{\sim}q]$ such that $q \in C(r)$ either (1) s is ∂-sequence-discarded, or
(2) $\exists t \in R[q]$ ∂-sequence-consumable, $t \succ s$, and
(3) if $\exists w \in R[{\sim}p]$ ∂-sequence-applicable, $t \succ w$, then
(1) $\forall a_j \in A(t)$, a_j is ∂-consumed, otherwise (2) $\forall a_k \in A(r)$, a_k is ∂-consumed.

In this latter variant, we limit the comparison on the individual literal.

$+\partial$: If $P(l+i, c+1) = +\partial p$, then
(1) $+\Delta p \in P(l, c)$ or
(2) (1) $-\Delta{\sim}p \in P(l, c)$ and
(2) $\exists r \in R_{sd}[p, j]$ ∂-sequence-consumable and
(3) $\forall s \in R[{\sim}p]$ either
(1) s is ∂-sequence-discarded, or
(2) $\exists t \in R[p]$ ∂-sequence-consumable, $t \succ s$, and
(3) if $\exists w \in R[{\sim}p]$ ∂-sequence-applicable, $t \succ w$, then
(1) $\forall a_j \in A(t)$, a_j is ∂-consumed, otherwise
(2) $\forall a_k \in A(r)$, a_k is ∂-consumed.

4 Applications of linear defeasible logic

This section is devoted to the description of two application scenarios for the combination of resource consumption and exceptions into a logical framework. In both scenarios, the basic idea of the application is to provide an abstract formalisation of a context. In the first case scenario, we shall discuss how a reasoning process can provide decision onto a set of rules that connect energy resources in a defeasible way. In the second case scenario we discuss two aspects of business process management that deserve linear defeasible logic treatment.

In both cases, the discriminant for choosing a logical framework sites on the reasoning capability of such frameworks. Ideally, we can specify the rules that govern a context and compute a solution to these rules, that consist, in one case, of a plan of energy resource production and consumption, and in the other case, of a process description.

Description of the application scenarios is made entirely in an informal way, for the sake of conciseness. Formalization is left to further work.

4.1 Energy management

A top-level description of an energy production and consumption context is defined by a set of rules and facts, where rules are employed to describe defeasible transformations of energy, and facts represent energy resources. We provide here three basic cases of application

Scenario 1 ***Energy saving***
One of the most difficult concepts to deal with in the definition of methods for energy saving is the definition of the comparison of processes (the so called consumption profiles*, in order to detect the best solution. Consider a classical home configuration, as represented in Figure 1.*

The most common approach is ranking*, that is based upon the usage of multiplicative or additive measures attached to the devices, able to represent the* typical *concumption, usually, the* average *estimated consumption. This approaches have shown to be very inaccurate in forecasting the behaviour, as numerous* exceptions *can significantly change the resulting value of energy, for the local measure used typically is* absorbed power*, and energy is instead the integral of power. If the sampling processes that has generated the above defined scenario, has valued the absorbed power in instants of time where the measure has lowered suddenly, then the result would be strongly inaccurate.*

A more sophisticated approach consists in using ambient intelligence *data. Again this configuration suffer from two drawbacks: the limited amount of data that can be collected during the decision process can be strongly influential on the computer solution, and reactivity to the environment behaviours can be quite useless in energy saving for a number of devices whose consumption curve is very inefficient if compressed in short intervals (for instance, electric heating is very inefficient if it is turned off and on frequently, therefore is a very bad candidate to the usage of methods based on presence sensors).*

Towards a more logical viewpoint support to the decision can be obtained by using defeasible logic, as shown in particular in [6, 29]. Starting from that concept, consisting in the usage of defeasible methods to represent different, possibly conflicting rules to save energy in home, home-office, small office contexts, we can apply the concept of linear defeasible logic to analogous contexts, enhancing

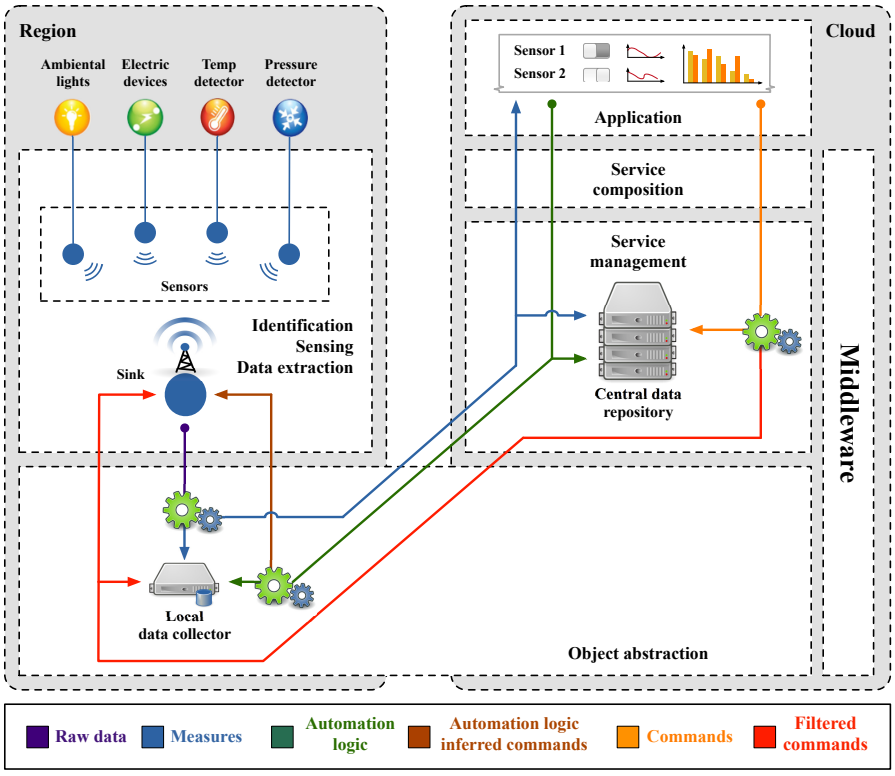

Fig. 1. The architecture of *MyElettra* an ambient intelligence technology for energy efficiency.

the representation with a novel correspondence.

Using LDL *We map into rules the behaviour that we desire for a compenent of an energy-absorbing system, that start how those confiurations produce the effects that we desire for them, and map into exceptions the* energy wastes. *In other terms, we assume that a rule* saves energy *in the sense that it achieves a specific goal of functionality of the system used to control, by using a certain amount of energy. An exception, while possibly stopping achievement of the goal (as in any common planning scenario with exceptions, as represented, for instance, in [11]) can be specifically use to represent the occasional possibility that the* energy consumption *needed for that particular functionality is higher than usual. That method will be able to prevent both drawbacks of the ambient intelligence and rule-based opposite approaches. In the first case, in fact, we have seen employed mainly statistical methods, whilst in the second approach the specification of native rules can provide a good accommodation for common sense rationalizations but no optimization issues can be applied.*

If we compute the least consuming scenario *but goal compliant, we obtain a system that reduces wastes, as modeled by the designer of the ambient intelligent system, a neat novelty in the context of energy saving.*

Scenario 2 ***Energy production waste control***
Analogously to the scenario of energy saving, we can represent the production of energy with linear defeasible rules, where the goal of a rule is twofold:

- *Representing the desired production;*
- *Providing room for the* disperded *energy, in the production process.*

Once we have specificed correctly the rules of such a production system, the optimization techniques that classically apply to plans (see, for instance, [25, 27, 3] for recent advancements) can generate perfect plans to execute. However, in this configurations, the practical cases should identify exceptional configurations in which it is not possible to forecast in a correct way the energy absorbed (or other numerical variables associated to the plan, such as delays of time, or money waste). The representation of this exception in an explicit way can be very useful to generate correct plans.

4.2 Modelling business processess

Scenario 3 ***Business process compliance by design or revision***
Long line of application of defeasible logic to business process compliance can be found in literature, mostly cited in Section 1. In particular we refer here to [17] where compliance is used in a planning context. What is precisely the business process compliance problem, and why should we consider applying Linear Defeasibly Logic to it?

A business process is said to be compliant when it achieves the goals it has been conceived for in a way that does not violate the norms in the normative background the business process has to be executed. For instance, a procedure to build a safety device in a car, say the safety belts, can be respectful of the norms of the country it is produced in while workers wear adequate shoes, the environment of the production is clean enough and so on. One of the most common problems with this issue is to provide algorithms that allow to build compliant business process while designing the process *and not* a posteriori*. In the second undesired scenario, in fact, there will be a risk of generating an uncompliant process, and therefore making it fail. Instead it will definitely make sense to provide* only *compliant processes. Methods used so far produce processes with the desired properties but do not take into consideration resource consumption.*

After having provided a configration issue as stated above, and in line to the discussion of Section 1, we may notice that the major problem in these configuration sites on the exception types *that can be introduced. In a pleminary study, related to MES applications (where the cost accounting is relevant) we have found that numerous exceptions have not been considered in the generation of business processes, and the combination of these have definite effects on the compliance itself. A partial yet incomplete list is below:*

- ***Safety exceptions*** *occur when a process need to be interrupted for an accident occured.*
- ***Stock break exceptions*** *occur when the feedstock of a machinery ends out.*
- ***Lack of energy*** *produces stops in the production process.*
- ***Strikes, protests, or other interruptions*** *produce stop in the process.*

Methods to reason about these exceptions cannot be safely based upon the usage of pure defeasible exceptions. It is rather obvious that all the above mentioned cases involve resource consumptions *and should therefore treated in the way presented in [19] and discussed here as an application issue.*

Business process revisions have been studied widely. A very specific study we refer to here is [26] where revision is studied in terms of preserving compliance when changing goals or normative background. Analogous to what has been provided above can occur when one configuration has been chosen and we aim at revising the choice for improvements.

5 Related work

Some of the authors have already exploit an effort in the direction of applying defeasible logic to energy management, in [6, 29]. This complemented the investigations on applying logic and machine learning to energy management [5, 4, 30]. Analogously some authors applied defeasible logic to business processes [10, 11] following the long line of investigations cited in the introduction [14, 28, 8, 24, 20, 18, 9].

The novelty introduced in [19] and discussed in this paper has no precedent for business processes for the combination itself being new as a whole. Conversely, usage of linear logic to model business processes is new for itself. There have been some investigations trying to determine applicability of linear logic to planning [] and to the modelling of complex processes, for Petri nets (see [23] for a relatively recent review, and [7] for more recent advacements).

6 Conclusions and related work

We discussed the applications of a recently introduced logical apparatus that deals with the problem of manipulating resource consumption in non-monotonic reasoning. We illustrated in particular applications to energy management and business processes.

Applications of linear logic to problems indirectly related to business processes such as Petri Nets can be found in [14, 28, 8]. However, such approaches are not able to handle in a natural fashion the aspect of exceptions. In [15, 21, 22], the authors propose the use of Linear Logic to generate which plans the agent adopts to achieve its goals. In the same spirit, [12] address the problem of agents being able to take decisions from partial, incomplete, and possibly inconsistent knowledge bases.

References

1. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Trans. Comput. Log. 2(2), 255–287 (2001)
2. Antoniou, G., Billington, D., Governatori, G., Maher, M.J., Rock, A.: A family of defeasible reasoning logics and its implementation. In: Horn, W. (ed.) ECAI. pp. 459–463. IOS Press (2000)
3. Burggraeve, S., Bull, S., Vansteenwegen, P., Lusby, R.: Integrating robust timetabling in line plan optimization for railway systems. Transportation Research Part C: Emerging Technologies 77, 134–160 (2017)
4. Cristani, M., Karafili, E., Tomazzoli, C.: Energy saving by ambient intelligence techniques. pp. 157–164 (2014)
5. Cristani, M., Karafili, E., Tomazzoli, C.: Improving energy saving techniques by ambient intelligence scheduling. vol. 2015-April, pp. 324–331 (2015)
6. Cristani, M., Tomazzoli, C., Karafili, E., Olivieri, F.: Defeasible reasoning about electric consumptions. vol. 2016-May, pp. 885–892 (2016)
7. DemeterovÃą, E., MihÃąlyi, D., NovitzkÃą, V.: Component composition using linear logic and petri nets. pp. 91–96 (2016)
8. Engberg, U., Winskel, G.: Completeness results for linear logic on petri nets. Ann. Pure Appl. Logic 86(2), 101–135 (1997)
9. Ghooshchi, N.G., van Beest, N., Governatori, G., Olivieri, F., Sattar, A.: Visualisation of compliant declarative business processes. In: EDOC 2017. pp. 89–94. IEEE Computer Society (2017)
10. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S., Cristani, M.: Picking up the best goal an analytical study in defeasible logic. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8035, 99–113 (2013)
11. Governatori, G., Olivieri, F., Scannapieco, S., Cristani, M.: Designing for compliance: Norms and goals. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7018 LNCS, 282–297 (2011)
12. Governatori, G., Olivieri, F., Scannapieco, S., Rotolo, A., Cristani, M.: The rational behind the concept of goal. Theory and Practice of Logic Programming 16(3), 296–324 (2016)
13. Governatori, G., Rotolo, A.: BIO logical agents: Norms, beliefs, intentions in defeasible logic. Journal of Autonomous Agents and Multi Agent Systems 17(1), 36–69 (2008)
14. Kanovich, M., Ito, T.: Temporal linear logic specs for concurrent processes. In: LICS 1997. pp. 48–57. IEEE Computer Society (1997)
15. Küngas, P., Matskin, M.: Linear logic, partial deduction and cooperative problem solving. In: DALT II. pp. 263–279. LNCS 3476, Springer (2004)
16. Nute, D.: Defeasible logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3. Oxford University Press (1987)
17. Olivieri, F., Cristani, M., Governatori, G.: Compliant business processes with exclusive choices from agent specification. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9387, 603–612 (2015)
18. Olivieri, F., Cristani, M., Governatori, G.: Compliant business processes with exclusive choices from agent specification. In: PRIMA 2015. pp. 603–612. LNCS 9387, Springer (2015)

19. Olivieri, F., Governatori, G., Cristani, M., van Beest, N., Tosatto, S.C.: Resource-driven substructural defeasible logic. LNCS, vol. 11224, pp. 594–602. Springer (2018)
20. Olivieri, F., Governatori, G., Scannapieco, S., Cristani, M.: Compliant business process design by declarative specifications. In: PRIMA 2013. pp. 213–228. LNCS 8291, Springer (2013)
21. Pham, D.Q., Harland, J.: Temporal linear logic as a basis for flexible agent interactions. In: AAMAS '07. pp. 28:1–28:8. ACM (2007)
22. Pham, D.Q., Harland, J., Winikoff, M.: Modeling agents' choices in temporal linear logic. In: DALT V. pp. 140–157. LNCS 5397, Springer (2008)
23. Pradin-ChÃľzalviel, B., Valette, R., RiviÃĺre, N.: Petri Nets and Linear Logic (2010)
24. Rao, J., Küngas, P., Matskin, M.: Composition of semantic web services using linear logic theorem proving. Information Systems 31(4-5), 340–360 (2006)
25. Rygielski, P., ÅŻwiaİğtek, P.: Graph-fold: An efficient method for complex service execution plan optimization. Systems Science 36(3), 25–32 (2010)
26. Scannapieco, S., Governatori, G., Olivieri, F., Cristani, M.: A methodology for plan revision under norm and outcome compliance. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8291 LNAI, 324–339 (2013)
27. Tambe, P., Kulkarni, M.: A superimposition based approach for maintenance and quality plan optimization with production schedule, availability, repair time and detection time constraints for a single machine. Journal of Manufacturing Systems 37, 17–32 (2015)
28. Tanabe, M.: Timed petri nets and temporal linear logic. In: ICATPN 1997. pp. 156–174. LNCS 1248 (1997)
29. Tomazzoli, C., Cristani, M., Karafili, E., Olivieri, F.: Non-monotonic reasoning rules for energy efficiency. Journal of Ambient Intelligence and Smart Environments 9(3), 345–360 (2017)
30. Tomazzoli, C., Cristani, M., Scannapieco, S., Olivieri, F.: Automatic detection of device types by consumption curve. Smart Innovation, Systems and Technologies 96, 164–174 (2018)