Integration of Web-based Forms with Ontologies in the Semantic Web

Sergio A. Gómez†, Carlos I. Chesñevar†‡, Guillermo R. Simari†

†Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)*
Depto. Cs. e Ing. de la Computación – Universidad Nacional del Sur
Alem 1253 (8000) Bahía Blanca - ARGENTINA –
Tel/Fax: (+54) 291-459 5135/5136 – E-mail: {sag, cic, grs}@cs.uns.edu.ar
‡CONICET (Consejo Nacional de Investigaciones Científicas y Técnicas), Argentina

Abstract

The notion of forms as a way of organizing and presenting data has been used since the beginning of the World Wide Web. Web-based forms have evolved together with the development of new markup languages, in which it is possible to provide validation scripts as part of the form code to test whether the intended meaning of the form is correct. However, for the form designer, part of this intended meaning frequently involves other features which are not constraints by themselves, but rather attributes emerging from the form, which provide plausible conclusions in the context of incomplete and potentially inconsistent information. As the value of such attributes may change in presence of new knowledge, we call them defeasible attributes. In previous works, we extended traditional web-based forms to incorporate defeasible attributes as part of the knowledge that can be encoded by the form designer. In this article, we recast that approach to make it suitable for the Semantic Web initiative; we then propose the specification of defeasible attributes by means of possibly inconsistent Description Logics ontologies known as δ-ontologies. Thus the value of a defeasible attribute will be associated to the membership of an individual to a certain concept. As the ontologies involved in the definition of defeasible attributes may be inconsistent, a dialectical analysis will be performed to take into account all the reasons in favor and against the value of such defeasible attributes.

Keywords: Description Logics, Defeasible Logic Programming, web forms, defeasible argumentation, ontologies, Semantic Web

Resumen

La noción de formularios como una manera de organizar y presentar datos ha sido usada desde el comienzo de la World Wide Web. Los formularios web han evolucionado junto con el desarrollo de nuevos lenguajes de marcado, en los cuales es posible proveer guiones de validación como parte del código del formulario para verificar si el significado pretendido del formulario es correcto. Sin embargo, para el diseñador del formulario, parte de este significado pretendido involucra

* LIDIA is a member of II CyTI (Instituto de Investigación en Ciencia y Tecnología Informática).
otros atributos que no son restricciones por sí mismas, sino más bien atributos emergentes del formulario, los cuales brindan conclusiones plausibles en el contexto de información incompleta y potencialmente contradictoria. Como el valor de tales atributos puede cambiar en presencia de nuevo conocimiento, los llamamos atributos rebatibles. En trabajos previos, extendimos los formularios web tradicionales para incorporar atributos rebatibles como parte del conocimiento que puede ser codificado por el disenador del formulario. En este artículo, adaptamos tal acercamiento para hacerlo adecuado para la iniciativa de la Web Semántica; entonces proponemos la especificación de atributos rebatibles por medio de ontologías expresadas en Lógicas para la Descripción posiblemente inconsistentes conocidas como δ-ontologías. El valor de tales atributos rebatibles será asociado a la pertenencia de un individuo a un concepto particular. Como las ontologías involucradas en la definición de atributos rebatibles pueden ser inconsistentes, un análisis dialéctico será realizado para tomar en cuenta todas las razones a favor y en contra del valor del atributo rebatable.

**Palabras clave:** Lógicas para la Descripción, Programación en Lógica Rebatible, formularios web, argumentación rebatable, ontologías, Web Semántica

### 1 Introduction

The notion of form as a way of organizing and presenting data is a well-known structural abstraction for data collection, storage, and information retrieval. Forms are an important means for designing and developing user-oriented information systems, and have long been used since the very beginning of the World Wide Web. Web-based forms have evolved together with the development of new markup languages (e.g., XML), in which it is possible to provide validation scripts as part of the form code in order to test whether the intensional meaning of the form is correct [14].

The **Semantic Web** [2] is a future vision of the web where stored information has exact meaning, thus enabling computers to understand and reason on the basis of such information. Assigning semantics to web resources is addressed by means of *ontology definitions* which are meant to be written in an *ontology description language*. Fulfilling the goals of the Semantic Web program requires having tools capable of dealing with the potential inconsistencies and incompleteness of web data sources. One particularly important application domain is e-commerce technologies, which typically demand validation of user data (e.g., credit card numbers) against a set of criteria for determining if a given user is eligible for certain prospective transaction. Performing validations on field values allows to determine whether the intended meaning of such fields is coherent according to some criteria established by the form designer. Such validations usually consist of a number of hard-coded decision criteria as a portion of imperative code in a script language. However, in many cases there are some emerging features which can be inferred as part of the “intended meaning” of the form without being field values themselves. Thus, in the case of a bank loan application, the notion of “reliable client” may be inferred as plausible from knowing the annual income and banking records of a particular customer. Such features (or *attributes*) of the form are difficult to model in terms of pieces of imperative code, particularly in presence of incomplete and potentially contradictory information. The associated conclusions turn out to be defeasible, as they may change in the light of new information.

Standard ontology description languages such as OWL-DL are based on *Description Logics* (DL) [1]. Although ontology definitions expressed in DLs can be processed with existing DL reasoners, such DL reasoners are incapable of dealing with inconsistent ontology definitions. This situation is particularly important in the Semantic Web setting, where ontologies are complex entities prone to suffer inconsistencies [10]. A particular source of inconsistency is related to the use of imported ontologies
when the knowledge engineer has no authority to correct them. As these imported ontologies are usu-
ally developed independently, their combination could also result in inconsistencies. The problem of
combining two or more different ontologies in order to obtain a unified, consistent ontology is known
as ontology merging [11].

There are two main ways to deal with inconsistency in ontologies [10]: one is to diagnose and
repair it when it is encountered; another is to avoid the inconsistency and to apply a non-standard
inference relation to obtain meaningful answers. In previous works [6, 7] we proposed to use defeasible argumentation [3] to focus on the latter by means of using δ-ontologies, a particular kind
of DL ontologies amenable for its treatment into Defeasible Logic Programming (DeLP) [5]. DeLP
is an argumentative framework based on logic programming that is capable of dealing with possibly
inconsistent knowledge bases codified as a set of Horn-like clauses called DeLP programs. When
presented with a query, DeLP performs a dialectical process in which all arguments in favor and
against a conclusion are considered; arguments regarded as ultimately undefeated will be considered
warranted.

In this paper we propose extending traditional web-based forms to incorporate defeasible attributes
as part of the knowledge that can be encoded by the form designer. The proposed extension allows
the specification of defeasible attributes by means of possibly inconsistent Description Logics ontolo-
gies known as δ-ontologies to make them suitable for the Semantic Web initiative. The value of a
defeasible attribute will be associated to the membership of an individual to a certain concept which
represents a defeasible attribute. As the ontologies involved in the definition of defeasible attributes
may be inconsistent, a dialectical analysis will be performed to take into account all the reasons sup-
porting and against the value of a defeasible attribute.

The rest of the article is structured as follows. Section 2 introduces the fundamentals of how
Description Logics ontologies are processed into Defeasible Logic Programming when the involved
ontologies are inconsistent. Section 3 presents the integration of web forms with DL ontologies.
Finally Section 4 concludes.

2 Reasoning with inconsistent Description Logics ontologies into Defeasible
Logic Programming

2.1 Description Logics

Description Logics (DL) are a well-known family of knowledge representation formalisms [1]. They
are based on the notions of concepts (unary predicates, classes) and roles (binary relations), and are
mainly characterized by constructors that allow complex concepts and roles to be built from atomic
ones. The expressive power of a DL system is determined by the constructs available for building
concept descriptions, and by the way these descriptions can be used in the terminological (Tbox) and
assertional (Abox) components of the system.

We now describe the basic language for building DL expressions. Let C and D stand for concepts
and R for a role name. Concept descriptions are built from concept names using the constructors
conjunction (C ⊓ D), disjunction (C ⊔ D), negation (¬C), existential restriction (∃R.C), and value
restriction (∀R.C). To define the semantics of concept descriptions, concepts are interpreted as sub-
sets of a domain of interest, and roles as binary relations over this domain. An interpretation I consists
of a non-empty set ΔI (the domain of I) and a function ·I (the interpretation function of I) which maps
every concept name A to a subset AI of ΔI, and every role name R to a subset RI of ΔI × ΔI. The
interpretation function is extended to arbitrary concept descriptions, concepts are interpreted as sub-
sets of a domain of interest, and roles as binary relations over this domain. An interpretation I consists
of a non-empty set ΔI (the domain of I) and a function ·I (the interpretation function of I) which maps
every concept name A to a subset AI of ΔI, and every role name R to a subset RI of ΔI × ΔI. The
interpretation function is extended to arbitrary concept descriptions as follows: (¬C)I = ΔI\CI; (C ⊓ D)I = CI ∩ DI; (C ⊔ D)I = CI ∪ DI; (∃R.C)I = \{x|∃y s.t. (x, y) ∈ RI and y ∈ CI\}, and
\[(\forall R.C)^I = \{x \mid y, (x, y) \in R^I \text{ implies } y \in C^I\}\]. Besides, the expressions \(\top\) and \(\bot\) are shorthands for \(C \sqcup \neg C\) and \(C \sqcap \neg C\), resp. Further extensions to the basic DL are possible including inverse and transitive roles noted as \(P^-\) and \(P^+\), resp.

A traditional DL ontology consists of two finite and mutually disjoint sets: a Tbox which introduces the terminology and an Abox which contains facts about particular objects in the application domain. Tbox statements have the form \(C \sqsubseteq D\) (inclusions) and \(C \equiv D\) (equalities), where \(C\) and \(D\) are (possibly complex) concept descriptions. The semantics of Tbox statements is as follows. An interpretation \(I\) satisfies \(C \sqsubseteq D\) iff \(C^I \subseteq D^I\), \(I\) satisfies \(C \equiv D\) iff \(C^I = D^I\). Objects in the Abox are referred to by a finite number of individual names and these names may be used in two types of assertional statements: concept assertions of the type \(C(a)\) and role assertions of the type \(R(a, b)\), where \(C\) is a concept description, \(R\) is a role name, and \(a\) and \(b\) are individual names. An interpretation \(I\) satisfies the assertion \(C(a)\) iff \(a^I \in C^I\), and it satisfies \(R(a, b)\) iff \((a^I, b^I) \in R^I\). An interpretation \(I\) is a model of a DL (Tbox or Abox) statement \(\phi\) iff it satisfies the statement, and is a model of an ontology \(\Sigma\) iff it satisfies every statement in \(\Sigma\). A DL ontology \(\Sigma\) entails a DL statement \(\phi\), written as \(\Sigma \models \phi\), iff every model of \(\Sigma\) is a model of \(\phi\).

### 2.2 Defeasible Logic Programming

Defeasible Logic Programming (DeLP) [5] provides a language for knowledge representation and reasoning that uses defeasible argumentation [3, 13] to decide between contradictory conclusions through a dialectical analysis. Codifying knowledge by means of a DeLP program provides a good trade-off between expressivity and implementability for dealing with incomplete and potentially contradictory information. In a defeasible logic program \(\mathcal{P} = (\Pi, \Delta)\), a set \(\Delta\) of defeasible rules \(P \leftarrow Q_1, \ldots, Q_n\), and a set \(\Pi\) of strict rules \(P \leftarrow Q_1, \ldots, Q_n\) can be distinguished. An argument \((A, H)\) is a minimal non-contradictory set of ground defeasible clauses \(A\) of \(\Delta\) that allows to derive a ground literal \(H\) possibly using ground rules of \(\Pi\). Since arguments may be in conflict (concept captured in terms of a logical contradiction), an attack relationship between arguments can be defined. A criterion is usually defined to decide between two conflicting arguments. If the attacking argument is strictly preferred over the attacked one, then it is called a proper defeater. If no comparison is possible, or both arguments are equi-preferred, the attacking argument is called a blocking defeater. In order to determine whether a given argument \(A\) is ultimately undefeated (or warranted), a dialectical process is recursively carried out, where defeaters for \(A\), defeaters for these defeaters, and so on, are taken into account. Given a DeLP program \(\mathcal{P}\) and a query \(H\), the final answer to \(H\) w.r.t. \(\mathcal{P}\) takes such dialectical analysis into account. The answer to a query can be: yes, no, undecided, or unknown. For an in-depth treatment of DeLP, see [5].

### 2.3 Interpreting DL ontologies into DeLP for solving inconsistencies

In the presence of inconsistent ontologies, traditional DL reasoners (such as RACER [9]) issue an error message and stop further processing. Thus the burden of repairing the ontology (i.e., making it consistent) is on the knowledge engineer. In previous works [6, 7], we showed that DeLP can be used for coping with inconsistencies in ontologies such that the task of dealing with them is automatically solved by the reasoning system. For doing this, the DL under consideration must respect certain constraints; we extend the definition given in [7] for considering DLs with concrete domains [12]:

**Definition 1 (\(\mathcal{L}_b\) language/classes \(\mathcal{L}_b\) language/classes. \(\mathcal{L}_bb\) language/classes (adapted from [7]))**

Let \(A\) be an atomic class name, \(C\) and \(D\) class expressions, \(R\) a property, and \(f\) a concrete attribute. Let \(\mathbb{N}\) be the set of natural numbers. Let \(n \in \mathbb{N}, op \in \{=, \neq, >, \geq, <, \leq\}\). In the \(\mathcal{L}_h\) language,
$C \cap D$ is a class, and $\forall R.C$ is also a class. Class expressions in $\mathcal{L}_h$ are called $\mathcal{L}_h$-classes. In the $\mathcal{L}_b$ language, $C \cap D$, $C \cup D$, $\exists R.C$ and $\exists R.\text{op}_a$ are classes. Class expressions in $\mathcal{L}_b$ are called $\mathcal{L}_b$-classes. The $\mathcal{L}_{hb}$ language is defined as the intersection of $\mathcal{L}_h$ and $\mathcal{L}_b$. Class expressions in $\mathcal{L}_{hb}$ are called $\mathcal{L}_{hb}$-classes.

In [7], we presented a mapping from DL to DeLP used for translating DL ontologies into DeLP programs. A function $\mathcal{T}_\Pi$ for translating DL ontologies into DeLP strict rules was presented along with a function $\mathcal{T}_\Delta$ for translating DL ontologies into defeasible rules. The proposal for extending forms presented in [8] requires the capability of representing concrete domains in ontologies; thus in Figure 1, we now extend such mapping for dealing with the concrete domain of natural numbers.

$$
\begin{align*}
\mathcal{T}_b((\exists f. = n)) &= \text{def} \quad f = n \\
\mathcal{T}_b((\exists f. \neq n)) &= \text{def} \quad f \neq n \\
\mathcal{T}_b((\exists f. > n)) &= \text{def} \quad f > n \\
\mathcal{T}_b((\exists f. \geq n)) &= \text{def} \quad f \geq n \\
\mathcal{T}_b((\exists f. < n)) &= \text{def} \quad f < n \\
\mathcal{T}_b((\exists f. \leq n)) &= \text{def} \quad f \leq n
\end{align*}
$$

Figure 1: Extension to mappings $\mathcal{T}_\Pi$ and $\mathcal{T}_\Delta$ for dealing with concrete domains

An ontology is defined as a set of classes and a set of individuals belonging to such classes. In [7], we redefined the notion of DL ontology for making it suitable for its treatment within DeLP.

**Definition 2** ($\delta$-Ontology [7])  Let $C$ be an $\mathcal{L}_b$-class, $D$ an $\mathcal{L}_h$-class, $A, B$ $\mathcal{L}_{hb}$-classes, $P, Q$ properties, $a, b$ individuals. Let $T$ be a set of inclusion and equality sentences in $\mathcal{L}_DL$ of the form $C \subseteq D$, $A \equiv B$, $\top \sqsubseteq \forall P.D$, $\top \sqsubseteq \forall P^-.D$, $P \sqsubseteq Q$, $P \equiv Q$, $P \equiv Q^-$, or $P^+ \sqsubseteq P$ such that $T$ can be partitioned into two disjoint sets $T_S$ and $T_D$. Let $A$ be a set of assertions disjoint with $T$ of the form $D(a)$ or $P(a, b)$. A $\delta$-ontology $\Sigma$ is a tuple $(T_S, T_D, A)$. The set $T_S$ is called the strict terminology (or Sbox), $T_D$ the defeasible terminology (or Dbox) and $A$ the assertional box (or Abox).

**Example 1** An international bank keeps track of its clients in order to determine whether to concede loans. Consider an ontology $\Sigma_{\text{criteria}} = (T_S^{\text{criteria}}, T_D^{\text{criteria}}, \emptyset)$ defining criteria for granting loans in a bank as presented in Figure 2. Sentence (1) in strict terminology $T_S^{\text{criteria}}$ expresses that all millionaires are candidate to loans. Defeasible terminology $T_D^{\text{criteria}}$ is composed by sentences (2) to (9). Sentence (2) establishes that usually a person with a right profile is candidate for receiving a loan. Sentence (3) says that a customer with high income (greater or equal to $1000$) and that asked for a reasonable loan (less than $300$) is usually a candidate for receiving a loan. Sentence (4) expresses that a customer with a good income and coming from a reliable country typically have a right profile. Sentence (5) says that customers with an income greater than 300 dollars are normally considered as having a good income. Sentence (6) establishes that non-solvent people usually do not have a good income. Sentence (7) says that a customer whose profession is university student usually is insolvent unless (as expressed by sentence (8)) she comes from a wealthy family. Sentence (9) says that if a customer is rich according to the bank’s records, then she is considered as coming from a wealthy family.

Consider also the $\delta$-ontology $\Sigma_{HSO} = (\emptyset, T_D^{HSO}, A^{HSO})$ presented in Figure 3 providing risk information about countries. The information defined in Dbox $T_D^{HSO}$ says that usually democratic
countries are trustworthy (sentence (1)) unless they are at war (sentence (2)), and democratic governments but with corrupt governments are not trustworthy (sentence (3)). Assertional box $A^{HSO}$ establishes that Greece and Krakosia are countries (assertions (4) and (5)), Greece is a democracy (assertion (6)), and Krakosia is democracy at war (assertions (7) and (8)).

Consider the δ-ontology $\Sigma^{bank} = (\emptyset, T^{bank}_{D}, A^{bank})$ with bank provided client information presented in Figure 4. Dbox $T^{bank}_{D}$ expresses that medical doctors, professors, lawyers, engineers, MSc, PhD and undergraduate students, and unemployed are professions (sentence (1)); rich, broken and none are client’s family status (sentence (2)); rich is the status of a rich family (sentence (3)); MSc, PhD and undergraduate students are university students (sentence (4)). Sentence (5) says that university student is a profession. Besides, attribute family record has family status as range and a customer as domain (sentences (6) and (7)); attribute customer name has a name as range and a customer as domain (sentences (8) and (9)); attribute customer has a profession as range and a customer as domain (sentences (10) and (11)); attribute country has a country as range and a customer as domain (sentences (12) and (13)); attribute required loan has a number as range and a customer as domain. Sentence (18) says that Ajax, Danae, John and Peter are names. The assertional box $A^{bank}$ says that individual $c_1$ is rich, family records about individuals $c_2$ and $c_3$ say that the bank has no information about them, and that $c_4$ is a millionaire (sentences (19)–(22)).

In Figure 5, the assertional box $A^{user}$ says that there are four customers $c_1$, $c_2$, $c_3$ and $c_4$ whose names are John, Ajax, Danae, and Peter, resp.; their incomes are $400, $350 and $1000 (notice that there is no income information for Peter); their countries are Krakosia and Greece; John and Ajax’s profession is PhDStudent, Danae’s is none and there is no information about Peter’s profession; John requested a loan of $2000, Ajax $4500, Danae $1000 and Peter $1000.

---

**Strict terminology (Sbox) $T^{criteria}_{S}$:**

1. millionaire $\sqsubseteq$ candidate

**Defeasible terminology (Dbox) $T^{criteria}_{D}$:**

2. profile_ok $\sqsubseteq$ candidate
3. $\exists$ cstmr_income. $\geq$ 1000 $\sqcap$ $\exists$ req_loan. $<$ 300 $\sqsubseteq$ candidate
4. good_income $\sqcap$ $\exists$ cstmr_country. credible $\sqsubseteq$ profile_ok
5. $\exists$ cstmr_income. $>$ 300 $\sqsubseteq$ good_income
6. $\neg$ solvent $\sqsubseteq$ $\neg$ good_income
7. customer $\sqcap$ $\exists$ cstmr_profession. univ_student $\sqsubseteq$ $\neg$ solvent
8. customer $\sqcap$ $\exists$ cstmr_profession. univ_student $\sqcap$ rich_family $\sqsubseteq$ solvent
9. $\exists$ family_record. rich_status $\sqsubseteq$ rich_family

Figure 2: Ontology $\Sigma^{criteria} = (T^{criteria}_{S}, T^{criteria}_{D}, \emptyset)$ with a set of criteria for granting loans to customers

As DLs are a subset of first-order logic (FOL), entailment has an explosive effect in the presence of inconsistent ontologies. In [7], we proposed an argumentative approach to reasoning with inconsistent ontologies. Thus a δ-ontology is interpreted as a DeLP program; intuitively, the Sbox is expressed as a set of strict rules, the Dbox is expressed as a set of defeasible rules, and the Abox is expressed as a set of facts.

**Definition 3 (Interpretation of a δ-ontology [7])** Let $\Sigma = (T_S, T_D, A)$ be a δ-ontology. If $T_S \cup A$ has a model, the interpretation of $\Sigma$ is a DeLP program $\mathcal{P} = (T_H(T_S) \cup T_H(A), T_\Delta(T_D))$. 
The following dialectical analysis arises from the interpretation of $\Sigma_{merge}$. For instance, for checking if John is a candidate to a loan (i.e., checking for the value of the attribute “candidate”), we find
to the concept candidate as there exists an argument \( \langle A_1, \text{candidate}(c_1) \rangle \) where:
\[
A_1 = \begin{cases} 
\text{candidate}(c_1) \leftarrow \text{profile}\_\text{ok}(c_1) \\
\text{profile}\_\text{ok}(c_1) \leftarrow \text{good}\_\text{income}(c_1), \text{cstmr}\_\text{country}(c_1, \text{krakosia}), \text{credible}(\text{krakosia}) \\
\text{credible}(\text{krakosia}) \leftarrow \text{country}(\text{krakosia}), \text{democracy}(\text{krakosia}) \\
\text{good}\_\text{income}(c_1) \leftarrow \text{cstmr}\_\text{income}(c_1, 400), 400 \geq 300 \end{cases}
\]
but this is defeated by another argument \( \langle A_2, \sim \text{credible}(\text{krakosia}) \rangle \) saying that Krakosia is not trustworthy because it is a country at war:
\[
A_2 = \{ \sim\text{credible}(\text{krakosia}) \leftarrow \text{country}(\text{krakosia}), \text{democracy}(\text{krakosia}), \text{atwar}(\text{krakosia}) \}
\]
In the case of Ajax, the value of the attribute “candidate” is undecided as we cannot determine that the individual \( c_2 \) potentially belongs to the concept “candidate”. There is an argument \( \langle B_1, \text{candidate}(c_2) \rangle \) (so \( c_2 \) potentially belongs to the concept “candidate”), with
\[
B_1 = \begin{cases} 
\text{candidate}(c_2) \leftarrow \text{profile}\_\text{ok}(c_2) \\
\text{profile}\_\text{ok}(c_2) \leftarrow \text{good}\_\text{income}(c_2), \text{cstmr}\_\text{country}(c_2, \text{greece}), \text{credible}(\text{greece}) \\
\text{credible}(\text{greece}) \leftarrow \text{country}(\text{greece}), \text{democracy}(\text{greece}) \\
\text{good}\_\text{income}(c_1) \leftarrow \text{cstmr}\_\text{income}(c_1, 350), 350 \geq 300 \end{cases}
\]
but this is defeated by another argument \( \langle B_2, \sim \text{good}\_\text{income}(c_2) \rangle \) with:
\[
B_2 = \begin{cases} 
\sim\text{good}\_\text{income}(c_2) \leftarrow \sim\text{solvent}(c_2) \\
\sim\text{solvent}(c_2) \leftarrow \text{customer}(c_2), \text{cstmr}\_\text{profession}(c_2, \text{phd}\_\text{student}), \text{univ}\_\text{student}(\text{phd}\_\text{student}) \\
\text{univ}\_\text{student}(\text{phd}\_\text{student}) \leftarrow \text{phd}\_\text{student} = \text{phd}\_\text{student} \end{cases}
\]
But in the case of Danae, the attribute “candidate” takes the value true as the individual \( c_3 \) justifies belongs to the concept “candidate” because there is an undefeated (and thus warranted) argument \( \langle C_1, \text{candidate}(c_3) \rangle \) where:
\[
B_1 = \begin{cases} 
\text{candidate}(c_3) \leftarrow \text{profile}\_\text{ok}(c_3) \\
\text{profile}\_\text{ok}(c_3) \leftarrow \text{good}\_\text{income}(c_3), \text{cstmr}\_\text{country}(c_3, \text{greece}), \text{credible}(\text{greece}) \\
\text{credible}(\text{greece}) \leftarrow \text{country}(\text{greece}), \text{democracy}(\text{greece}) \\
\text{good}\_\text{income}(c_1) \leftarrow \text{cstmr}\_\text{income}(c_3, 1000), 1000 \geq 300 \end{cases}
\]
On the other hand, Peter is also a candidate as he is a millionaire and there are no defeaters for that argument, so individual \( c_4 \) strictly belongs to the concept “candidate”.

---

**Figure 5:** User provided assertional information \( A^{\text{user}} \)
in the previous sections, a concept like reliable client being field values themselves. Thus, in the case of a bank loan application such as the one discussed some emerging features which can be inferred as part of the “intended meaning” of the form without data is finally processed by a program located in a remote server. However, in many cases there are technologies as well as the envisioning of the Semantic Web motivated the specification of sophisticated applications which outperformed static web pages. The growing popularity of e-commerce technologies as well as the envisioning of the Semantic Web motivated the specification of sophisticated standards for web-based forms, notably XForms [4].

In spite of the evolution of web-based form technologies, most form designers perform validation of form fields by enforcing constraints (e.g., numeric ranges) encoded as pieces of imperative code in a scripting language (e.g., JavaScript). Thus, validation of data is done client-side, and the form data is finally processed by a program located in a remote server. However, in many cases there are some emerging features which can be inferred as part of the “intended meaning” of the form without being field values themselves. Thus, in the case of a bank loan application such as the one discussed in the previous sections, a concept like reliable client, modeled on the basis of the field values for a particular customer, could prove useful for the form designer in order to codify decision making issues associated with form processing. To identify every relevant attributes needed to infer a concept like “reliable customer” using only imperative code may be a difficult task, as in complex situations such conclusions are defeasible (particularly in presence of incomplete and potentially inconsistent information).

To address the above problem, the concept of form can be suitably extended to formalize such complex scenarios on the basis of DeLP by means of defeasible attributes. In order to do this, we will first provide a rather generic definition of the concept of form on the basis of the notions of form schema and form instance based on a previous work of ours [8] but adapted to a DL setting.

Definition 5 (Form schema. Form instance) A form schema is a triple $\mathcal{F} = \langle F, T, C \rangle$, where $F = \{f_1, \ldots, f_n\}$ is a list of form fields, $T = \{T_1, \ldots, T_n\}$ is a list of types or concepts, and $C$ is a concept including the individual filling the form. If $V = \{v_1, \ldots, v_n\}$ is a list of values such that $v_i \in T_i$, $i = 1, \ldots, n$ is the value for $f_i \in F$ for individual $c_j$, a form instance based on $\mathcal{F}$ with value $V$ and individual identifier $c_j$ (noted as $\mathcal{F}^c_{Vj}$) is a triple $\mathcal{F}^c_{Vj} = \langle F, V, c_j \rangle$.
Defeasible rules $\Delta^\text{bank}$:

1. profession(X) $\rightarrow$ X = medical_doctor.
2. profession(Y) $\rightarrow$ X = professor.
3. profession(Z) $\rightarrow$ X = lawyer.
4. profession(W) $\rightarrow$ X = engineer.
5. profession(V) $\rightarrow$ X = msc_student.
6. profession(U) $\rightarrow$ X = phd_student.
7. profession(T) $\rightarrow$ X = undergrad_student.
8. customer(Y) $\rightarrow$ cstmr_name(X, Y).
9. customer(Y) $\rightarrow$ cstmr_profession(X, Y).
10. profession(Y) $\rightarrow$ cstmr_profession(X, Y).
11. customer(Y) $\rightarrow$ cstmr_profession(X, Y).
12. country(Y) $\rightarrow$ cstmr_country(X, Y).
13. customer(Y) $\rightarrow$ cstmr_country(X, Y).
14. number(Y) $\rightarrow$ cstmr_income(X, Y).
15. customer(Y) $\rightarrow$ cstmr_income(X, Y).
16. number(Y) $\rightarrow$ req_loan(X, Y).
17. customer(Y) $\rightarrow$ req_loan(X, Y).
18.a) name(X) $\rightarrow$ X = ajaz.
18.b) name(X) $\rightarrow$ X = danae.
18.c) name(X) $\rightarrow$ X = john.
18.d) name(X) $\rightarrow$ X = peter.
19. family_record(c1, rich)
20. family_record(c2, unknown)
21. family_record(c3, unknown)
22. millionaire(c4)

\[ \text{Figure 7: Ontology } \Sigma^\text{bank} \text{ expressed as a DeLP program } \mathcal{P}^\text{bank} = (\Pi^\text{bank}, \Delta^\text{bank}) \]

Defeasible rules $\Delta^\text{HSO}$:

1. credible(X) $\rightarrow$ country(X), democracy(X).
2. $\neg$credible(X) $\rightarrow$ country(X), democracy(X), atwar(X).
3. $\neg$credible(X) $\rightarrow$ country(X), democracy(X), corruptgov(X).
4. country(krakosia)
5. country(greece)
6. democracy(greece)
7. democracy(krakosia)
8. atwar(krakosia)

\[ \text{Figure 8: Ontology } \Sigma^\text{HSO} \text{ expressed as a DeLP program } \mathcal{P}^\text{HSO} = (\Pi^\text{HSO}, \Delta^\text{HSO}) \]

Example 3 Let $F$ be a list of form fields and $T$ a list of values such that:

\[ F = [\text{cstmr_name}, \text{cstmr_profession}, \text{cstmr_income}, \text{req_loan}, \text{cstmr_country}] \]
\[ T = [\text{name}, \text{profession}, \text{N}, \text{N}, \text{country}] \]

then $F = \langle F, T, \text{customer} \rangle$ is a form schema. Let $c_1$ be an individual identifier, let $V$ be a list of values such that: $V = [\text{john}, \text{phd_student}, 400, 2000, \text{krakosia}]$, then $F^V_{c_1} = \langle F, V, c_1 \rangle$ is a form instance based on $F$.

Figure 9 shows the typical graphical appearance of a web-based form according to the form schema given in Example 3. In [8] we showed how to extend traditional web-based forms to incorporate defeasible knowledge expressed in terms of a DeLP program, characterizing the notion of forms with defeasible attributes or $\delta$-forms. In this article, our goal is to provide a way to associate a $\delta$-ontology $\Sigma$ with an arbitrary form schema (which will correspond to a number of different possible form instances). The $\delta$-ontology $\Sigma$ is assumed to represent declarative knowledge associated with the problem domain of the form schema. Thus, as discussed in the previous example, a form schema corresponding to a bank application could have an associated ontology which represents tentative (and possibly conflicting) policies for granting loans.

Given a form schema $F = \langle F, T, C \rangle$ and a particular form instance $F^V_{c_1}$ we will capture the factual knowledge involved in $F^V_{c_1}$ in terms of an Abox $A^{F^V_{c_1}}$. Such assertions will be obtained by
introducing new concept names associated with those field names in a form \( \mathcal{F} \), and new constant names corresponding to the values present in \( V \). Formally:

**Definition 6 (Form assertional box)** Let \( \mathcal{F} = (F,T,C) \) be a form schema, with \( F = \{ f_1, \ldots, f_n \} \), and let \( \mathcal{F}_V^j \) be a form instance. We define the form assertional box \( A_{\mathcal{F}_V^j} \) as the set:

\[
A_{\mathcal{F}_V^j} = \{ C(c_j), f_1(c_j,v_1), \ldots, f_n(c_j,v_n) \}.
\]

**Example 4 (Continues Example 3)** Given the form instance in Example 3, its assertional box is the set:

\[
A_{\mathcal{F}_V^j} = \{ \text{customer}(c_1), \text{cstmr\_name}(c_1,john), \\
\quad \text{cstmr\_profession}(c_1,\text{phd\_student}), \text{cstmr\_income}(c_1,400), \\
\quad \text{req\_loan}(c_1,2000), \text{cstmr\_country}(c_1,\text{krakosia}) \}.
\]

Next we will show how field values can be integrated with an arbitrary \( \delta \)-ontology \( \Sigma = (T_S, T_D, A) \), adapting thus the concept of \( \delta \)-forms presented in [8] to a DL setting. Formally:

**Definition 7 (Form schema with defeasible attributes. \( \delta \)-form instance)** Let \( \mathcal{F} = (F,T,C) \) be a form schema, and \( \Sigma = (T_S, T_D, A) \) a \( \delta \)-ontology. A form schema with defeasible attributes (or \( \delta \)-form schema) \( D \) is a pair \( (\mathcal{F}, \Sigma) \). If \( V \) is a set of values for the form \( \mathcal{F} \), a \( \delta \)-form instance \( D_V \) is the pair \( (\mathcal{F}_V, \Sigma) \). The set of defeasible attributes for \( D_V \) is defined as the set of predicates \( \text{Pred}((\mathcal{T}_\Pi(T_S) \cup T_\Pi(A) \cup T_\Pi(A_{\mathcal{F}_V^j}), T_\Delta(T_D))) \).

Given a \( \delta \)-form schema \( (\mathcal{F}, \Sigma) \), the above definition aims at identifying features or attributes in the form \( \mathcal{F} \) encoded by the form designer as distinguished predicates in the program \( P = (T_\Pi(T_S) \cup T_\Pi(A) \cup T_\Pi(A_{\mathcal{F}_V^j}), T_\Delta(T_D))) \). Such attributes are said to be defeasible, as their associated value will be determined by DeLP queries solved w.r.t. the DeLP program \( P \). Hence, changing the field values in the form \( \mathcal{F} \) or changing the ontology \( \Sigma \) will result in changing the value for these attributes. As stated before, defeasible attributes will represent relevant features for the form designer, whose value depends on the ontology encoding relevant domain knowledge with the addition of particular assertions which represent the field values for a given form instance.

**Example 5** If individual \( c_1 \) named John (as presented in Figure 5) fills in the form (as in Figure 9), a form instance as the one presented in Example 3 is obtained. When a dialectical process is carried out on the DeLP program obtained from \( \Sigma_{\text{john}} = (T_S^{\text{criteria}}, T_D^{\text{criteria}} \cup T_D^{\text{bank}} \cup T_D^{\text{HSO}}, A_{\text{bank}} \cup A_{\text{HSO}} \cup A_{\mathcal{F}_V^j}) \) the membership of the individual \( c_1 \) would not be determined to the concept candidate w.r.t. to the form \( (\mathcal{F}, \Sigma_{\text{john}}) \) and no loan will be granted to John. When Danae fills in the form, she will be granted a loan because the membership of the individual \( c_3 \) to the concept candidate will be determined w.r.t. to the form \( (\mathcal{F}, \Sigma_{\text{danae}}) \) where \( \Sigma_{\text{danae}} = (T_S^{\text{criteria}}, T_D^{\text{criteria}} \cup T_D^{\text{bank}} \cup T_D^{\text{HSO}}, A_{\text{bank}} \cup A_{\text{HSO}} \cup \{(3), (13)-(16), (20)\}) \).

\[1\]
In the case of the ontology \( \Sigma_{\text{danae}} \), the set \( \{(3), (13)-(16), (20)\} \) is considered w.r.t. the Abox \( A_{\text{user}} \) in Fig. 5.
4 Conclusions

We have presented a novel argument-based approach for enriching traditional forms for web-based environments, which can be suitably adapted to existing technologies in the context of the Semantic Web initiative. Our proposal involves providing the possibility of modeling inferences based on concepts which are part of the intended meaning of a form, which we have formalized as defeasible attributes. These defeasible attributes are in turn specified by a form designer by means of a DL ontology. As this DL ontology can be inconsistent, an argumentative process has to be carried out for determining the ultimate value of such defeasible attributes.

Acknowledgments

This research was funded by Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 No. 13.096, PICT 2003 No. 15.043, PAV 2004 076), by CONICET (Argentina), by Project TIN2006-15662-C02-01 (MEC, Spain), Project 24/ZN10 (SGCyT, UNS).

References