ASP Based Generation of Information Terms for Constructive $\mathcal{EL}$

Loris Bozzato

Fondazione Bruno Kessler,
Via Sommarive 18, 38123 Trento, Italy
bozzato@fbk.eu

Abstract. Constructive description logics define interpretations of description logics under different constructive semantics. These logics have been mostly studied from the point of view of their formal properties: limited practical approaches have been shown for their use in knowledge representation and Semantic Web languages and tools (which, on the other hand, constitute the distinctive applications of description logics).
In this paper we demonstrate a solution to address this aspect: from the theoretical point of view, we first introduce an information terms semantics for the minimal description logic $\mathcal{EL}$ and we establish formal results linking this constructive semantics to answer set semantics. Using these results, on the practical side, we then present a prototype managing one aspect of such semantics (the generation of information terms of a knowledge base) using OWL-EL ontologies and "off the shelf" tools.

1 Introduction

Constructive description logics define interpretations of description logics under constructive semantics. The need for such reinterpretation of description logics in non-classical semantics mostly arises for the interest in applying the formal properties of these semantics to solve modelling or reasoning problems. Starting from different representation interests and reference constructive semantics, several constructive characterizations of description logics have been recently proposed, like e.g. [6,11,19,22].

However, constructive description logics have been mostly studied from the point of view of their formal properties, but limited practical approaches have been shown for their application to knowledge representation and Semantic Web languages and tools (which, on the other hand, constitute the distinctive application of description logics). Among the "real world" use of constructive description logics in applications and systems, for example, we can cite reasoning over incomplete data streams [18], managing conflicts over legal ontologies [15], and a framework for the composition of semantic services in heterogeneous domains [16] (based on [5]).

In this paper we want to demonstrate a direction for the solution of this aspect: from the theoretical point of view, we introduce a minimal constructive description logic based on $\mathcal{EL}$ [2] and we extend to its semantics formal results linking it to *Answer Set Programming (ASP)*; on the practical side, by taking advantage of these properties,

we present a prototype managing one task over the constructive semantics (namely, the generation of valid "snapshots" of a knowledge base) over (a subset of) the standard OWL-EL profile [21] and "off the shelf" tools for ontology management (OWL API) and Answer Set Programming (DLV solver).

In particular, in our work we chose to concentrate on the description logic $\mathcal{EL}$ because, on one hand, it is one of the simplest description logics over which semantics enjoying constructive properties can be defined (cfr. *explicit definability* property [11]). On the other hand, $\mathcal{EL}$ is recognized as one of the reference languages for (low complexity) description logics and, as such, it is at the base of the OWL-EL profile [21] and languages of well-known large ontologies (e.g. *Gene Ontology*[1] and *GALEN*[2]).

The task we consider in this work regards the generation of valid *information terms* for a given knowledge base. Intuitively, in our semantics information terms are mathematical objects providing a constructive justification for the truth of a formula. Notably, they can be seen as representing the *state* of such formula: thus, in this light, generating information terms for a knowledge base correspond to *validate* its representations by generating a set of its possible valid states. This approach was used in the *CooML* modelling language [23] and related works for the generation of valid "snapshots" of information systems descriptions [12,14]: we will follow the direction of these works (in particular the relations to ASP studied in [13]) to formulate our solution. Related to this task is the algorithm GENIT used in [7] to generate states and validate the output of actions over constructive $\mathcal{ALC}$.

As mentioned, the solution we propose is based on the relations across information terms semantics and answer sets semantics: we note that this allow us, for the first time, to examine these relations also in the context of constructive description logics.

We can summarize the contributions of this paper as follows:

- We introduce (in Section 2) an information terms semantics for the minimal description logic $\mathcal{EL}$ [2]. This constructive semantics is a straightforward restriction of the basic constructive description logic $\mathcal{BCDL}$ [11] to the syntax of $\mathcal{EL}$.
- On the base of the relations across information terms semantics and answer sets for nested expressions highlighted in [13], we provide (in Section 3) results establishing a formal relation between answer sets for formulas in $\mathcal{EL}$ and for their information terms. In Section 4, these properties are used to formulate a datalog rewriting for the generation of the sets of information terms of an input $\mathcal{EL}$ knowledge base
- Using these formal results, in Section 5 we present Asp-it[3], a prototype implementation of an information terms generator for ontologies in OWL-EL. Asp-it applies the presented datalog rewriting to the input ontology: the computation of answer sets is obtained by interacting with the DLV solver and the resulting information terms are returned as an annotation to the OWL axioms in the original ontology.

[1] `http://geneontology.org/`
[2] `http://www.opengalen.org/`
[3] `https://github.com/dkmfbk/asp-it`

2 $\mathcal{EL}$c: a constructive semantics for $\mathcal{EL}$

In this section we present a constructive semantics based on the minimal description logic $\mathcal{EL}$ [2]. We will refer to this logic as $\mathcal{EL}$c: the presentation and information terms semantics of $\mathcal{EL}$c are defined as a straightforward restriction of the logic $\mathcal{BCDL}$ [11].

Syntax. The language $\mathcal{L}$ for $\mathcal{EL}$c is based on the disjoint denumerable sets $\mathtt{NR}$ of *role names*, $\mathtt{NC}$ of *concept names* and $\mathtt{NI}$ of *individual names*.

Differently with respect to standard presentations of description logics, in $\mathcal{L}$ we consider a set $\mathtt{NG}$ of special concepts, called *generators*, where $\mathtt{NG} \cap \mathtt{NC} = \emptyset$. Generators are used in the definition of a limited form of subsumption, which facilitates the characterization of the logic in a constructive semantics. A generator G is an atomic concept with associated a finite set of individual names $\textsc{dom}(G)$ (the *domain* of G) which fixes the interpretation of G. In our language, we use bounded quantified formulas of the kind $\forall_G C$, meaning that every element of $\textsc{dom}(G)$ belongs to the concept C. Also, differently from the usual presentation of $\mathcal{EL}$, we limit the use of the $\top$ constructor as a special kind of generator $\top_{\mathcal{N}} \in \mathtt{NG}$ such that $\textsc{dom}(\top_{\mathcal{N}}) = \mathcal{N}$ and for all generators G, $\textsc{dom}(G) \subseteq \mathcal{N}$. In the language $\mathcal{L}$ for $\mathcal{EL}$c, *concepts* C are expressions of the kind:

$$C ::= A \,|\, C \sqcap C \,|\, \exists R.C$$

where $A \in \mathtt{NC} \cup \mathtt{NG}$ and $R \in \mathtt{NR}$. Let $\mathtt{VAR}$ be a denumerable set of individual variables, the *formulas* K of $\mathcal{L}$ are defined as:

$$K ::= R(s,t) \,|\, C(t) \,|\, \forall_G C$$

where $s, t \in \mathtt{NI} \cup \mathtt{VAR}$, $R \in \mathtt{NR}$, $G \in \mathtt{NG}$ and C is a concept. A formula is *atomic* if it is of the kind $A(t)$ with $A \in \mathtt{NC} \cup \mathtt{NG}$ or $R(s,t)$. A formula is *closed* if it does not contain variables.

A theory $\mathcal{K}$ (namely, a knowledge base) can be divided as usual in TBox and ABox. An ABox is a finite set of closed formulas of the kind $C(d)$ and $R(c,d)$ with $R \in \mathtt{NR}$, $c, d \in \mathtt{NI}$ and C a concept. A TBox is a finite set of formulas of the kind $\forall_G C$ (i.e. "restricted" concept inclusions).

In the following we refer to languages $\mathcal{L}_{\mathcal{N}}$ restricted to subsets $\mathcal{N}$ of $\mathtt{NI}$. Given $\mathcal{N} \subseteq \mathtt{NI}$, let $\mathtt{NG}_{\mathcal{N}}$ be the set of generators $G \in \mathtt{NG}$ s.t. $\textsc{dom}(G) \subseteq \mathcal{N}$ with $\textsc{dom}(\top_{\mathcal{N}}) = \mathcal{N}$. We denote with $\mathcal{L}_{\mathcal{N}}$ the language built on the set $\mathcal{N}$ of individual names, the set $\mathtt{NC}$ of concept names, the set $\mathtt{NR}$ of role names and the set $\mathtt{NG}_{\mathcal{N}}$ of generators.

Classical semantics. A *model* $\mathcal{M}$ for $\mathcal{L}_{\mathcal{N}}$ is a pair $\langle \Delta^{\mathcal{M}}, \cdot^{\mathcal{M}} \rangle$, where the domain $\Delta^{\mathcal{M}}$ is a non-empty set and $\cdot^{\mathcal{M}}$ is a valuation map such that: for every $c \in \mathcal{N}$, $c^{\mathcal{M}} \in \Delta^{\mathcal{M}}$; for every $C \in \mathtt{NC}$, $C^{\mathcal{M}} \subseteq \Delta^{\mathcal{M}}$; for every $R \in \mathtt{NR}$, $R^{\mathcal{M}} \subseteq \Delta^{\mathcal{M}} \times \Delta^{\mathcal{M}}$; for every $G \in \mathtt{NG}_{\mathcal{N}}$, if $\textsc{dom}(G) = \{c_1, \ldots, c_n\}$, then $G^{\mathcal{M}} = \{c_1^{\mathcal{M}}, \ldots, c_n^{\mathcal{M}}\}$. Classical interpretation of non-atomic concepts is defined by the evaluation of description logics operators:

$$(C_1 \sqcap C_2)^{\mathcal{M}} = C_1^{\mathcal{M}} \cap C_2^{\mathcal{M}}$$

$$(\exists R.C)^{\mathcal{M}} = \{c \in \Delta^{\mathcal{M}} \mid \text{there exists } d \in \Delta^{\mathcal{M}} \text{ s.t. } (c,d) \in R^{\mathcal{M}} \text{ and } d \in C^{\mathcal{M}}\}$$

An *assignment* on $\mathcal{M}$ is a map $\theta : \mathtt{VAR} \to \Delta^{\mathcal{M}}$. If $t \in \mathtt{NI} \cup \mathtt{VAR}$, $t^{\mathcal{M},\theta}$ is the element of $\Delta^{\mathcal{M}}$ denoting t in $\mathcal{M}$ w.r.t. θ, namely: $t^{\mathcal{M},\theta} = \theta(t)$ if $t \in \mathtt{VAR}$ and $t^{\mathcal{M},\theta} = t^{\mathcal{M}}$ if

$t \in \mathtt{NI}$. A formula K is *valid* in $\mathcal{M}$ w.r.t. θ, and we write $\mathcal{M}, \theta \models K$, if one of the following conditions holds:

$$\begin{aligned} \mathcal{M} \models R(s,t) &\text{ iff } (s^{\mathcal{M},\theta}, t^{\mathcal{M},\theta}) \in R^{\mathcal{M}} \\ \mathcal{M} \models C(t) &\text{ iff } t^{\mathcal{M},\theta} \in C^{\mathcal{M}} \\ \mathcal{M} \models \forall_G C &\text{ iff } G^{\mathcal{M}} \subseteq C^{\mathcal{M}} \end{aligned}$$

If Γ is a set of formulas, $\mathcal{M} \models \Gamma$ means that $\mathcal{M} \models K$ for every $K \in \Gamma$. K is a *logical consequence* of Γ, and we write $\Gamma \models K$ iff, for every $\mathcal{M}$, $\mathcal{M} \models \Gamma$ implies $\mathcal{M} \models K$.

As noted in [11], while non-conventional in description logics languages, the provided definition of generators with a fixed domain simplifies the following presentations of the logic: alternatively, domain of generators can be defined by extending the language with nominals.

Example 1. We build our running example over the one presented in [3,11]. In this example, inspired to the classical example of [8], we want to describe the correct pairings between food and wines using an $\mathcal{EL}$c knowledge base $\mathcal{K}_W$. The TBox of $\mathcal{K}_W$ contains:

$$(Ax_1)\ \forall_{Food} \exists goesWith.Color \qquad (Ax_2)\ \forall_{Color} \exists isColorOf.Wine$$

with $Wine \in \mathtt{NC}$, $isColorOf, goesWith \in \mathtt{NR}$ and $Food, Color \in \mathtt{NG}$. Intuitively, axiom (Ax_1) states that each $Food$ has an appropriate wine $Color$ and (Ax_2) defines that there exists a $Wine$ for every $Color$. Domains of the generators are defined as:

$$\textsc{dom}(Food) = \{fish, meat\} \qquad \textsc{dom}(Color) = \{red, white\}$$

The ABox of $\mathcal{K}_W$ contains the following assertions:

$Wine(barolo)$	$isColorOf(red, barolo)$	$goesWith(fish, white)$
$Wine(chardonnay)$	$isColorOf(white, chardonnay)$	$goesWith(meat, red)$

We implicitly consider as the finite set $\mathcal{N}$ of individual names the set containing all the individual names used in $\mathcal{K}_W$.

A (classical) model $\mathcal{M}$ for $\mathcal{K}_W$ must interpret $Food^{\mathcal{M}} = \{fish^{\mathcal{M}}, meat^{\mathcal{M}}\}$ and $Color^{\mathcal{M}} = \{red^{\mathcal{M}}, white^{\mathcal{M}}\}$. Note that, by the interpretation of ABox assertions, $barolo^{\mathcal{M}}$ and $chardonnay^{\mathcal{M}}$ belong to $Wine^{\mathcal{M}}$, but since it is not a generator, $Wine^{\mathcal{M}}$ might contain other domain elements. $\diamond$

Information terms semantics. The constructive semantics for $\mathcal{EL}$c is based on the notion of *information term* [20]. Information terms semantics is related to the ***BHK (Brower-Heyting-Kolmogorov)*** interpretation of logical connectives [26]: intuitively, an information term η for a formula K is a syntactical object that constructively *justifies* the truth of K in a classical model $\mathcal{M}$. For example, the validity of the formula $\exists R.C(a)$ in a model $\mathcal{M}$ can be explained by an information term (b, α) providing the filler b s.t. $(a^{\mathcal{M}}, b^{\mathcal{M}}) \in R^{\mathcal{M}}$ and, inductively, an information term α justifying $b^{\mathcal{M}} \in C^{\mathcal{M}}$.

Given a finite subset $\mathcal{N}$ of $\mathtt{NI}$ and a closed formula K of $\mathcal{L}_{\mathcal{N}}$, we define the set of *information terms* $\textsc{it}_{\mathcal{N}}(K)$ by induction on K as follows.

$$\begin{aligned} \textsc{it}_{\mathcal{N}}(K) &= \{\, \mathtt{tt} \,\}, \text{ if } K \text{ is an atomic formula} \\ \textsc{it}_{\mathcal{N}}(C_1 \sqcap C_2(c)) &= \{\, (\alpha, \beta) \mid \alpha \in \textsc{it}_{\mathcal{N}}(C_1(c)) \text{ and } \beta \in \textsc{it}_{\mathcal{N}}(C_2(c)) \,\} \\ \textsc{it}_{\mathcal{N}}(\exists R.C(c)) &= \{\, (d, \alpha) \mid d \in \mathcal{N} \text{ and } \alpha \in \textsc{it}_{\mathcal{N}}(C(d)) \,\} \\ \textsc{it}_{\mathcal{N}}(\forall_G C) &= \{\, \phi : \textsc{dom}(G) \to \textstyle\bigcup_{d \in \textsc{dom}(G)} \textsc{it}_{\mathcal{N}}(C(d)) \mid \phi(d) \in \textsc{it}_{\mathcal{N}}(C(d)) \,\} \end{aligned}$$

The justification of formulas in classical models with respect to one of their information terms is given by the realizability relation. Let $\mathcal{M}$ be a model for $\mathcal{L}_\mathcal{N}$, K a closed formula of $\mathcal{L}_\mathcal{N}$ and $\eta \in \text{IT}_\mathcal{N}(K)$. We define the *realizability relation* $\mathcal{M} \rhd \langle \eta \rangle\, K$ by induction on the structure of K.

$$\begin{aligned}
\mathcal{M} \rhd \langle \mathtt{tt} \rangle\, K &\text{ iff } \mathcal{M} \models K, \text{ where } K \text{ is an atomic formula} \\
\mathcal{M} \rhd \langle (\alpha, \beta) \rangle\, C_1 \sqcap C_2(c) &\text{ iff } \mathcal{M} \rhd \langle \alpha \rangle\, C_1(c) \text{ and } \mathcal{M} \rhd \langle \beta \rangle\, C_2(c) \\
\mathcal{M} \rhd \langle (d, \alpha) \rangle\, \exists R.C(c) &\text{ iff } \mathcal{M} \models R(c, d) \text{ and } \mathcal{M} \rhd \langle \alpha \rangle\, C(d) \\
\mathcal{M} \rhd \langle \phi \rangle\, \forall_G C &\text{ iff, for every } d \in \text{DOM}(G),\ \mathcal{M} \rhd \langle \phi(d) \rangle\, C(d)
\end{aligned}$$

If Γ is a finite set of closed formulas $\{K_1, \ldots, K_n\}$ of $\mathcal{L}_\mathcal{N}$, $\text{IT}_\mathcal{N}(\Gamma)$ denotes the set of n-tuples $\overline{\eta} = (\eta_1, \ldots, \eta_n)$ such that, for every $j \in \{1, \ldots, n\}$, $\eta_j \in \text{IT}_\mathcal{N}(K_j)$. We write $\mathcal{M} \rhd \langle \overline{\eta} \rangle\, \Gamma$ iff, for every $j \in \{1, \ldots, n\}$, $\mathcal{M} \rhd \langle \eta_j \rangle\, K_j$.

Example 2. Let us consider the example knowledge base $\mathcal{K}_W$ and the axiom (Ax_1) in its TBox: every element $\phi \in \text{IT}_\mathcal{N}(Ax_1)$ is a function mapping each food $f \in \text{DOM}(Food)$ to an information term $\phi(f) \in \text{IT}_\mathcal{N}(\exists goesWith.Color)$. Thus, every $\phi(f)$ has the form $(c, \mathtt{tt})$, meaning that c is a $Color$ that is admitted for f. For example, we can consider $\psi_1 \in \text{IT}_\mathcal{N}(Ax_1)$ defined as:

$$[\ fish \mapsto (white, \mathtt{tt}),\ meat \mapsto (red, \mathtt{tt})\ ]$$

It is easy to see that if $\mathcal{M}$ is a model for the ABox of $\mathcal{K}_W$, then $\mathcal{M} \rhd \langle \psi_1 \rangle\, Ax_1$. Similarly, if we consider $\psi_2 \in \text{IT}_\mathcal{N}(Ax_2)$ where

$$[\ red \mapsto (barolo, \mathtt{tt}),\ white \mapsto (chardonnay, \mathtt{tt})\ ]$$

we have that $\mathcal{M} \rhd \langle \psi_2 \rangle\, Ax_2$. ◇

The following result, provable by induction on the structure of K, shows the relation across the classical and constructive semantics.

Proposition 1. *Let $\mathcal{N}$ be a finite subset of* NI*, K a closed formula of $\mathcal{L}_\mathcal{N}$ and $\eta \in \text{IT}_\mathcal{N}(K)$. For every model $\mathcal{M}$, $\mathcal{M} \rhd \langle \eta \rangle\, K$ implies $\mathcal{M} \models K$.* □

Thus, the constructive semantics is compatible with the classical one: a consequence of this is that the constructive semantics maintains the classical declarative reading of DL formulas.

Such definition of constructive realization leads to a constructive version of the logical consequence relation, that we call constructive consequence. Given a set of closed formulas $\Gamma \cup \{K\} \in \mathcal{L}_\mathcal{N}$, we say that K is a *constructive consequence* of Γ (denoted $\Gamma \overset{\text{c}}{\models} K$) iff, for every $\overline{\gamma} \in \text{IT}_\mathcal{N}(\Gamma)$, there exists a $\eta \in \text{IT}_\mathcal{N}(K)$ such that for every model $\mathcal{M}$ for $\mathcal{L}_\mathcal{N}$, $\mathcal{M} \rhd \langle \overline{\gamma} \rangle\, \Gamma$ implies $\mathcal{M} \rhd \langle \eta \rangle\, K$.

One important aspect (deeper investigated in [3,11]) is that such relation $\Gamma \overset{\text{c}}{\models} K$ implicitly defines a semantic map $\Phi_\mathcal{N}$ transforming information terms in $\text{IT}_\mathcal{N}(\Gamma)$ to information terms of $\text{IT}_\mathcal{N}(K)$ such that if $\mathcal{M} \rhd \langle \overline{\gamma} \rangle\, \Gamma$, then $\mathcal{M} \rhd \langle \Phi_\mathcal{N}(\overline{\gamma}) \rangle\, K$. This idea is strongly related to the *proofs-as-programs* paradigm: in $\mathcal{BCDL}$ it is shown how it is possible to extract this map from the proofs of a natural deduction calculus which is sound and complete w.r.t. constructive consequence.

Our interest in the remainder of the paper is somewhat preliminary to this step: we want to be able to compute the information terms of a set of input $\mathcal{EL}$ formulas (corresponding to the input knowledge base) by means of the relations between answer sets semantics and information terms semantics.

3 Answer sets semantics for formulas and information terms

In this section we build on the work in [13], which investigates the relations between information terms and answer set semantics of logic programs with nested expressions, and we reinterpret it in our scenario.

Given the definition of formulas and information terms given above, we can give a characterization of the "snapshot" of a formula defined by one of its information terms with the definition of piece of information. We call *piece of information* over $\mathcal{L}_\mathcal{N}$ an expression of the kind $\langle\eta\rangle K$ with $K \in \mathcal{L}_\mathcal{N}$ a closed formula and $\eta \in \text{IT}_\mathcal{N}(K)$.

Following the construction in [13], we adapt the definitions for logic programs with nested expressions [17] to the structure and reading of $\mathcal{EL}$c formulas. In this regard, we call *lp-intepretation* I any set of closed atomic formulas. Given a closed formula $K \in \mathcal{L}_\mathcal{N}$, the usual satisfiability relation $I \models K$ can naturally be defined as:

$$
\begin{aligned}
&I \models K, \text{ iff } K \in I \text{ and } K \text{ is atomic}\\
&I \models C \sqcap D(c) \text{ iff } I \models C(c) \text{ and } I \models D(c)\\
&I \models \exists R.C(c) \text{ iff } R(c,d) \in I \text{ for } d \in \mathcal{N} \text{ and } I \models C(d)\\
&I \models \forall_G C \text{ iff for every } e \in \text{DOM}(G), I \models C(e)\\
&I \models \Gamma \text{ iff } I \models K \text{ for } K \in \Gamma
\end{aligned}
$$

Clearly, this is consistent with the definition of classical logical consequence. We can define as follows the notion of answer set for our formulas:

Definition 1. *An lp-interpretation I is an* answer set *for a set of closed formulas $\Gamma \subseteq \mathcal{L}_\mathcal{N}$ iff $I \models \Gamma$ and, for every $I' \subseteq I$, $I' \models \Gamma$ implies $I' = I$.*

Given this definition of answer set on $\mathcal{EL}$c closed formulas, in the following we want to extend this notion to the pieces of information. The idea is to denote the set of "answers" that one can derive from the contents of informations terms on the given formula. As noted in [13], the formula represents intuitively a "query" for the information terms that realize it in an interpretation: for example, asking "$\exists R.A(c)$?" corresponds to asking for an information term (d, α) for the formula such that $R(c, d)$ holds and α is an answer satisfying $A(d)$. Given a piece of information $\langle\eta\rangle K$, the following defines the sets of answers $\texttt{ans}(\langle\eta\rangle K)$ obtainable from it:

$$
\begin{aligned}
\texttt{ans}(\langle \texttt{tt}\rangle K) &= \{K\}, \text{ with } K \text{ an atomic formula}\\
\texttt{ans}(\langle(\alpha,\beta)\rangle A_1 \sqcap A_2(c)) &= \texttt{ans}(\langle\alpha\rangle A_1(c)) \cup \texttt{ans}(\langle\beta\rangle A_2(c))\\
\texttt{ans}(\langle(d,\alpha)\rangle \exists R.A(c)) &= \{R(c,d)\} \cup \texttt{ans}(\langle\alpha\rangle A(d))\\
\texttt{ans}(\langle\phi\rangle \forall_G A) &= \textstyle\bigcup_{d\in \text{DOM}(G)} \texttt{ans}(\langle\phi(d)\rangle A(d))
\end{aligned}
$$

We remark that $\texttt{ans}(\langle\eta\rangle K)$ is a finite set of atomic formulas[4].

Example 3. If we recall the information term $\psi_1 \in \text{IT}_{\mathcal{N}}(Ax_1)$, by the definition of sets of answers for $\langle\psi_1\rangle Ax_1$ we have that:

$$\begin{aligned}\texttt{ans}(\langle\psi_1\rangle Ax_1) &= \texttt{ans}(\langle\psi_1(fish)\rangle\, H_1(fish)) \cup \texttt{ans}(\langle\psi_1(meat)\rangle\, H_1(meat)) \\ &= \texttt{ans}(\langle\texttt{tt}\rangle Color(white)) \cup \{goesWith(fish, white)\} \cup \\ &\quad\ \texttt{ans}(\langle\texttt{tt}\rangle Color(red)) \cup \{goesWith(meat, red)\} \\ &= \{Color(white), goesWith(fish, white), \\ &\quad\ Color(red), goesWith(meat, red)\}\end{aligned}$$

where we abbreviate $H_1 = \exists goesWith.Color$. Similarly, considering $\psi_2 \in \text{IT}_{\mathcal{N}}(Ax_2)$ from previous example:

$$\begin{aligned}\texttt{ans}(\langle\psi_2\rangle Ax_2) = \{&Wine(barolo), isColorOf(red, barolo), \\ &Wine(chardonnay), isColorOf(white, chardonnay)\}\end{aligned}$$

$\diamond$

We can show the following relation to lp-interpretations (by an easy induction on the structure of formulas):

Theorem 1. *Let $\mathcal{N}$ be a finite subset of* NI*, K a closed formula of $\mathcal{L}_{\mathcal{N}}$. For every lp-interpretation I, $I \models K$ iff there exists an information term $\eta \in \text{IT}_{\mathcal{N}}(K)$ such that $I \models \texttt{ans}(\langle\eta\rangle K)$.* □

Moreover, we can connect this notion of set of atomic answers to the realizability relation with the following theorem. By induction on the structure of K, we have:

Theorem 2. *Let $\mathcal{N}$ be a finite subset of* NI*, K a closed formula of $\mathcal{L}_{\mathcal{N}}$. For every model $\mathcal{M}$, $\mathcal{M} \rhd \langle\eta\rangle\, K$ iff $\mathcal{M} \models \texttt{ans}(\langle\eta\rangle K)$.* □

Note that this result reduces the problem of determining the realizability of a piece of information to the classical satisfiability of a finite set of atomic formulas.

Now we can study the relations between answer sets for $\mathcal{EL}$c formulas and pieces of information. Intuitively, $\texttt{ans}(\langle\eta\rangle K)$ represents the information needed to get evidence for K according to the information term η: we want to define a set that describes the *minimal knowledge needed* to justify the realizability of its piece of information.

We say that an lp-interpretation I is a minimal model of $\langle\eta\rangle K$ if, for every model $\mathcal{M}$ of $\mathcal{L}_{\mathcal{N}}$, I is the subset minimal lp-interpretation such that $\mathcal{M} \models I$ implies $\mathcal{M} \rhd \langle\eta\rangle\, K$. Using previous results, in one direction we can show:

Theorem 3. *If I is an answer set for a closed formula $K \in \mathcal{L}_{\mathcal{N}}$, then there exists a piece of information $\langle\eta\rangle K$, with $\eta \in \text{IT}_{\mathcal{N}}(K)$, such that I is a minimal model of $\langle\eta\rangle K$.*

Proof. Since I is an answer set for K, we have that $I \models K$ and by Theorem 1 there exists $\eta \in \text{IT}_{\mathcal{N}}(K)$ s.t. $I \models \texttt{ans}(\langle\eta\rangle K)$. By Theorem 2, for each $\mathcal{M} \models I$ this implies $\mathcal{M} \rhd \langle\eta\rangle\, K$. We can prove that I is minimal: suppose that $I' \subseteq I$ s.t. $I' \models \texttt{ans}(\langle\eta\rangle K)$, then by Theorem 1, $I' \models K$. Since I is an answer set for K, then $I' = I$. □

[4] This definition of answer sets for pieces of information corresponds to the notion of *information content* used in [7,14].

In the other direction, we need to define a notion of minimality on pieces of information as follows:

Definition 2. *Let K be a closed formula of $\mathcal{L}_{\mathcal{N}}$. A piece of information $\langle\eta\rangle K$ is* minimal *iff there is no $\eta' \in \text{IT}_{\mathcal{N}}(K)$ such that $\texttt{ans}(\langle\eta'\rangle K) \subset \texttt{ans}(\langle\eta\rangle K)$.*

Intuitively, the answers $\texttt{ans}(\langle\eta\rangle K)$ of minimal pieces of information characterize the sets of atoms whose truth is strictly necessary to get evidence for K. Using this definition, we can then prove the other direction of the relation:

Theorem 4. *Let K be a closed formula of $\mathcal{L}_{\mathcal{N}}$ and $\langle\eta\rangle K$ be a minimal piece of information for K. Then, $\texttt{ans}(\langle\eta\rangle K)$ is an answer set for K.*

Proof. By Theorem 1, for every lp-interpretation I s.t. $I \models \texttt{ans}(\langle\eta\rangle K)$ (that is, every I s.t. $\texttt{ans}(\langle\eta\rangle K) \subseteq I$, given that it is a set of atomic formulas) we have $I \models K$. Hence, considering $I = \texttt{ans}(\langle\eta\rangle K)$, also $\texttt{ans}(\langle\eta\rangle K) \models K$.

Moreover, let us consider $I' \subseteq \texttt{ans}(\langle\eta\rangle K)$ with $I' \models K$. Then, by Theorem 1, there exists a $\beta \in \text{IT}_{\mathcal{N}}(K)$ s.t. $I' \models \texttt{ans}(\langle\beta\rangle K)$. Thus, $\texttt{ans}(\langle\beta\rangle K) \subseteq I'$ and, for the minimality of $\langle\eta\rangle K$, this implies that $\texttt{ans}(\langle\beta\rangle K) = I' = \texttt{ans}(\langle\eta\rangle K)$. □

Thus, we can complete these results with the following theorem:

Theorem 5. *Let K be a closed formula of $\mathcal{L}_{\mathcal{N}}$. I is an answer set for K iff there exists a minimal piece of information $\langle\eta\rangle K$ such that $I = \texttt{ans}(\langle\eta\rangle K)$.*

Proof. By Theorem 4 we directly have the "only-if" direction: if $\langle\eta\rangle K$ is a minimal piece of information, $I = \texttt{ans}(\langle\eta\rangle K)$ is an answer set for K.

In the other direction, by Theorem 3 if I is an answer set for K, then there exists $\langle\eta\rangle K$ s.t. I is a minimal model for $\langle\eta\rangle K$. Thus, by Theorem 2, $I \models \texttt{ans}(\langle\eta\rangle K)$ which implies that $\texttt{ans}(\langle\eta\rangle K) = I$ (since I is the minimal set of atoms s.t. $I \models K$).

We can show that $\langle\eta\rangle K$ is a minimal piece of information: let us suppose that there exists a $\beta \in \text{IT}_{\mathcal{N}}(K)$ such that $\texttt{ans}(\langle\beta\rangle K) \subseteq \texttt{ans}(\langle\eta\rangle K)$. Thus, $\texttt{ans}(\langle\beta\rangle K) \subseteq I$ and, for the minimality of the answer set I, we have $\texttt{ans}(\langle\beta\rangle K) = I = \texttt{ans}(\langle\eta\rangle K)$. □

4 ASP based generation of information terms

From the results of previous section, one solution to generate (minimal) information terms consists in computing the answer sets of the input set of formulas and then, for each formula K, use the recursive definition of $\texttt{ans}(\langle\eta\rangle K)$ to reconstruct each information term η. The computation of answer sets can be achieved by translating the initial knowledge base to a datalog program, for instance by applying transformations of formulas to datalog facts and rules akin to the ones used for the translation of nested expressions (see e.g. [24]).

In the following we provide a similar translation, but we also include a set of rules that keep track and recursively compose the information terms for the input formulas. The datalog rewriting we propose basically follows a *generate-and-test* approach: a first rewriting (P_1) of the input $\mathcal{EL}$ knowledge base translate the formulas by using

their logical reading, in order to generate all the alternative interpretations that satisfies them; the second part (P_2) of the rewriting, adds rules to reconstruct from these all the possible information terms of input formulas. Thus, while P_1 is related to the definition of interpretation for each formula K, P_2 corresponds to the definition of $\texttt{ans}(\langle\eta\rangle K)$.

In the following, we assume the usual definitions for *normal logic programs* under answer set semantics (see e.g. [10] for an introduction). Note, however, that in the formulation of rules (and in their implementation) we used the DLV notation for complex *list terms*: thus considerations about management of complex and function terms and finiteness of the domain apply (see e.g. [9]).

In this version of the rewriting, we consider the case in which knowledge about roles is complete: in other words, we only consider roles assertions that are included in the input set of formulas. Formally, given a finite subset $\mathcal{N} \in \texttt{NI}$, we consider subsets $\mathcal{R}$ of the set $\mathcal{R}_\mathcal{N} = \{R(c,d) \mid R \in \texttt{NR} \text{ and } c,d \in \mathcal{N}\}$ of role assertions over $\mathcal{N}$. As proposed in [24] for the translation of nested expressions, we use a labelling of concepts to decompose complex formulas. Considering a finite set L of constants, we call $l_C \in L$ the label encoding the (possibly complex) concept C.

Let $\mathcal{R}$ be a finite subset of $\mathcal{R}_\mathcal{N}$. Given an input set Γ of closed formulas of $\mathcal{L}_\mathcal{N}$ with $\mathcal{R} \subseteq \Gamma$, for each formula $K \in \Gamma$, the *model generating* rewriting $P_1(K)$ is defined as:

$$
\begin{aligned}
A(b) &\mapsto \{\, is(b,A) \leftarrow is(b,l_A).\,\} \\
R(a,b) &\mapsto \{\, rel(a,R,b).\,\} \\
C \sqcap D(a) &\mapsto \{\, is(a,l_C) \leftarrow is(a,l_{C\sqcap D}).\\
&\qquad\quad is(a,l_D) \leftarrow is(a,l_{C\sqcap D}).\,\} \cup P_1(C(a)) \cup P_1(D(a)) \\
\exists R.C(a) &\mapsto \{\, is(x,l_C) \leftarrow rel(a,R,x), is(a,l_{\exists R.C}).\,\} \cup P_1(C(x)) \\
\forall_G C &\mapsto \{\, is(x,l_C) \leftarrow is(x,G).\,\} \cup P_1(C(x))
\end{aligned}
$$

where $A \in \texttt{NC} \cup \texttt{NG}$, C, D are possibly complex concepts, $R \in \texttt{NR}$, $a, b \in \texttt{NI}$ (or variables) and x is a variable. For each (possibly complex) concept assertion $C(d) \in \Gamma$, we add in $P_1(\Gamma)$ its "labelled" fact $L(C(d)) = \{\, is(d,l_C).\,\}$. Moreover, we assume that for each generator G used in Γ, with $\textsc{dom}(G) = \{c_1,\ldots,c_n\}$, the facts $\{\, is(c_1,G).,\ldots,is(c_n,G).\,\}$ are added to $P_1(\Gamma)$.

The *IT generating* rewriting $P_2(K)$, for each $K \in \Gamma$ is defined as follows:

$$
\begin{aligned}
A(b) &\mapsto \{\, is\_it(\texttt{tt},b,l_A) \leftarrow is(b,A).\,\} \\
R(a,b) &\mapsto \{\, rel\_it(\texttt{tt},a,R,b) \leftarrow rel(a,R,b).\,\} \\
C \sqcap D(a) &\mapsto \{\, is\_it([x,y],a,l_{C\sqcap D}) \leftarrow \\
&\qquad\quad is\_it(x,a,l_C), is\_it(y,a,l_D).\,\} \cup P_2(C(a)) \cup P_2(D(a)) \\
\exists R.C(a) &\mapsto \{\, is\_it([x,y],a,l_{\exists R.C}) \leftarrow \\
&\qquad\quad rel\_it(\texttt{tt},a,R,x), is\_it(y,x,l_C).\,\} \cup P_2(C(x)) \\
\forall_G C &\mapsto \{\, isa\_it([x,y],G,l_C) \leftarrow is(x,G), is\_it(y,x,l_C).\,\} \cup P_2(C(x))
\end{aligned}
$$

where $A \in \texttt{NC} \cup \texttt{NG}$, C and D are concepts, $R \in \texttt{NR}$, $a, b \in \texttt{NI}$ (or variables) and x, y are variables. Note that this encoding is consistent with the vision of formulas as "queries" and uses the definition of $\texttt{ans}(\langle\eta\rangle K)$ to solve the problem for bindings of a "variable" η. The complete rewriting for Γ is obtained as $P(\Gamma) = P_1(\Gamma) \cup P_2(\Gamma)$.

Example 4. Let us consider again our example knowledge base $\mathcal{K}_W$ and suppose that we add a new wine $teroldego$ to $\mathcal{K}_W$, by adding the ABox assertions:

$$Wine(teroldego) \qquad isColorOf(red, teroldego)$$

Now, by applying the rewritings to $\mathcal{K}_W$, we want to compute all possible alternative combinations of food and wines. We assume that $\mathcal{R}$ coincides to the set of role assertions in the ABox of $\mathcal{K}_W$. By applying the model generating rewriting P_1 to $\mathcal{K}_W$, it is easy to check that an answer set for $P_1(\mathcal{K}_W)$ contains the set of facts (derived from the rewriting of the ABox and domains of the generators):

$$\{is(fish, Food), is(meat, Food),\ is(red, Color), is(white, Color)$$
$$is(chardonnay, Wine), is(barolo, Wine),\ is(teroldego, Wine),$$
$$rel(fish, goesWith, white), rel(meat, goesWith, red),$$
$$rel(white, isColorOf, chardonnay), rel(red, isColorOf, barolo),$$
$$rel(red, isColorOf, teroldego)\}$$

By applying the IT generating rewriting P_2 to the axiom (Ax_2) we obtain the rules:

$$isa\_it([x, y], Color, l_{H_2}) \leftarrow is(x, Color), is\_it(y, x, l_{H_2}).$$
$$is\_it([x, y], z, l_{H_2}) \leftarrow rel(\mathtt{tt}, z, isColorOf, x), is\_it(y, x, l_{Wine}).$$
$$is\_it(\mathtt{tt}, x, l_{Wine}) \leftarrow is(x, Wine).$$

where $H_2 = \exists isColorOf.Wine$. Intuitively, by applying these rules to the model computed for $P_1(\mathcal{K}_W)$, we obtain this set of substitutions for the first term of the derived isa_it facts: $[white, [chardonnay, \mathtt{tt}]]$, $[red, [barolo, \mathtt{tt}]]$, $[red, [teroldego, \mathtt{tt}]]$. These correspond to two functions mapping each color to the alternative wines: the first equals ψ_2 from previous examples, the second is $[white \mapsto (chardonnay, \mathtt{tt}), red \mapsto (teroldego, \mathtt{tt})]$. Similarly, the application of the rewriting for (Ax_2) produces the previously presented function ψ_2.

Moreover, we can consider the following axiom that combines (Ax_1) and (Ax_2):

$$(Ax_3)\ \ \forall_{Food} \exists goesWith.(Color \sqcap \exists isColorOf.Wine)$$

One can verify that, by applying $P_2(Ax_3)$ to the presented model, its computed information terms provide all the alternative associations between food and wines:

$$[\, fish \mapsto (white, (\mathtt{tt}, (chardonnay, \mathtt{tt}))),\ meat \mapsto (red, (\mathtt{tt}, (barolo, \mathtt{tt})))\,]$$
$$[\, fish \mapsto (white, (\mathtt{tt}, (chardonnay, \mathtt{tt}))),\ meat \mapsto (red, (\mathtt{tt}, (teroldego, \mathtt{tt})))\,]$$

$\diamond$

We can provide a notion of correctness for these rewritings by the following results. By the definition of P_1 and the definition of lp-interpretations, we can prove:

Lemma 1. *Given a closed formula $K \in \mathcal{L}_{\mathcal{N}}$ and an answer set I for $P_1(\Gamma)$:*

(i). if $K = C(a)$ and $I \models is(a, l_C)$, then there exists an lp-interpretation I' for Γ s.t. $I' \models K$.

(ii). if $K = R(a, b)$ and $I \models rel(a, R, b)$, then there exists an lp-interpretation I' for Γ s.t. $I' \models K$.

Proof (Sketch). Point (ii). follows immediately considering that the fact $rel(a, R, b)$ is added in $P_1(\Gamma)$ only if $R(a, b) \in \Gamma$.

Point (i). can be shown by induction on the definition of the rules of P_1. If the fact $is(a, l_C)$. is present in $P_1(\Gamma)$, then it has been added by the labelling and $C(a) \in \Gamma$. Thus in every lp-interpretation I' for Γ it holds that $I' \models K$. Otherwise, $is(a, l_C) \in I$ has been added in the application of a rule in $P_1(\Gamma)$ relative to a complex concept B or to a formula $\forall_G C$.

If $B = C \sqcap D$, then $is(a, L_{C \sqcap D}) \in I$ and, by induction, there exists an lp-intepretation I' of Γ such that $I' \models C \sqcap D(a)$. By definition, this implies $I' \models C(a)$.

If $B = \exists R.C$, then $rel(c, R, a) \in I$ and $is(c, l_{\exists R.C}) \in I$. The first implies that $R(c, a) \in \mathcal{R}$. The second, by induction, implies that there exists an lp-interpretation I' of Γ such that $I' \models \exists R.C(c)$ with $I' \models R(c, a)$. For the definition of lp-interpretations, one of the admissible interpretations I'' for $\exists R.C(c)$ is the one in which $I'' \models R(c, a)$ and $I'' \models C(a)$. Thus, if $I' = I''$, then $I' \models C(a)$.

If $is(a, l_C) \in I$ has been added by a rule relative to $\forall_G C$, by definition $I \models is(a, G)$ and $a \in \text{DOM}(G)$ with $\forall_G C \in \Gamma$. If I' is an lp-interpretation of Γ, then $I' \models \forall_G C$ and thus, for every $d \in \text{DOM}(G)$, $I' \models C(d)$. Hence, $I' \models C(a)$. □

Given an lp-interpretation I for $P(\Gamma)$ and a closed formula $K \in \mathcal{L}_\mathcal{N}$, we define $IT(K, I)$ as the set of information terms "returned" by P_2 as follows:

- If $K = C(a)$, let $IT(C(a), I) = \{\alpha \mid I \models is\_it(\alpha, a, l_C)\}$.
- If $K = R(a, b)$, let $IT(R(a, b), I) = \texttt{tt}$ iff $I \models rel\_it(\texttt{tt}, a, R, b)$.
- If $K = \forall_G C$, let $D = \{c \mid I \models isa\_it([c, \alpha], G, l_C)\}$:
  - if $D = \text{DOM}(G)$, let:

    $$IT(\forall_G C, I) = \{\phi \mid \phi(c) = \alpha \text{ for } c \in \text{DOM}(G) \text{ and } I \models isa\_it([c, \alpha], G, l_C)\};$$

  - otherwise, $IT(\forall_G C, I) = \emptyset$.

Then, with respect to the reconstruction of information terms we can show, using the definition of P_2 and $\texttt{ans}(\langle\eta\rangle K)$:

Theorem 6. *Let $\Gamma \subseteq \mathcal{L}_\mathcal{N}$ be an input set of closed formulas with $\mathcal{R} \subseteq \Gamma$, K a closed formula of $\mathcal{L}_\mathcal{N}$, and I be the (unique) answer set for $P(\Gamma)$.*

If $\eta \in IT(K, I)$, then there exists an lp-interpetation I' for Γ s.t. $\texttt{ans}(\langle\eta\rangle K) \subseteq I'$

Proof. We can show by induction on the structure of K that $\texttt{ans}(\langle\eta\rangle K) \subseteq I'$.

If $K = R(a, b)$, then $P_2(R(a, b)) = \{rel\_it(\texttt{tt}, a, R, b) \leftarrow rel(a, R, b).\}$ and $I \models rel\_it(\texttt{tt}, a, R, b)$ only if $I \models rel(a, R, b)$., which by P_1 implies that $R(a, b) \in \mathcal{R}$ with $\mathcal{R} \subseteq \Gamma$. Thus for every lp-interpretation s.t. $I' \models \Gamma$, also $\{K\} = \texttt{ans}(\langle\eta\rangle K) \subseteq I'$.

If $K = A(c)$ with $A \in \text{NC} \cup \text{NG}$ and $\texttt{tt} \in IT(K, I)$, by the definition of P_2, we have that $I \models is(c, A)$. Thus for Lemma 1 and the definition of P_1, $I \models is(c, l_A)$ and there exists an lp-interpretation I' for Γ s.t. $A(c) \in I'$. This implies that $\texttt{ans}(\langle\eta\rangle K) \subseteq I'$.

If $K = C \sqcap D(c)$ and $(\alpha, \beta) \in IT(K, I)$, by the definition of P_2, we have that $I \models is\_it(\alpha, c, l_C)$ and $I \models is\_it(\beta, c, l_D)$. Thus, for induction hypothesis, there exist I_1, I_2 interpretations of Γ s.t. $\texttt{ans}(\langle\alpha\rangle C(c)) \subseteq I_1$ and $\texttt{ans}(\langle\beta\rangle D(c)) \subseteq I_2$. Thus $\texttt{ans}(\langle(\alpha, \beta)\rangle K) \subseteq I_1 \cup I_2$.

If $K = \exists R.D(c)$ and $(d, \alpha) \in IT(K, I)$, by the definition of P_2, we have that $rel\_it(\mathtt{tt}, a, R, d), is\_it(\alpha, d, l_C) \in I$. By the definition of P_2, the first implies that $R(a, d) \in \mathcal{R}$ and, by induction hypothesis, there exists an lp-interpretation for Γ $\mathtt{ans}(\langle\alpha\rangle D(c)) \subseteq I'$. This implies that $\mathtt{ans}(\langle(d, \alpha)\rangle K) \subseteq I'$.

If $K = \forall_G C$ and $\phi \in IT(K, I)$, by P_2 we have that, for every $c \in \text{DOM}(G)$, $\phi(c) = \alpha$ and $isa\_it([c, \alpha], G, l_C) \in I$. This means that $is\_it(\alpha, c, l_C) \in I$. By induction hypothesis, there exists I_c lp-interpretation of Γ s.t. $\mathtt{ans}(\langle\alpha\rangle C(c)) \subseteq I_c$. Thus, considering the lp-interpretation $I_G = \bigcup_{c \in \text{DOM}(G)} I_c$, we have $\mathtt{ans}(\langle\phi\rangle K) \subseteq I_G$. □

We remark that an answer set for $P_1(\Gamma)$ does *not* coincides, in general, to an answer set for Γ as defined in Section 3. Basically, the difference lies in the generation of fillers for existential formulas: for a formula $\exists R.C(a)$, while an answer set for Γ "chooses" one of the possible $b \in \mathcal{N}$ such that $R(a, b)$ and $C(b)$ are verified, the approach of P_1 is to generate in its model all such possible alternatives of the fillers. If we want to be more faithful to the first interpretation, one option would be to generate a model for each filler alternative by adding in the translation of existential formulas $K = \exists R.C(a)$ disjunctive rules of the kind:

$$fills_K(x) \vee \neg fills_K(x) \leftarrow rel(a, R, x).$$
$$is(x, l_C) \leftarrow fills_K(x).$$
$$\leftarrow fills_K(x), fills_K(y), y \neq x.$$

It is easy to see, however, that this generation approach leads to a "combinatorial explosion" of the number of the models, as one has to consider all admitted combinations of fillers for each existential formula. Similar considerations can be given if, moreover, one does not want to restrict to a fixed input $\mathcal{R} \subseteq \mathcal{R}_\mathcal{N}$ but aims at computing all possible $b \in \mathcal{N}$ satisfying $rel(a, R, x)$. In this regard, connections of such generation with existential extensions of datalog [1] can be studied.

Note that the two rewritings P_1 and P_2 can be also used separately. For example, we might only apply the IT generating rewriting P_2 if we are interested in verifying that the input knowledge base contains all the necessary "constructive information" needed to justify (i.e. compute an information term for) every input axiom[5].

5 Asp-it prototype

The datalog translation presented in previous section has been implemented in a prototype, called Asp-it. Basically, the Asp-it prototype takes as input an OWL-EL ontology and, by using the presented datalog rewritings, outputs the ontology annotated with the information terms computed for each of its axioms.

Asp-it is implemented as a Java-based command line application: it accepts as input an OWL-EL ontology (using the presented $\mathcal{EL}$ fragment), which corresponds to the input set of formulas Γ of the presented rewritings; Asp-it produces as output the same OWL ontology in which all logical axioms are annotated with the derived information terms using an OWL annotation property `elc:hasIT`; optionally, Asp-it can save to file the datalog rewriting used in the computation.

[5] This basically corresponds to the role of the GENIT algorithm in [7].

Thus, information terms in the output ontology take the form of string RDF literals and are structured as the list terms generated by the P_2 rewriting (like, e.g. `"[meat,[red,tt]]"`). The newly added `elc:hasIT` annotation property is defined in a support schema file imported in the rewriting.

The structure of Asp-it has been realized mostly around our previous work on CKR*ew*[6], a datalog rewriter for Contextualized Knowledge Repositories [4]. The loading and saving of OWL ontology files is managed using the *OWL API*[7]. Computation of models for the datalog rewritings is managed using an external call to the *DLV solver*[8], by means of the *DLVWrapper* Java library [25].

The information terms generation process of Asp-it is shown in Figure 1. First of all,

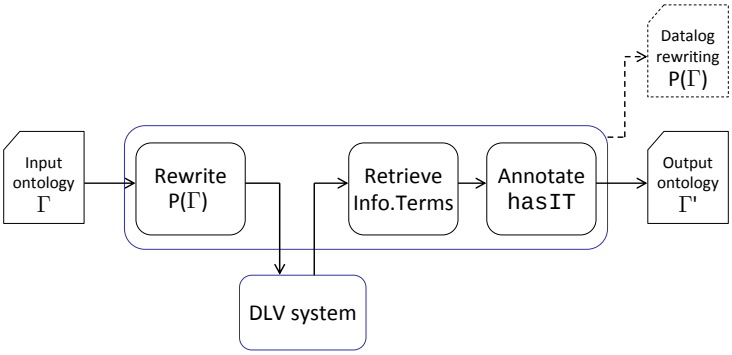

Fig. 1. Asp-it information terms generation process

the input OWL-EL ontology Γ is loaded using the methods of the OWL API. Then the rewriting takes place, building up the datalog program associated to the input knowledge base: for each of the OWL logical axioms in Γ, their structure is recursively traversed by the rewriting methods and the corresponding rules of P_1 and P_2 are added to the program. Next, the program is submitted to the DLV solver to compute its answer sets: the output models are then filtered to retrieve the terms corresponding to information terms of the input axioms (equivalent to the $IT(K, I)$ set from Section 4). The information terms `elc:hasIT` annotations are then written to the output ontology by means of the OWL API. Finally, the output ontology Γ' (and, if requested, the computed datalog program) is saved to file.

Asp-it is distributed as an open source software at `https://github.com/dkmfbk/asp-it`. The latest binary release of Asp-it can be downloaded at `https://dkm.fbk.eu/resources/asp-it/asp-it.zip`. The binary package contains (in the folder `demo`) example files implementing the "food and wines" running example.

[6] `http://ckrew.fbk.eu/`
[7] `http://owlcs.github.io/owlapi/`
[8] `http://www.dlvsystem.com/dlv/`

6 Conclusions and future works

In this paper, our interest was to demonstrate a practical approach to the realization of semantics for constructive description logics (and the application of their formal properties) on the base of Semantic Web languages and tools. We first introduced a minimal constructive description logic $\mathcal{EL}$c based on the language of $\mathcal{EL}$. Then, we extended to this interpretation results linking its information terms semantics to answer set semantics: on these bases, we proposed a datalog rewriting aimed at the computation of information terms of an input $\mathcal{EL}$c knowledge base. Finally, we have developed an open source tool implementing this computation, using well-known tools for the management of OWL ontologies and answer set programming. We remark that, while demonstrative from an applicative point of view, this exercise also lead to a first study of the relations of constructive semantics for description logics with answer set programming.

Of course, the presented work and prototype only represent a first step towards the use of constructive description logics in practical applications on Semantic Web data. As noted in previous sections, one fundamental direction would be to develop and integrate in current work procedures that are able to manipulate the computed information terms. In this regard, for example, it will be interesting to study the applicability of this work in conjunction to the Semantic Web service composition calculus based on $\mathcal{BCDL}$ presented in [5]. From a formal point of view, a prosecution of this work should involve the study of the expandability of the presented results and rewritings to more expressive description logics. For example, one direction would be to extend the results to the language of $\mathcal{ALC}$, thus aiming at the full $\mathcal{BCDL}$ logic. Another direction would be to broaden the language in the $\mathcal{EL}$ family towards $\mathcal{SROEL}$, corresponding to the full language of OWL-EL.

References

1. Alviano, M., Faber, W., Leone, N., Manna, M.: Disjunctive datalog with existential quantifiers: Semantics, decidability, and complexity issues. TPLP 12(4-5), 701–718 (2012)
2. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: IJCAI-03. pp. 325–330. Morgan Kaufmann (2003)
3. Bozzato, L., Ferrari, M., Fiorentini, C., Fiorino, G.: A constructive semantics for $\mathcal{ALC}$. In: DL2007. CEUR-WP, vol. 250, pp. 219–226. CEUR-WS.org (2007)
4. Bozzato, L., Eiter, T., Serafini, L.: Contextualized Knowledge Repositories with Justifiable Exceptions. In: DL2014. CEUR-WP, vol. 1193, pp. 112–123. CEUR-WS.org (2014)
5. Bozzato, L., Ferrari, M.: Composition of semantic web services in a constructive description logic. In: RR2010. Lecture Notes in Computer Science, vol. 6333, pp. 223–226. Springer (2010)
6. Bozzato, L., Ferrari, M., Fiorentini, C., Fiorino, G.: A decidable constructive description logic. In: JELIA 2010. Lecture Notes in Computer Science, vol. 6341, pp. 51–63. Springer (2010)
7. Bozzato, L., Ferrari, M., Villa, P.: Actions over a constructive semantics for description logics. Fundam. Inform. 96(3), 253–269 (2009)
8. Brachman, R., McGuinness, D., Patel-Schneider, P., Resnick, L., Borgida, A.: Living with CLASSIC: When and how to use a KL-ONE-like language. In: Principles of Semantic Networks. pp. 401–456. Morgan Kaufmann (1991)

9. Calimeri, F., Cozza, S., Ianni, G., Leone, N.: Enhancing ASP by functions: Decidable classes and implementation techniques. In: AAAI 2010. AAAI Press (2010)
10. Eiter, T., Ianni, G., Krennwallner, T.: Answer set programming: A primer. In: Reasoning Web 2009. Lecture Notes in Computer Science, vol. 5689, pp. 40–110. Springer (2009)
11. Ferrari, M., Fiorentini, C., Fiorino, G.: $\mathcal{BCDL}$: basic constructive description logic. J. of Automated Reasoning 44(4), 371–399 (2010)
12. Ferrari, M., Fiorentini, C., Momigliano, A., Ornaghi, M.: Snapshot generation in a constructive object-oriented modeling language. In: LOPSTR 2007, Selected Papers. Lecture Notes in Computer Science, vol. 4915, pp. 169–184. Springer (2008)
13. Fiorentini, C., Ornaghi, M.: Answer set semantics vs. information term semantics. In: ASP2007: Answer Set Programming, Advances in Theory and Implementation (2007)
14. Fiorentini, C., Momigliano, A., Ornaghi, M., Poernomo, I.: A constructive approach to testing model transformations. In: ICMT 2010. Lecture Notes in Computer Science, vol. 6142, pp. 77–92. Springer (2010)
15. Haeusler, E.H., de Paiva, V., Rademaker, A.: Intuitionistic description logic and legal reasoning. In: DEXA 2011 Workshops. pp. 345–349. IEEE Computer Society (2011)
16. Hilia, M., Chibani, A., Djouani, K., Amirat, Y.: Semantic service composition framework for multidomain ubiquitous computing applications. In: ICSOC 2012. Lecture Notes in Computer Science, vol. 7636, pp. 450–467. Springer (2012)
17. Lifschitz, V., Tang, L.R., Turner, H.: Nested expressions in logic programs. Ann. Math. Artif. Intell. 25(3-4), 369–389 (1999)
18. Mendler, M., Scheele, S.: Towards a type system for semantic streams. In: SR2009 - Stream Reasoning Workshop (ESWC 2009). CEUR-WP, vol. 466. CEUR-WS.org (2009)
19. Mendler, M., Scheele, S.: Towards Constructive DL for Abstraction and Refinement. J. Autom. Reasoning 44(3), 207–243 (2010)
20. Miglioli, P., Moscato, U., Ornaghi, M., Usberti, G.: A constructivism based on classical truth. Notre Dame Journal of Formal Logic 30(1), 67–90 (1989)
21. Motik, B., Fokoue, A., Horrocks, I., Wu, Z., Lutz, C., Grau, B.C.: OWL 2 Web Ontology Language Profiles. W3C recommendation, W3C (Oct 2009), http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/
22. Odintsov, S., Wansing, H.: Inconsistency-tolerant description logic. Part II: A tableau algorithm for $CALC^C$. J. of Applied Logic 6(3), 343–360 (2008)
23. Ornaghi, M., Benini, M., Ferrari, M., Fiorentini, C., Momigliano, A.: A Constructive Modeling Language for Object Oriented Information Systems. Electr. Notes Theor. Comput. Sci. 153(1), 55–75 (2006)
24. Pearce, D., Sarsakov, V., Schaub, T., Tompits, H., Woltran, S.: A polynomial translation of logic programs with nested expressions into disjunctive logic programs: Preliminary report. In: ICLP 2002. Lecture Notes in Computer Science, vol. 2401, pp. 405–420. Springer (2002)
25. Ricca, F.: The DLV java wrapper. In: AGP-2003. pp. 263–274 (2003)
26. Troelstra, A.S.: From constructivism to computer science. Theor. Comput. Sci. 211(1-2), 233–252 (1999)