Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 256 (2009) 19–31

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

How to Produce Information About a Given Entity Using Automated Deduction Methods

Robert Demolombe[1] Luis Fariñas del Cerro[2]

Institut de Recherche en Informatique de Toulouse
France

Abstract

The standard method to retrieve information can be formally defined as follows. To ask a query, one gives the properties of the entities to be retrieved, and the answer is the set of all the entities that satisfy the query. Another method, is to ask the overall information about a given entity, and the answer is the corresponding information. An example of the first kind of query is: "*what are the drugs that contain a given molecule?*", an example of the second kind is: "*what are the properties of a given drug?*".
This latter method has deserved very few researches though it has great potential practical applications. However, it raises many non trivial issues. The first one is to find a precise definition of the fact that a piece of information "*is about*" a given entity. We recall the formal definition that have been proposed in formal classical logic, and the main properties that follow from this definition. The second one is to adapt existing automated deduction methods to compute this new kind of answer, using either deduction or abduction techniques.
Finally, we present potential extensions to our definition and guidelines for automated deduction strategies.

Keywords: Resolution, Abduction, Information Retrieval.

1 Introduction

Since the beginning of the seventies many research works have been devoted to define theoretical foundations and to develop tools to retrieve data in the form of so called Relational Databases. In these databases the information is formally organized with a rather limited number of predicates whose extensions may be very large. This approach requires to define a priori the predicates and to store the information in this predefined form. This is not a constraint for applications in the field of management because the predicates are rarely changed. These predicates are known by users and they are used to express queries in formal languages like SQL. The retrieved information must satisfy exactly these formulas.

[1] Email: robert.demolombe@orange.fr
[2] Email: luis.farinas@irit.fr

doi:10.1016/j.entcs.2009.11.003

Now, the information which can be accessed via Internet is not formally structured in that way. Rather, in most cases, it is represented in natural languages and queries are no more expressed in a formal language but by keywords. These keywords, from a logical point of view, may be constant symbols or predicate symbols. Then, it is important to extend automated deduction methods to retrieve the overall information about one or several given entities.

The work presented in this paper tries to propose theoretical foundations and practical methods to design tools that allow to retrieve the overall information about a given entity. It is based on previous works by philosophers and logicians [13,16,8,1,15], and also on works about automated deduction [18,2,14,12,10,11]. Our main objective in defining these practical methods was to define a bridge between the theoretical definitions of relevant information and an automated resolution strategy which is as efficient as linear resolution. We are perfectly aware that more efficient strategies could be designed.

In the following section 2 we have classified the different approaches to store and retrieve information using examples. In section 3 are presented the theoretical foundations that allow to characterize the information which is about a given entity. The next section is devoted to the presentation of automated deduction methods to retrieve this information. Finally, in the last section is presented a set of open questions that deserve further researches.

2 Selecting information about a given entity: a new perspective for information retrieval

The **standard approach** to retrieve information from a knowledge base (KB) is to request all the entities that satisfy given properties. If this approach is formalized in classical logic, KB is represented by a set of formulas (many of them represent atomic facts), and the properties are represented by a formula of the form $F(x)$ [3].

The answer obtained by **deduction** is the set of entities a defined by: $\{ a : \vdash KB \rightarrow F(a)\}$.

For instance, in a knowledge base about car accidents we may have the information that: if someone is driving and is drunk then he may have an accident, if someone is driving and the road is icy then he may have an accident, Smith and Dupont are driving, and Smith may have an accident. This information is formally represented by:
$KB = \{\forall x(drunk(x) \land driving(x) \rightarrow accident(x)), \forall x(icy \land driving(x) \rightarrow accident(x)), driving(Smith), driving(Dupont), accident(Smith)\}$

Then, the query is represented by:
$F(x) = driving(x) \land accident(x)$,
and the answer is: $\{Smith\}$.

If we want to know whether Dupont may have an accident the query is represented by: $F = accident(Dupont)$,

[3] In general the formula may have zero or several free variables.

and the answer is: *unknown*, because from KB we cannot infer that this fact is true, and we cannot infer that it is false.

Now, if we want to know in which circumstances Dupont may have an accident we have to ask another kind of query that gives in the answer these circumstances. In formal terms this answer is obtained reasoning by **abduction**, that is by looking for the minimal assumptions that must be added to KB to derive the fact that Dupont may have an accident.

The general formalization of these answers is defined as follows. The query is represented by a formula of the form: F, and the answer is the set of formulas H which is minimal, in some formal sense like subsumption, such that:
$\{\ H\ \ :\ \ \vdash KB \rightarrow (H \rightarrow F)\}$.

For instance, for the query: $F\ =\ accident(Dupont)$,
the answer is: $\{drunk(Dupont), icy\}$, because, if these assumptions are added to KB it is possible to derive F.

A **new approach** to retrieve information is to focus on the entities rather than on their properties. That leads to give the priority to the notion of aboutness.

According to this approach, a query is defined by the entity c we are interested in, and the answer is defined by the set of formulas F that are about this entity c and that can be derived by **deduction** from KB. That is:
$\{\ F\ \ :\ \ \vdash KB \rightarrow F\ \ and\ \ About(F, c)\}$,
where $About(F, c)$ means that the formula F is about the entity represented by the constant c.

For instance, if the query is to know everything about Dupont that can be inferred from KB, the answer is:
$\{drunk(Dupont) \rightarrow accident(Dupont), icy \rightarrow accident(Dupont), driving(Dupont)\}$.

In the case of answers defined by **abduction**, the answer is defined in a similar way as in the standard approach, except that we only want to get the assumptions that are about a given entity c.

Then, the answer is formally defined by:
$\{\ H\ \ :\ \ \vdash KB \rightarrow (H \rightarrow F)\ \ and\ \ About(H, c)\}$,
where H is minimal in the same sense as above.

For instance, if the query is: $F\ =\ accident(Dupont)$,
the answer about Dupont is: $\{drunk(Dupont)\}$, and the answer *icy* has been removed because this fact is not about Dupont.

In another context, if KB contains information about patients and diseases, if we want to know the assumptions that are possible explanations for the fact that Dupont has a given disease, it could be interesting to focus on the explanations that are about a given drug.

To sum up, in the new approach, information that can be retrieved either by deduction or by abduction is focused on information about a given entity. Then, we need a clear definition of the fact that a piece of information is about a given entity.

3 Formal characterization of sentences that inform about a given entity

In [6] sentences that are about a given entity in the semantics have been defined. A first order predicate calculus language L_c is defined, where c is some given constant symbol. Neither function symbols [4] nor the equality predicate are allowed in the language. Terms are either variable symbols or constant symbols.

Definition 3.1 Syntactical definition of language L_c.

L_c is defined by the following rules.

(i) If p is an n-ary predicate and t is an n-tuple of terms, then $p(t) \in L_c$.

(ii) If $F \in L_c$ and $G \in L_c$, then $(\neg F) \in L_c$ and $(F \vee G) \in L_c$.

(iii) if $F \in L_c$, then $(\exists x F) \in L_c$ and $(\exists x \neq c\ F) \in L_c$ [5].

(iv) All the sentences in L_c are defined by rules 1, 2 and 3.

As usual we adopt the following notations: $p \wedge q \overset{\text{def}}{=} \neg((\neg p) \vee (\neg q))$, $p \rightarrow q \overset{\text{def}}{=} (\neg p) \vee q$, $p \leftrightarrow q \overset{\text{def}}{=} (p \rightarrow q) \wedge (q \rightarrow p)$ and $\forall x \neq c\ F \overset{\text{def}}{=} \neg(\exists x \neq c\ \neg F)$.

Quantifiers of the form $\forall x \neq c$ and $\exists x \neq c$ are called “restricted quantifiers”.

Definition 3.2 Interpretation.

Let's consider a language L_c as defined in Definition 3.1. An interpretation M of L_c is a tuple $M =< D, i >$ such that

- D is a non empty set of individuals,
- i is a function that assigns
  - · to each predicate symbol of arity n a subset of D^n,
  - · to each variable symbol an element of D,
  - · to each constant symbol an element of D,

In the following D will be called the domain of the interpretation, and i will be called the interpretation function, or, for short, the interpretation.

Notation: the domain of M will be denoted by D_M and the interpretation function of M will be denoted by i_M.

Definition 3.3 Satisfiability conditions.

Let M be an interpretation of the language L_c. The fact that a formula F of L_c is true in M is denoted by $M \models F$, and is inductively defined as follows.

- If F is an atomic sentence of the form $p(t)$, where t is a tuple of constant symbols or variable symbols, we have $M \models F$ iff $i_M(t) \in i_M(p)$.

[4] The absence of function symbols is not a strong limitation of the expressive power of the language. Indeed, if there are function symbols in a language we can define another language where function symbols are replaced by predicates, and formulas of the former language can be easily translated into latter one. For instance, the formula: $driving(father(Smith))$, which means that Smith's father is driving, can be translated into: $\exists x(driving(x) \wedge father(x, Smith))$, where $father(x, y)$ is the predicate associated to the function symbol $father(x)$.

[5] Here $x \neq c$ is used as a notation to denote restricted quantifiers, it is not taken as a sentence with an occurrence of equality predicate.

- $M \models \neg F$ and $M \models F \vee G$ are defined from $M \models F$ and $M \models G$ as usual.
- $M \models \exists x F$ iff there exists an interpretation $M_{x/d}$ that only differs from M by the interpretation of variable symbol x, such that $i_{M_{x/d}}(x)$ is the element d of $D_{M_{x/d}}$ and $M_{x/d} \models F$.
- $M \models \exists x \neq c\ F$ iff there exists an interpretation $M_{x/d}$ that only differs from M by the interpretation of variable symbol x, such that $i_{M_{x/d}}(x)$ is the element d of $D_{M_{x/d}}$ and $i_{M_{x/d}}(c)$ is not d and $M_{x/d} \models F$.

A formula F is a valid formula iff for every interpretation M we have $M \models F$. This is denoted by $\models F$.

Definition 3.4 Variants of an interpretation with regard to an entity.

Let L_c be a language as defined in Definition 3.1. We call variants of M with regard to c the set M^c of interpretations M' defined from M in the following way.

- $D_{M'} = D_M$
- $i_{M'} = i_M$ for every variable symbol and constant symbol,
- $i_{M'}$ is defined from i_M for each predicate symbol as follows: if p is a predicate symbol of arity n
  - · if t is an n-tuple of terms of language L_c that contain no occurrence of the constant symbol c, then $i_{M'}(t) \in i_{M'}(p)$ iff $i_M(t) \in i_M(p)$,
  - · if an element $< d_1, \dots, d_n >$ of D^n is such that, for every j in [1,n], $d_j \neq i_M(c)$, then $< d_1, \dots, d_n > \in i_{M'}(p)$ iff $< d_1, \dots, d_n > \in i_M(p)$.

Intuitively, a variant of a given interpretation for a given constant symbol differs only by the truth assignment of atomic formulas where this constant appears as an argument of the atomic formula.

The set of variants M' of the interpretation M with regard to an entity named with the constant symbol c is denoted by M^c. Notice that M belongs to M^c, and that, if M' belongs to M^c, M belongs to M'^c too.

Definition 3.5 Sentences that are not about an entity.

Let F be a sentence of language L_c. We say that F is not about an entity named by the constant symbol c iff for every interpretation M, we have $M \models F$ iff for every interpretation M' in M^c we have $M' \models F$.

The fact that F is not about entity c is denoted by $NAbout(F, c)$. In short we have:

$NAbout(F, c)$ *holds* iff $\forall M (M \models F$ iff $\forall M' \in M^c\ M' \models F)$

We say that a **formula** F **is about the entity** c, if it is not the case that $NAbout(F, c)$. This fact is denoted by $About(F, c)$. In short terms we have:

$About(F, c)$ *holds* iff $\exists M (\exists M' \in M^c (M \models F \textit{ and } M' \not\models F))$

It can be checked that, according to Definition 3.5, sentence $p(a)$ is not about the entity c, and that sentences $p(c)$ and $\forall x p(x)$ are about entity c.

The most important properties about the notion of aboutness that have been proved in [6] are listed below.

• $\models F \leftrightarrow G \Rightarrow (NAbout(F, c) \Leftrightarrow NAbout(G, c))$
•$NAbout(F, c) \Leftrightarrow NAbout(\neg F, c)$
•$NAbout(F, c)$ and $NAbout(G, c) \Rightarrow NAbout(F \vee G, c)$
•$NAbout(F, c)$ and $NAbout(G, c) \Rightarrow NAbout(F \wedge G, c)$
• $\models F \leftrightarrow G \Rightarrow (About(F, c) \Leftrightarrow About(G, c))$
•$About(F \vee G, c) \Rightarrow About(F, c)$ or $About(G, c)$
•$About(F \wedge G, c) \Rightarrow About(F, c)$ or $About(G, c)$

• $\models F \rightarrow G \not\Rightarrow (NAbout(F, c) \Rightarrow NAbout(G, c))$
• $\models F \rightarrow G \not\Rightarrow (NAbout(G, c) \Rightarrow NAbout(F, c))$
• $\models F \rightarrow G \not\Rightarrow (About(F, c) \Rightarrow About(G, c))$
• $\models F \rightarrow G \not\Rightarrow (About(G, c) \Rightarrow About(F, c))$

In [6] a syntactical characterization of a large subset of formulas that are not about c has been proposed.

4 Automated deduction methods to retrieve information about a given entity

To consider automated deduction tools to retrieve information we define automated deduction methods based on Resolution [17]. These methods require to formally represent sentences in clausal form. Clausal form restrict slightly the expressive power of the language but for a large number of real applications this is not a practical limitation. Then, in the following it will be assumed that the overall information is represented in clausal form.

In the next subsections we present first a classical abduction method, second a deduction method to retrieve information about a given entity, and in the third subsection we extend the classical abduction method to retrieve information about a given entity.

4.1 Classical abduction

Definition 4.1 Clause.

A clause is a first order formula of the form: $L_1 \vee L_2 \vee \ldots \vee L_n$, where each L_i is a literal.

A literal is an atomic formula or the negation of an atomic formula.

The free variables of a clause are implicitly universally quantified.

Many strategies have been defined to compute answers in the standard approach when the answers are obtained by deduction.

When the answers are obtained by abduction the number of strategies is rather limited. There is, for instance, the SOL-resolution defined by Inoue in [9,10,11], and the l-inference defined by Demolombe and Fariñas del Cerro in [5].

We briefly resume below what is the l-inference.

Definition 4.2 l-Clause.

A clause C is an l-clause iff there is an atomic formula in C with l as predicate symbol.

Definition 4.3 l-Inference.

A resolvent C from C_1 and C_2 by Resolution Principle is obtained by an l-inference iff C is an l-clause.

Remark. If C is an l-clause one of the parent clauses C_1 or C_2 is an l-clause.

Definition 4.4 l-Deduction. Let S be a set of clauses. An l-deduction of C_n from S is a finite sequence $C_0 \dots C_n$ of clauses such that : each C_i is either a clause in S or there are C_{i_1} and C_{i_2} in the l-deduction such that $i_1 < i$, $i_2 < i$ and C_i is the l-resolvent of C_{i_1} and C_{i_2}.

Definition 4.5 R-Deduction. An R-deduction of C_n from S is a finite sequence $C_0 \dots C_n$ of clauses such that : each C_i is either in S or there are C_{i_1} and C_{i_2} in the R-deduction such that $i_1 < i$, $i_2 < i$ and C_i is the resolvent by Resolution Principle of C_{i_1} and C_{i_2}.

Theorem 4.6 *Let S be a set of clauses, if C is a clause derivable from S, there is a clause C', subsuming C, such that C' is derivable from S by an R-deduction.*

The Theorem 4.6 has been proved by Lee in [14].

Theorem 4.7 *Let S be a set of clauses and l a given predicate. If there is an R-deduction of the l-clause C, then there is an l-deduction of C.*

The proof of Theorem 4.7 can be found in [5].[6]

4.2 Deduction restricted to an entity

We present now an automated deduction method to derive answers which are about a given entity.

This method should have good performances because it derives only clauses which are about a given entity represented by the constant c, and they are obtained by linear deductions.

Definition 4.8 c-Clause.

A c-Clause is a clause such that there is a literal in the clause and a term in this literal which is either a free variable or the constant symbol c.

In the following it is assumed that clauses are not tautologies. It can be easily checked that c-clauses are formulas about c.

Definition 4.9 c-Inference. An inference of C from C_1 and C_2 by Resolution Principle is a c-inference iff C is a c-clause.

[6] The Theorem given in [5] is more general in the sense that the l-inference is defined as an hyperesolution. That gives a deduction method which is more efficient.

Definition 4.10 c-Deduction. A c-deduction of C_n from S is a finite sequence of clauses $C_0 \dots C_n$ such that each C_i is either a clause in S or there are C_{i_1} and C_{i_2} in the c-deduction, with $i_1 < i$ and $i_2 < i$, such that C_i is obtained by a c-inference from C_{i_1} and C_{i_2}.

C_0 is called the top clause.

Theorem 4.11 *If C is obtained by a c-inference from C_1 and C_2, and C_2 is a c-clause, and C_1 is the resolvent by Resolution Principle of the two clauses E_1 and E_2, then there exists a c-inference of E_2 and C_2 whose resolvent is F, and there exists a c-inference of E_1 and F whose resolvent is C.*

Proof schema. Let L_2 be the literal in C_2 which is resolved with a literal in C_1. Without loss of generality we can assume that this literal in C_1 is an instance of a literal in E_2 which is called L'_1.

Let M_2 be the literal in E_2 which is resolved with some literal M_1 in E_1. Then, E_1 and E_2 have the following form:

$E_1 = M_1 \vee e_1$

$E_2 = M_2 \vee L'_1 \vee e_2$

Let σ_1 be the mgu of M_1 and M_2, and σ_2 be the mgu of $L'_1\sigma_1$ and L_2. then the clauses C_1, C_2 and C have the form:

$C_1 = L'_1\sigma_1 \vee e_1\sigma_1 \vee e_2\sigma_1$

$C_2 = L_2 \vee c_2$

$C = e_1\sigma_1\sigma_2 \vee e_2\sigma_1\sigma_2 \vee c_2\sigma_2$

Since $L'_1\sigma_1$ and L_2 can be unified there exists a mgu σ'_1 of L'_1 and L_2. Let F be the resolvent by Resolution Principle of E_2 and C_2. F has the form:

$F = M_2\sigma'_1 \vee e_2\sigma'_1 \vee c_2\sigma'_1$

The literals M_1 and $M_2\sigma'_1$ can be unified by the mgu σ'_2. Let C' be the resolvent by Resolution Principle of E_1 and F. Then, C' has the form:

$C' = e_1\sigma'_2 \vee e_2\sigma'_1\sigma'_2 \vee c_2\sigma'_1\sigma'_2$

It can be proved that F and C' are c-clauses. Then, they are obtained by a c-inference.

It can also be proved that the clause C' is the same clause as C. □

Theorem 4.12 *If there is an R-deduction of C from S and the clause C_2 such that C is a c-clause and C is the c-resolvent of the clauses C_1 and C_2, then there exists a c-deduction of C from S and C_2, such that C_2 is the top clause.*

Proof. The proof is by induction on the length n of the R-deduction of C_1.

Induction hypothesis. If there is an R-deduction of C from S and the clause C_2 such that: C is a c-clause, C is the c-resolvent of the clauses C_1 and C_2 and the length of the R-deduction of C_1 is $\leq n$, then there exists a c-deduction of C from S and C_2 such that C_2 is the top clause.

For $n = 0$ the induction hypothesis is true (trivial).

Assumption: there is an R-deduction of C from S and the clause C_2 such that: C is a c-clause, C is the c-resolvent of the clauses C_1 and C_2 and the length of the

R-deduction of C_1 is $n+1$.

Let E_1 and E_2 be the two clauses whose inference by Resolution principle is C_1. The length of their R-deductions is $\leq n$.

From Theorem 4.11 the deduction of C_1 from E_1 and E_2, and of C from C_2 can be transformed into a deduction of F from E_2 and C_2 and of C from E_1 and F, where C and F are obtained by c-inference.

From the induction hypothesis, there exists a c-deduction δ_1 of F from S and C_2. From the induction hypothesis we can also infer that there exists a c-inference δ_2 of C from S and F.

Therefore the sequence $\delta_1\delta_2$ is a c-deduction of C from S and C_2. □

Theorem 4.13 *If there is an R-deduction of C from S such that C is a c-clause, then there exists a c-deduction of C from S.*

Proof. The proof is by induction on the length n of the R-deduction of C.

Induction hypothesis. If there is an R-deduction of length $\leq n$ of C from S such that C is a c-clause, then there exists a c-deduction of C from S.

Assumption. There is an R-deduction of length $n+1$ of C from S such that C is a c-clause.

Let C_1 and C_2 be the clauses whose inference by Resolution principle is C. Therefore either C_1 or C_2 is a c-clause. Let C_2 be that c-clause. Since the length of the R-deduction of C_2 is $\leq n$, by induction hypothesis there exists a c-deduction δ_1 of C_2 from S.

From Theorem 4.12 there exists a c-deduction of C δ_2 from S and C_2.

Therefore the sequence $\delta_1\delta_2$ is a c-deduction of C from S. □

4.3 Abduction restricted to an entity

To only retrieve answers that are assumptions about an entity we have to design an abduction methods which is more specific than the SOL-deduction [7] or the L-deduction.

Definition 4.14 lc-Clause. A clause is an lc-clause iff it is both an l-clause and a c-clause.

Definition 4.15 lc-Inference. An lc-inference is an inference which is both an l-inference and a c-inference.

It is worth noting that there are lc-clauses that are R-deductibles from a given set of clauses S, such that it does not exist an R-deduction in which each resolvent is an lc-clause. Let's consider, for instance, the set of clauses: $S = \{\neg q \vee t,\ q \vee l,\ \neg t \vee p(c)\}$ and the lc-clause: $l \vee p(c)$.

Definition 4.16 lc-Deduction. An lc-deduction of C_n from S is a finite sequence of clauses $C_0 \dots C_n$ such that there exists i, $0 < i < n$, such that the sequence

[7] The SOL-deduction [9] cannot be directly applied because the property of being a c-clause does not define a stable production field.

$C_0 \dots C_i$ is an l-deduction, and the sequence $C_{i+1} \dots C_n$ is a c-deduction, and C_n is an lc-clause.

Theorem 4.17 *If C is obtained by an lc-inference from C_1 and C_2, and C_1 is a c-clause and C_2 is an l-clause, and C_1 is the resolvent by Resolution Principle of the two clauses E_1 and E_2, then there exists an l-inference of E_2 and C_2 whose resolvent is F, and there exists a c-inference of E_1 and F whose resolvent is C.*

Proof schema. The proof is very close to the proof of Theorem 4.11. □

Theorem 4.18 *If there is an R-deduction of C from S such that C is the resolvent of C_1 and C_2, and:*

- *C is an lc-clause, C_1 is a c-clause and C_2 is an l-clause,*
- *there exists a c-deduction of C_1 from S,*
- *there exists an l-deduction of C_2 from S,*

then there exists an lc-deduction of C from S.

Proof schema. The proof is by induction on the length of the c-deduction of C_1.

Induction hypothesis. If there is an R-deduction of C from S such that C is the resolvent of C_1 and C_2, and:

- C is an lc-clause, C_1 is a c-clause and C_2 is an l-clause,
- C_1 is obtained by a c-deduction from S whose length is $\leq n$,
- there exists an l-deduction of C_2 from S

then there exists an lc-deduction of C from S.

Assumption. There is an R-deduction of C from S such that C is the resolvent of C_1 and C_2, and:

- C is an lc-clause, C_1 is a c-clause and C_2 is an l-clause,
- C_1 is obtained by a c-deduction from S whose length is $n+1$,
- there exists an l-deduction of C_2 from S

Let E_1 and E_2 be the two clauses such that C_1 is their resolvent by c-inference. Either E_1 or E_2 can be resolved with the clause C_2 (see the proof of Theorem 4.17). Without lost of generality it can be assumed that this clause is E_2.

From Theorem 4.17 the R-deduction of C can be transformed as follows: an l-inference infers the clause F from E_2 and C_2, and a c-inference infers the clause C from E_1 and F. In this transformation the l-deduction of C_2 and the R-deductions of E_1 and E_2 remain unchanged.

Since F is an l-clause, from Theorem 4.7 there exists an l-deduction δ_1 of F from S.

Since C is a c-clause, either E_1 or F is a c-clause.

Case 1. F is a c-clause. From Theorem 4.12 (replacing C_1 by E_1, and C_2 by F), there exists a c-deduction δ_2 from S and F whose top clause is F. Therefore the sequence $\delta_1\delta_2$ is an lc-deduction of C from S.

Case 2. E_1 is a c-clause. Since C_1 is obtained by a c-deduction of length $n + 1$, the c-deduction of E_1 is of length n. Then, by induction hypothesis, there exists an lc-deduction of C from S. □

Theorem 4.19 *If there is an R-deduction of C from S such that C is an lc-clause, then there exists an lc-deduction of C from S.*

Proof. The proof is by induction on the length of the c-deduction of C.

Induction hypothesis. If there is an R-deduction of C from S of length $\leq n$ such that C is an lc-clause, then there exists an lc-deduction of C from S.

Assumption. There is an R-deduction of C from S of length $n + 1$ such that C is an lc-clause.

Case 1. Either C_1 or C_2 is an lc clause.

Let's assume that C_2 is an lc -clause. The length of the R-deduction of C_2 is n. Then by induction hypothesis there is an lc-deduction of C_2 from S. Since C and C_2 are c-clauses, from Theorem 4.12 there is a c-deduction of C from S whose top clause is C_2. Therefore the lc-deduction of C_2 and this deduction make an lc-deduction of C.

Case 2. Neither C_1 nor C_2 is an lc-clause.

Therefore C_1 is a c-clause and C_2 is an l-clause (or vice versa). Then, from Theorem 4.11 there is an l-deduction of C_2 from S and from Theorem 4.13 there is a c-deduction of C_1 from S. Therefore, from Theorem 4.18 there is an lc-deduction of C from S. □

For most of the applications, as mentioned in section 2, we have to generate the set of clauses (l-clauses, c-clauses or lc-clauses) which are in the answer. The previous theorems could be applied to generate this set. In addition this set should be minimal with respect to redundancy. The design of algorithms to implement these methods deserves further researches.

5 Further works and conclusion

The lc-deduction which has been presented combines the benefits of traditional Relational Database methods and Internet information retrieval methods to retrieve information, and it does not have their drawbacks. In the former methods the expressive power is too much limited, while in the latter it is not enough specific. For example, to retrieve information about Dupont and about accidents, we may get answers of the kind: "*Smith has an accident*, or *Dupont's birthdate is July 18th 1975.*

From a technical point view there are still many problems to be solved. Some of them are listed below.

Semantic selection of c-Clauses. According to Definition 4.8 any clause which contains a free variable is a c-Clause whatever is c. For example, the clausal form of the formula: $\forall x(drunk(x) \wedge driving(x) \rightarrow accident(x))$ is a c-Clause because the universal quantifier can be instantiated by any constant c. For example, it is a clause about $Dupont$ and there is no doubt that its instance: $drunk(Dupont) \wedge driving(Dupont) \rightarrow accident(Dupont)$ is about $Dupont$. However, if we assume that $Pussy$ intuitively denotes a cat, it is quite odd to say that the instance: $drunk(Pussy) \wedge driving(Pussy) \rightarrow accident(Pussy)$, and when we want derive information about $Pussy$ it is desirable not to derive this kind of formula. A possible research direction to reach that goal could be to define a specific treatment for literals in clauses that characterizes the types of the variables. For instance, if we have in the knowledge base a formula which expresses that drivers are human being and another one that expresses that cats are not human being, from the fact that $Pussy$ is a cat we could infer that the general sentence about drink drivers is useless when we are interested in $Pussy$. From works about Domain Independent formulas [3] we know that the domains of universally quantified formulas must be explicit, else we get formulas which have no intuitive meaning. Then, the identification of literals in a clause that characterize the domain could be used for this semantic treatment.

Complete syntactical characterization. The decidability of the class of formulas that are about a given entity is an open problem. In the case of a positive answer to this problem it would be interesting to find a complete characterization of this class.

These problems should be refined to the case where aboutness is analyzed in the context of a given theory (see [6]). Indeed, a sentence may be or not about an entity depending on the fact that it is considered or not in the context of a theory. For instance, the formula: $accident(Smith) \vee accident(Dupont)$ is about Dupont in the absence of context. However, in the context of a theory where we have: $accident(Smith)$, the same formula is not about Dupont, because in this context $accident(Smith) \vee accident(Dupont)$ is always true.

Language extension to function symbols. If we accept function symbols in the language we have to extend the Definition 3.4 to functions. A possible extension to be investigated is to have a very similar treatment of predicate symbols and function symbols in this definition, that is: the interpretation of function symbols should remain unchanged if the arguments of the function are different of c or if their interpretations are not the same as the interpretation of c.

Specific problems related to Skolem functions. Skolem functions are introduced only to apply automated deduction methods and require a specific analysis. The reason is that they have a meaning which is defined by the formulas that have been transformed by skolemisation. For instance, the formula: $\exists x(driving(x) \wedge father(x, Smith))$ leads to the clauses: $driving(\alpha)$ and $father(\alpha, Smith)$. Since the formula $\exists x(driving(x) \wedge father(x, Smith))$ is about Smith, the skolem constant α implicitly refers to Smith because it refers to an entity who is Smith's father. From this simple example we can see that skolem constants, or skolem functions, cannot bee freely interpreted and require specific treatment in the definition of aboutness.

Language extension to equality. There are many applications where the background theory contains information about equality and the definition of aboutness has to be revised if equality is added to the language. For instance, if in the theory Smith's father is Dupont, in formal terms: $Dupont = father(Smith)$, the sentence $accident(father(Smith))$ is clearly about Dupont.

Equality also raises difficult problems in designing efficient automated deduction methods. Indeed, a brute force application of paramodulation rule leads to extremely expensive computations. It is possible to find heuristics to reduce the problem, but there are few works in this direction. [8]

Sentences about a given topic. It may be that the information about a given an entity is too large to be efficiently exploited. For instance, if one asks the overall information about a given drug the answer may be extremely large. In that case it can be more convenient to select the information about that drug which is about a given topic, like, for instance, toxicity. In [4] a logic has been proposed for reasoning about sentences that are about a given topic. The combination of this logic with the definition of sentences that are about a given entity should deserve further researches.

References

[1] R. Carnap. The logical syntax of language. 1937.

[2] C.L. Chang and R.C.T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.

[3] R. Demolombe. Syntactical Characterization of a Subset of Domain Independent Formulas. *Journal of ACM*, 39(1), 1982.

[4] R. Demolombe and A.J.I. Jones. On sentences of the kind "sentence "p" is about topic "t": some steps toward a formal-logical analysis. In H-J. Ohlbach and U. Reyle, editor, *Logic, Language and Reasoning. Essays in Honor of Dov Gabbay.* Kluwer Academic Press, 1999.

[5] R. Demolombe and L. Fariñas del Cerro. An Inference Rule for Hypothesis Generation. In *Proc. of International Joint Conference on Artificial Intelligence*, Sydney, 1991.

[6] R. Demolombe and L. Fariñas del Cerro. Towards a logical characterisation of sentences of the kind "sentence p is about object c". In S. Holldobler, editor, *Intellectics and Computational Logic. Papers in Honor of Wolfang Bibel.* Kluwer Academic Press, 2000.

[7] R. Demolombe and M. P. Pozos Parra. An extension of sol-resolution to theories with equality. In *Proceedings of the International Joint Conference on Automated Reasoning*, 2001.

[8] N. Goodman. About. *Mind*, LXX(277), 1961.

[9] K. Inoue. Consequence-Finding Based on Oredered Linear Resolution. In *Proc. of International Joint Conference on Artificial Intelligence*, Sydney, 1991.

[10] K. Inoue. Linear Resolution for Consequence Finding. *Artificial intelligence, an International Journal*, 56, 1992.

[11] K. Inoue. *Studies on Abductive and Nonmonotonic Reasoning.* PhD thesis, Kyoto University, 1992.

[12] R. Kowalski and D. Kuhner. Linear resolution with selection function. *Artificial Intelligence*, 2:227–260, 1971.

[13] L. Fariñas del Cerro and V. Lugardon. Sequents for dependence logic. *Logique et Analyse*, 133-134, 1994.

[14] R.C.T. Lee. *A completeness theorem and a computer program for finding theorems derivable from given axioms.* PhD thesis, Univ. of California at Berkley, 1967.

[15] D. K. Lewis. Relevant implication. *Theoria*, LIV(3), 1988.

[16] H. Putnam. Formalization of the concept "About". *Philosophy of Science*, XXV:125–130, 1958.

[17] J. A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12:23–41, 1965.

[18] J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning.* MIT Press, 2001.

[8] See, for instance, [7] where SOL deduction has been extended to equality.