

Constrained Consequence Generation

Sylvie CAZALENS Robert DEMOLOMBE
ONERA/CERT
Département Informatique
2, av. Edouard Belin
B.P. 4025
31055 Toulouse
FRANCE

1 Introduction

To introduce Constrained Consequence Generation we would like to present a new perspective of the history of Automated Deduction. This history is illustrated with a toy example, where we consider a Data and Knowledge Base represented by the following first order theory T:

$\forall x(\text{Mortal}(x) \leftarrow \text{Man}(x))$
 $\forall x(\text{Mortal}(x) \leftarrow \text{Animal}(x))$
 $\text{Man}(\text{Socrates}), \text{Man}(\text{Aristoteles}), \text{Athenian}(\text{Socrates})$

Initially, the problem of automated deduction was to find out whether a given sentence q is a logical consequence of a given theory T. Indeed, if q is viewed as a query addressed to T, the answer is “yes” if q is a logical consequence of T, “I do not know” otherwise. For instance if the query is: $\text{Mortal}(\text{Socrates})$, the answer is “yes”, and if the query is: $\text{Athenian}(\text{Aristoteles})$, the answer is “I do not know”. In general the problem can be represented by:

query: sentence q ,

answer: “yes” iff $T \vdash q$, “I do not know” iff $T \not\vdash q$.

The second step arised with Logic Programming and Deductive Databases. The problem was different in the sense that, in general, a query is represented by an open formula $q(x)$ ¹, and the answer to this query is the set of substitutions t/x such that $q(x).\{t/x\}$ is a logical consequence of T. For instance, if the query is: $\text{Mortal}(x)$, the answer is $\{\text{Socrates}/x, \text{Aristoteles}/x\}$. In general the problem can be represented by:

¹As a matter of simplification we consider only one free variable. Generalization to n free variables is rather trivial.

query: formula $q(x)$,

answer: $\{t/x : T \vdash q(x).\{t/x\}\}$.

A third step was motivated by applications like automated diagnosis, conditional answers, or truth maintenance systems and more generally, by abductive reasoning. In that case the query can be represented by an open formula, and the answer is the set of sentences $h(x)$ such that $\forall x(q(x) \leftarrow h(x))$ is a logical consequence of T . In the example, the answer to the query: $Mortal(x)$, is the set of formulas $\{Man(x), Animal(x)\}$. In that case, a general representation of the problem is:

query: formula $q(x)$,

answer: $\{h(x) : T \vdash \forall x(q(x) \leftarrow h(x))\}$.

To avoid useless answers it is also required that $\forall x(q(x) \leftarrow h(x))$ is not a tautology, and, in most applications, that it is minimal with regard to subsumption.

The problem considered in this paper is a generalization of the previous problems. In each step the problem was finding the consequences of a theory that verify some given property P . Indeed, for a given query q , we are looking for one of the following sets of consequences:

step one: $\{q : T \vdash q\}$

step two: $\{q(x).\{t/x\} : T \vdash q(x).\{t/x\}\}$

step three: $\{\forall x(q(x) \leftarrow h(x)) : T \vdash \forall x(q(x) \leftarrow h(x))\}$.

If we consider formulas in clausal form, in step one property P is just: “to be the clause q ”; in step two the property P is: “the clause is an instance of the formula $q(x)$ ”; and in step three it is: “the clause contains the literal $q(x)$ ”. Now, we want to consider more general properties. For example, if we are interested in all the consequences of T about Socrates, the property P is: “the consequence contains an instance of the constant symbol “Socrates””. In that case the answer is $\{Man(Socrates), Athenian(Socrates), Mortal(Socrates)\}$.

Thus, we view Constrained Consequence Generation as a next step where the problem is to find all the logical consequences of a theory that satisfy a given property P . Therefore, a general definition of the problem is:

query: property P about sentences,

answer: $\{c : T \vdash c \text{ and } P(c)\}$

where $P(c)$ means that c satisfies the property P .

The property P plays the role of a filter for the set of consequences, but, for efficiency reasons, we do not want to first generate all the consequences and then to select those that satisfy P . On the contrary, we want to only generate

consequences that satisfy P. In fact, P plays the role of a constraint on the inference rule used to generate consequences. This is why this new approach has been called Constrained Consequence Generation.

This problem may be solved in different ways. Our approach is to define a new kind of inference rule and an associated deduction strategy which only generate the consequences satisfying the property P. The originality of our work is that *the definitions of both the inference rule and the deduction strategy are generic*: they are not just defined for a particular property, but for a generic property P; so they are respectively called P-inference and P-deduction.

In the same spirit, we wanted the completeness proof to be generic too. The idea is that it is enough to check that the property P satisfies some conditions to ensure completeness.

The paper is organized as follows: in section 2, we give the definitions of the P-inference and P-deduction. In section 3, we show that if the property P verifies three particular conditions, then P-inference is complete. This is the most significant result presented in the paper. Indeed, from this result, for a given property P, it is sufficient to prove that P satisfies the three conditions to be warranted that the corresponding P-inference rule is complete. In most cases, it is much easier to prove that the conditions are satisfied than to directly prove completeness. Section 4 provides some examples of application: we consider particular properties, and we show that they satisfy the conditions stated in section 3.

Notice that, for technical reasons, in the rest of the paper it will be assumed that the theory T is a first order theory in clausal form. Also, to make the paper easier to read, some details, or some particular cases in the proofs are not explicated; however, when that is the case, it will be explicitly mentioned.

2 P-inference and P-deduction

Definition: P-clause

Let P be a property of clauses. Then, a P-clause is a clause that has the property P.

We will adopt the following notations:

$P(C)$: denotes that clause C is a P-clause.

$\text{Res}(A,B)$: denotes the resolvent, by Resolution Principle, of clauses A and B.

$A = \text{Inst}(A')$: denotes that clause A is an instance of clause A'.

Definition: P-inference

Let $\{e_1, e_2, \dots, e_p\}$ be a set of clauses, called electrons, and n be a clause,

called the nucleus. The clause C is the resolvent by P-inference of the set of electrons $\{e_1, e_2, \dots, e_p\}$, and of the nucleus n iff we have:

- e_1, e_2, \dots, e_p are P-clauses,
- $R_0 = n$ and $R_p = C$,
- for i in $[0, p-1]$: $R_{i+1} = \text{Res}(e_{i+1}, R_i)$ and R_{i+1} is a P-clause.

Notice that P-inference is a particular kind of Hyperresolution [5].

Definition: P-deduction

Let S be a set of clauses. A P-deduction of the clause C from S is a finite sequence of clauses: C_0, C_1, \dots, C_n such that $C=C_n$, and for i in $[0, n]$ we have: $C_i \in S$ or C_i is derived by a P-inference from the set of electrons $\{C_{i_1}, C_{i_2}, \dots, C_{i_p}\}$, and the nucleus C_j , where $C_j \in S$ and i_1, i_2, \dots, i_p, j are less than i .

It is worth noting that all the derived consequences in a P-deduction are P-clauses. Another important feature is that every nucleus is in S .

As a matter of fact, P-deduction defines a whole class of deductions, depending on how the clause property P is instantiated. All these deductions are sound; this is a direct consequence of the soundness of Resolution Principle. The next section studies the completeness problem.

3 Completeness of P-inference

We say that P-inference is complete iff any P-clause that is a consequence of a given theory can be derived from this theory by a P-deduction.

P-inference is not complete for any property P , but we have found sufficient conditions that ensure the completeness of P-inference. We first explain what the conditions consist of, and then give the completeness theorem.

The conditions are expressed for any clauses $A, B, C, D, E, F, A', B', C', E'$, where A, B, C, E are instances of A', B', C', E' .

1. **Backward Chaining condition:** a property P satisfies the Backward Chaining condition iff:

$$\text{If } C=\text{Res}(A,B) \text{ then } P(C) \text{ implies } P(A) \text{ or } P(B)$$

The intuitive meaning of this condition is that the property P propagates from a resolvent to one of the parent clauses.

2. **Forward Chaining condition:** a property P satisfies the Forward Chaining condition iff:

If $C = \text{Res}(A, B)$ and $A = \text{Inst}(A')$ and $B = \text{Inst}(B')$ and $C' = \text{Res}(A', B')$
then $P(A)$ and $P(B)$ and $P(A')$ implies $P(C')$.

The intuitive meaning of this condition is that, under some hypotheses, the property P propagates from a parent clause to the resolvent.

3. **Associativity condition:** A property P satisfies the associativity condition iff:

If $E = \text{Res}(A, \text{Res}(B, C))$ with $D = \text{Res}(B, C)$ and $E' = \text{Res}(\text{Res}(A, B), C)$ with $F = \text{Res}(A, B)$ then $P(A)$ and $P(D)$ and $P(E)$ implies $P(F)$ and $P(E')$.

The intuitive meaning of this condition is that, under some conditions, the property P is preserved when a sequence of resolutions is transformed using associativity.

However, (3) is a simplified version of the exact Associativity condition, which is in fact more complex. There are two main reasons why the definition is in fact more complex.

Firstly, classical resolution is not associative: using associativity not all the sequences of classical resolutions can be transformed in an equivalent proof (i.e. with the same resolvent). But the problem may be partially resolved if we consider clauses that are multi-sets of literals. So, as a first step, we have adapted classical resolution to deal with clauses that are multi-sets. We have called this new resolution, “extended resolution”. We have also introduced the notion of “extension” of a clause, as defined below.

Definition: extension of a clause

A clause C' is an extension of a clause C , iff there exists a substitution “ s ” such that $C'.s = C''$, and C can be obtained from C'' , by removing duplicated occurrences of some literals.²

In fact, all the definitions and conditions above do not use the classical definition of resolution but the extended one. In particular, $A = \text{Inst}(A')$ means that A' is an extension of A . We have checked in [2] that, if in the standard definition of resolution we consider multisets instead of sets, then the derived clauses are the same, or are extensions of clauses derived by standard resolution.

²For example, $P \vee M \vee M$ is an extension of $P \vee M$, but $P \vee M \vee N$ is not. Notice also that a clause may be considered as a (trivial) extension of itself.

In the following, we will just use the term “resolution” instead of “extended resolution”.

Secondly, strictly speaking, not even extended resolution is associative, and two different cases have to be considered. For the sake of simplicity, we only present two examples that give the intuition of why we need to distinguish these two cases. In the first example below resolution is associative.

Let us first consider the sequence of two resolutions:

$$\frac{\frac{A \vee N(a, a)}{\frac{B \vee M \vee \neg N(x, y) \vee \neg N(z, z) \quad C \vee \neg M}{B \vee C \vee \neg N(x, y) \vee \neg N(z, z)}}{B \vee C \vee \neg N(x, x)}}{A \vee B \vee C}$$

where $B \vee C \vee \neg N(x, x)$ is a factor of $B \vee C \vee \neg N(x, y) \vee \neg N(z, z)$.

If this sequence is transformed by associativity we get the same resolvent:

$$\frac{\frac{A \vee N(a, a) \quad B \vee M \vee \neg N(x, y) \vee \neg N(z, z)}{A \vee B \vee M} \quad C \vee \neg M}{A \vee B \vee C}$$

This example works well. But let us now consider this second example of resolution:

$$\frac{\frac{A \vee N(a, a)}{\frac{B \vee M \vee \neg N(x, y) \vee \neg N(z, z) \quad C \vee \neg M \vee \neg N(z, z)}{B \vee C \vee \neg N(x, y) \vee \neg N(z, z)}}{B \vee C \vee \neg N(x, x)}}{A \vee B \vee C}$$

If it is transformed by standard associativity we get:

$$\frac{\frac{A \vee N(a, a) \quad B \vee M \vee \neg N(x, y)}{A \vee B \vee M} \quad C \vee \neg M \vee \neg N(z, z)}{A \vee B \vee C \vee \neg N(z, z)}$$

If we want the final resolvent to be the same, we have to use the clause $A \vee N(a, a)$ twice, in order to delete the literal $N(z, z)$:

$$\frac{\frac{\frac{A \vee N(a, a) \quad B \vee M \vee \neg N(x, y)}{A \vee B \vee M} \quad C \vee \neg M \vee \neg N(z, z)}{A \vee B \vee C \vee \neg N(z, z)} \quad A \vee N(a, a)}{A \vee B \vee C}$$

The significant difference between the two cases is that the two factorised literals $\neg N(x, y)$ and $\neg N(z, z)$ are, or are not, in the same clause at the beginning. We have defined an extended notion of associativity that corresponds to each of the two above transformations, and we have shown that derivations that are transformed by this associativity derive the “same” (up to an extension) clauses.

The three conditions above ensure that P-inference is complete.

Theorem: Completeness of P-deduction

Let S be a set of clauses. Let P be a property of clauses that satisfies the Backward Chaining, Forward Chaining, and Associativity conditions. Let C be a P-clause. If there exists an R-deduction³ of C from S, then there exists a P-deduction of a clause C' such that C' is an extension of C.

According to this theorem, for a particular property of clause it is sufficient to check that the property satisfies the three conditions to be guaranteed that the corresponding inference is complete. This result is really attractive as it is often easier to check whether the property satisfies the conditions than to directly prove completeness.

Proof:

The proof is by induction on the number “n” of inferences in the R-deduction of C from the set of clauses S. The general idea is to show that any R-deduction of a P-clause can be transformed into a P-deduction of that clause.

Because the proof is sometimes long and boring, we only give a detailed sketch of the proof. The reader should be able to complete the missing details.

We use the notation $l(R)$ to express the number of inferences in the deduction R.

Case n=1

Let C_1 and D_1 be the parent clauses of C. As P satisfies the Backward Chaining condition, either C_1 or D_1 is a P-clause. Let us assume C_1 is a P-clause. Then the inference

$$\frac{C_1 \quad D_1}{C}$$

is a P-inference whose nucleus D_1 belongs to S. So the sequence C_1, D_1, C is a P-deduction of C from S.

³An R-deduction of a clause C from a set of clauses S is a finite sequence of clauses such that: for every C_i either C_i is a clause of S, or there exists C_{i1}, C_{i2} in the R-deduction such that C_i is the resolvent, by (extended) resolution principle of C_{i1} and C_{i2} .

Induction Hypothesis: For the following, we assume that the theorem is true for any R-deduction of a clause C from S, whose length is less or equal to n.

General case

Let us assume $n > 1$. Let R be the R-deduction of C from S, and C be the resolvent of C_1 and D_1 .

Because P satisfies the Backward Chaining condition and C is a P-clause, at least one of the two parent clauses of C is a P-clause. Let us assume that C_1 is a P-clause. In that case, we have to consider several cases:

- D_1 belongs to S.
- D_1 does not belong to S and D_1 is a P-clause.
- D_1 does not belong to S and D_1 is not a P-clause.

D_1 belongs to S

Let R_1 be a minimal R-deduction in R to deduce C_1 from S (see Figure 1). The number of inferences in R_1 is less or equal to n. As C_1 is a P-clause, by induction hypothesis, there exists a P-deduction P_1 of a P-clause C'_1 from S such that C'_1 is an extension of C_1 . As P satisfies the Forward chaining condition, C'_1 can be resolved with D_1 , and the resolvent is a P-clause C' , extension of C. As D_1 belongs to S, the sequence (P_1, D_1, C') is a P-deduction of C' from S.

$$\begin{array}{ccc} [R_1] & & [P_1] \\ \vdots & & \vdots \\ \frac{C_1 \quad D_1}{C} & & \frac{C'_1 \quad D_1}{C'} \end{array}$$

Figure 1: Case where D_1 belongs to S.

D_1 does not belong to S and it is a P-clause

Let R_1 and R_2 be the minimal R-deductions in R to derive C_1 and D_1 respectively (see Figure 2). Notice that $l(R_1) + l(R_2) + 1 = n$. Then $l(R_1) < n$ and $l(R_2) < n$. As C_1 and D_1 are both P-clauses, by induction hypothesis, there exists a P-deduction P_1 from S of a P-clause C'_1 which is an extension of C_1 ; and a P-deduction P_2 of a P-clause D'_1 , which is an extension of D_1 . As P satisfies the

Forward Chaining condition, C'_1 and D'_1 can be resolved into a P-clause C' which is an extension of C .

$$\begin{array}{c} [R_1] \quad [R_2] \\ \vdots \quad \vdots \\ \underline{C_1 \quad D_1} \\ C \end{array} \qquad \begin{array}{c} [P_1] \quad [P_2] \\ \vdots \quad \vdots \\ \underline{C'_1 \quad D'_1} \\ C' \end{array}$$

Figure 2: Case where D_1 does not belong to S and it is a P-clause.

Then, it is easy to show [2] that the sequence (P_1, P_2, C') can be transformed in a P-deduction of C' from S ⁴.

D_1 does not belong to S and it is not a P-clause

In that case, let us call C_2 and D_2 the parent clauses of D_1 . Let us call R_1, R_2, R_3 , the R-deductions from S of C_1, C_2, D_2 respectively (see Figure 3).

As property P satisfies the Associativity condition, it is possible to switch the order of resolutions, and preserve the property P (remember that C_1 is assumed to be a P-clause): C_1 can be resolved with C_2 to give a P-clause C_3 ; and C_3 can be resolved with clause D_2 , the resolvent being a P-clause C' , which is an extension of C .

$$\begin{array}{c} [R_1] \quad [R_2] \quad [R_3] \\ \vdots \quad \vdots \quad \vdots \\ \underline{C_1 \quad \underline{C_2 \quad D_2}} \\ C \end{array} \qquad \begin{array}{c} [R_1] \quad [R_2] \\ \vdots \quad \vdots \\ \underline{C_1 \quad C_2} \\ C_3 \end{array} \quad \begin{array}{c} [R_3] \\ \vdots \\ \underline{D_2} \\ C' \end{array}$$

Figure 3: Case where D_1 does not belong to S and it is not a P-clause.

Now, notice that in the original R-deduction the number n of inferences is equal to: $l(R_1)+l(R_2)+l(R_3)+2$. So the number of inferences in the R-deduction of C_3 , which is equal to $l(R_1) + l(R_2) + l(R_3) + 1$ is less than n . Since C_3 is a P-clause, by induction hypothesis, there exists a P-deduction P_1 from S of a clause C'_3 , extension of the clause C_3 . Now, we can also consider the R-deduction of the P-clause C' from the set of clauses $S \cup \{C_3\}$. The length of this deduction,

⁴If N is the nucleus, and E is the set of electrons in the P-inference that derives D'_1 in P_2 , then, it is possible to define a P-deduction that derives C' from the nucleus N and the set of electrons $E \cup \{C'_1\}$.

equal to $l(R_3) + 1$, is less than n . So, by induction hypothesis, there exists a P-deduction P_2 from $S \cup \{C_3\}$, of a P-clause C'' which is an extension of C' .

It is then possible to show that P_1 and P_2 can be connected in order to form a single resulting P-deduction⁵ from S of a clause C' , extension of C .

Notice that in fact, we should study another case, corresponding to the second example at the beginning of section 3. However, the proof may be longer but not more difficult than in the previous case.

This concludes the proof. The next section gives examples of particular complete deductions.

4 Instanciating property P

4.1 L-inference

The context of abductive reasoning has already be mentionned in the introduction: part of the problem is to find the hypothesis h to be added to a given theory T for a given sentence q to be a logical consequence of $T \cup h$. From the Deduction Theorem we know that it is the same problem to find h such that: $T \cup h \vdash q$, or such that: $T \vdash q \leftarrow h$, or such that: $T \vdash q \vee \neg h$. In the case where q is an open formula we add a new predicate symbol $L(x)$ to the language, and we add to T the clauses corresponding to $\forall x(L(x) \leftrightarrow q(x))$. Let T' be this new theory. For abductive reasoning we have to find all the clauses of the form $L(x) \vee c(x)$ such that $T' \vdash L(x) \vee c(x)$.

Now we can see that the problem is to find all the clauses that have the property of containing an instance of the given literal $L(x)$. For this purpose we define a particular kind of P-clause, called L-clause.

Definition: L-clause

A clause C is an L-clause iff it contains a literal L' such that L is an instance of L' .

It is easy to check that L-clauses satisfy the three conditions that warrant the completeness of the L-inference.

Backward Chaining condition:

Let C be an L-clause, resolvent of two clauses A and B . Then C contains a literal L' such that L is an instance of L' . The literal L' comes from one of the two parent clauses A or B . So, there is either in A or in B a literal L'' such that L' is an instance of L'' . Therefore either A or B is an L-clause.

⁵In fact, we consider the P-inference of P_2 which uses the clause C_3 . Then two cases have to be studied: C_3 is used as a nucleus, or C_3 is used as an electron. The fact we obtain C'_3 (and not C_3) is not a problem because of the Forward chaining condition.

Forward Chaining condition:

If C is the resolvent of A and B , and C' is the resolvent of A' and B' , where A' and B' are respectively generalizations of A and B , then C is an instance of C' . If C is an L-clause it contains a literal L' such that L is an instance of L' , and, since C is an instance of C' , C' contains a literal L'' such that L' is an instance of L'' , and therefore C' is an L-clause.

Associativity condition:

We consider the same notations as in the definition of the condition. Since E is an L-clause it contains a literal L' such that L is an instance of L' . Since D is not an L-clause, L' is an instance of some literal L'' in A . Let M_1 be the literal in D resolved upon the literal M_2 in A , in the resolution of A and D . M_1 is an instance of some literal M'_1 in B or C , or in both of them. Let us assume that M'_1 is only in B (other cases are similar); it is then possible to resolve A and B by resolving M_2 in A upon M'_1 in B . The resolvent F contains an instance L''' of L'' , and it can be shown that L is an instance of L''' . Therefore F is an L-clause.

The resolvent F also contains an instance N'_2 of N_2 , and it is possible to resolve F and C by resolving N'_2 in F upon N_1 in C . Since extended clauses are in fact multisets, even if L''' and N'_2 are two occurrences of the same literal, they are not factorized, and the resolvent of F and C contains an instance of L''' . It can be shown that this instance of L''' is more general than L , and therefore E' is an L-clause.

Given the completeness of L-inference it has also been shown in [6] that all the consequences that contain an instance of a given literal L can be generated by the following strategy.

In the first step all the instances of clauses in T that contain a literal L_i unifiable with L are generated, where the generated instance is obtained by applying the most general unifier of L and L_i . In the next steps all the L_j -resolvent are generated, where L_j is an instance of L , obtained from a set of electrons in the current set of clauses, and a nucleus in T .

The strategy is illustrated by the example below, where clauses are represented in implicative form for easier understanding only. In the example a predicate $p(x)$ means that “student x takes a class in discipline p ”.

- (a) $\text{humanities}(x) \leftarrow \text{logic}(x)$
- (b) $\text{humanities}(x) \leftarrow \text{aesthetics}(x)$
- (c) $\text{logic}(x) \vee \text{aesthetics}(x) \leftarrow \text{philosophy}(x)$
- (d) $\leftarrow \text{aesthetics}(x) \wedge \text{mathematics}(x)$

We consider two different queries:

Query 1: $\text{humanities}(x) \wedge \text{mathematics}(x)$; then, the following clauses are added to the theory:

- (e) $L(x) \leftarrow \text{humanities} \wedge \text{mathematics}(x)$
- (f) $\text{humanities}(x) \leftarrow L(x)$
- (g) $\text{mathematics}(x) \leftarrow L(x)$

The clauses generated at each step are:

- 1) $L(x) \leftarrow \text{humanities}(x) \wedge \text{mathematics}(x)$
- 2) $L(x) \leftarrow \text{logic}(x) \wedge \text{mathematics}(x)$,
 $L(x) \leftarrow \text{aesthetics}(x) \wedge \text{mathematics}(x)$

The clause $L(x) \leftarrow \text{aesthetics}(x) \wedge \text{mathematics}(x)$ is removed because it is subsumed by the clause (d).

- 3) $L(x) \leftarrow \text{philosophy}(x) \wedge \neg \text{aesthetics}(x) \wedge \text{mathematics}(x)$

From 3) and (d) we can generate:

- 4) $L(x) \leftarrow \text{philosophy}(x) \wedge \text{mathematics}(x)$

Since 4) subsumes 3), 3) is removed, and we get the answer:

- $L(x) \leftarrow \text{humanities}(x) \wedge \text{mathematics}(x)$
- $L(x) \leftarrow \text{logic}(x) \wedge \text{mathematics}(x)$
- $L(x) \leftarrow \text{philosophy}(x) \wedge \text{mathematics}(x)$

Query 2: $\text{humanities}(x)$; then we add the following clauses to the theory:

- (h) $L(x) \leftarrow \text{humanities}(x)$
- (i) $\text{humanities}(x) \leftarrow L(x)$

and the following clauses are generated:

- 1) $L(x) \leftarrow \text{humanities}(x)$
- 2) $L(x) \leftarrow \text{logic}(x)$, $L(x) \leftarrow \text{aesthetics}(x)$

Since the L-inference is an hyperresolution, from the two clauses generated in 2) and (c), we can generate:

- 3) $L(x) \leftarrow \text{philosophy}(x)$

and the final answer is:

- $L(x) \leftarrow \text{humanities}(x)$

$L(x) \leftarrow \text{logic}(x)$
 $L(x) \leftarrow \text{aesthetics}(x)$
 $L(x) \leftarrow \text{philosophy}(x)$

4.2 T-inference

In some applications, e.g. the generation of cooperative answers [2, 3, 8], it is interesting to derive all the consequences of a theory that are related to a given topic T . For example, if the theory represents the regulations of a university, one may be interested in all the consequences of the regulations that are about the topic “Teaching”.

Here, as a matter of simplification, we consider that a clause is about the topic T if it contains a literal formed with a predicate symbol related to the topic T ⁶. In that case a set of predicate symbols is associated with each topic. For instance the set of predicate symbols related to the topic Teaching may be: $\text{Teach}(x,y)$, that means that the professor x teaches the course y , and $\text{Pre}(x,y)$, that means that the course x is a prerequisite for the course y . In this example, the clause: $\text{PhD}(x,y) \vee \neg \text{Teach}(x,y)$, meaning that if x teaches y then x has a PhD in y , is a clause about the topic Teaching.

A general definition of clauses that are about a topic T is the following one.

Definition: T-clause

Let P_1, P_2, \dots, P_n be the set of predicates related to the topic T . A literal is related to the topic T if it is formed with one of the predicate symbols P_1, P_2, \dots, P_n . A clause C is a T -clause iff it contains at least one literal related to the topic T .

It is easy to check that T -clauses satisfy the three conditions that ensure the completeness of T -inference.

Backward Chaining condition: if the resolvent C of A and B is a T -clause, then it contains some literal L related to the topic T . Since L -clauses satisfy the Backward Chaining condition, either A or B is an L -clause, and therefore either A or B is a T -clause.

Forward chaining condition: if C is the resolvent of A and B , and A and C are T -clauses, then C contains a literal L related to T , and C is an L -clause. Then, either A or B is an L -clause. If C' is the resolvent of A' and B' , that are respectively generalizations of A and B , then, because L -clauses satisfy the Forward Chaining condition, C' is an L -clause, and therefore it is a T -clause.

Associativity condition: let us consider the same notations as for the

⁶A more sophisticated analysis of the links between sentences and topics can be found in [4].

definition of the condition. It is assumed that A and E are T-clauses, while D is not a T-clause, and we have to show that a consequence is that F and E' are T-clauses.

Since E is a T-clause, E is an L-clause for some literal L related to the topic T. Since L-clauses satisfy backward chaining condition, either A or B is an L-clause. However if D were an L-clause, it would be a T-clause, which contradicts the assumptions. Therefore A and E are L-clauses, and D is not an L-clause, and from associativity property of L-clauses, F and E' are L-clause, and then they are T-clauses.

The following example shows the interest of T-inference.

Let's consider the predicates:

A(x): x is a student,

B(x): x is tall,

P(x): x plays basket ball,

Q(x): x is a sportman,

R(x): x is in good shape, and the topic T: Sport.

Let assume the predicates P(x), Q(x), and R(x) are related to the topic T. The following deduction is a T-deduction of the clause: $R(a) \vee \neg A(a)$.

$$\frac{\frac{\frac{P(x) \vee \neg A(x) \vee \neg B(x) \quad B(a)}{P(a) \vee \neg A(a)} \quad Q(x) \vee \neg P(x)}{Q(a) \vee \neg A(a)} \quad R(x) \vee \neg Q(x)}{R(a) \vee \neg A(a)}$$

It is interesting to notice that this deduction cannot be considered as a P(a)-deduction, or as a Q(a)-deduction, or as an R(a)-deduction. Indeed $R(a) \vee \neg A(a)$ is neither a P(a)-clause nor a Q(a)-clause, and there are intermediate resolvents that are not R(a)-clauses.

However, since the L-inference is complete there exists an R(a)-deduction of $R(a) \vee \neg A(a)$. Given this remark, one could think of another means of deriving all the T-clauses, namely by deriving all the L_i -clauses from L_i -deductions, for all the literals L_i related to the topic T. However this method would be less efficient. Indeed, if there are, for example, T-clauses that are both L_i -clauses and L_j -clauses, they would be derived twice, from an L_i -deduction and from an L_j -deduction.

4.3 a-inference

The introduction provides the example of a query where one is interested in all the consequences about the individual "Socrates". This refers to the more

general problem of generating clauses containing a given constant symbol “a”, which we investigate now.

Definition: a-clause

A clause C is an a-clause iff it contains the constant symbol “a”.

Unfortunately the a-clauses do not satisfy the Forward Chaining condition as shown in the following example.

$$\frac{A : P(a) \vee Q(a) \quad B : \neg P(x)}{C : Q(a)} \qquad \frac{A' : P(a) \vee Q(y) \quad B' : \neg P(x)}{C' : Q(y)}$$

Even worse, there are a-clauses that cannot be derived by an a-deduction.

For example, let us consider the three clauses:

$$(1): N(a,z), (2): L(x) \vee \neg N(x',x) \vee M, (3): \neg N(y,y) \vee \neg M.$$

If clause (1) is resolved with the resolvent of (2) and (3), we get the a-clause L(a), but the deduction is not an a-deduction.

$$\frac{N(a,z) \quad \frac{L(x) \vee \neg N(x',x) \vee M \quad \neg N(y,y) \vee \neg M}{L(x) \vee \neg N(x',x) \vee \neg N(y,y)}}{L(x) \vee \neg N(x,x)} \quad \frac{L(x) \vee \neg N(x,x)}{L(a)}$$

However, this deduction cannot be transformed into an a-deduction.

Indeed, if we resolve the resolvent of (1) and (2) with (3), we get L(x) ∨ ¬N(y,y), which is not an a-clause, and contains new predicates. It is worth noting that if this consequence] is resolved again with (1) we get the clause L(x) which is not an a-clause but subsumes L(a):

$$\frac{N(a,z) \quad \frac{L(x) \vee \neg N(x',x) \vee M}{L(x) \vee M}}{L(x) \vee \neg N(y,y)} \quad \frac{\neg N(y,y) \vee \neg M}{N(a,t)} \quad \frac{L(x) \vee \neg N(y,y) \quad N(a,t)}{L(x)}$$

Another possibility is to resolve the resolvent of (1) and (3) with (2); and then to resolve again with (1). But, once more, we do not get an a-clause but a clause that subsumes L(a):

$$\frac{\frac{N(a, z) \quad \neg N(y, y) \vee \neg M}{\neg M} \quad L(x) \vee \neg N(x', x) \vee M}{\frac{L(x) \vee \neg N(x', x)}{L(x)} \quad N(a, t)}$$

There is no other possible derivation from the three clauses (1), (2) and (3).

In all the examples we have found, where an a-clause is not derivable by an a-deduction, it happens that this a-clause is not minimal with respect to subsumption. This lets us hope that we can find a proof of the following conjecture:

Conjecture:

Let C be a consequence of a given theory T . If C is an a-clause which is minimal with respect to subsumption, then there exists an a-deduction of C from T .

5 Conclusion

In this paper Constrained Consequence Generation has been introduced as a possible next step in the field of Automated Deduction. The idea is that, given a theory, the set of consequences one is interested in can be characterized in a more flexible way by a property which is verified by all the consequences of the set.

In order to tackle this problem, we have defined the P-inference rule, and the notion of P-deduction, which is a generic definition of a whole class of deductions. Moreover, the completeness proof we provide is generic too: we have isolated three sufficient conditions that P has to verify to guarantee the completeness. We have also provided some particular examples in application domains as important as Abductive Reasoning, or Cooperative Answering.

We see one main restriction to our work: the three sufficient conditions are somewhat restrictive. However, it has not been proven that they are necessary conditions, leaving some hope for future investigations aiming at relaxing these conditions.

Nevertheless, focusing the generation of consequences seems of great importance to us, and we know few works in this direction except [1, 10, 8, 9, 7].

References

References

[1] J.M. Boi, E. Innocente, A. Rauzy, P. Siegel. Production fields: A New

Approach to Deduction Problems and two Algorithms for Propositional Calculus. *The Journal of Artificial Intelligence*, To appear.

- [2] S. Cazalens. *Formalisation en Logique non standard de certaines méthodes de raisonnement pour fournir des réponses coopératives, dans des systèmes de Bases de Données et de Connaissances*. PhD thesis, Université Paul Sabatier, Toulouse France, 1992.
- [3] S. Cazalens and R. Demolombe. Intelligent Access to Data and Knowledge Bases via User's Topics of Interest. In *IFIP Congress'92, Madrid*, 1992.
- [4] S. Cazalens, R. Demolombe, and A. Jones. A Logic for Reasoning about Is About. Technical report, ONERA-CERT, 1992.
- [5] R.C.T. Lee C.L. Chang. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.
- [6] R. Demolombe and L. Fariñas del Cerro. An Inference Rule for Hypothesis Generation. In *Proc. of International Joint Conference on Artificial Intelligence*, Sydney, 1991.
- [7] K. Inoue. Consequence-Finding Based on Ordered Linear Resolution. In *Proc. of International Joint Conference on Artificial Intelligence*, Sydney, 1991.
- [8] P. Marquis. Extending Abduction from Propositional to First Order Logic. In *Proc International Workshop on Fundamentals of AI Research*, 1991.
- [9] P. Marquis. Novelty Revisited. In *Proc. of the 6th International Symposium on Methodologies for Intelligent Systems*, 1991.
- [10] P.Siegel. *Représentation et utilisation de la connaissance en Calcul Propositionnel*. PhD thesis, Université de Aix-Marseille, 1987.