Finding new consequences of an observation in a system of agents
Gauvain Bourgne, Katsumi Inoue, Nicolas Maudet

To cite this version:
Finding new consequences of an observation in a system of agents

Gauvain Bourgne  
NII  
Tokyo, Japan  
bourgne@nii.ac.jp

Katsumi Inoue  
NII  
Tokyo, Japan  
ki@nii.ac.jp

Nicolas Maudet  
LIP6 - SMA Team, UPMC  
Paris, France  
nicolas.maudet@lip6.fr

ABSTRACT
When a new observation is added to an existing logical theory, it is often necessary to compute new consequences of this observation together with the theory. This paper investigates whether this reasoning task can be performed incrementally in a distributed setting involving first-order theories. We propose a complete asynchronous algorithm for this non-trivial task, and illustrate it with a small example. As some produced consequences may not be new, we also propose a post-processing technique.

Categories and Subject Descriptors
I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence — Multiagent systems

General Terms
Algorithms, Theory

Keywords
Distributed Consequence Finding, Incremental Consequence Finding, Abduction

1. INTRODUCTION
This paper deals with the problem of finding all interesting new consequences which can be derived from some observations, given a full clausal theory. A consequence is deemed interesting if it respects a given language bias, and new if it is a consequence of the observations taken together with the theory but was not a consequence of the theory alone. Consequence finding is a general reasoning problem which lies at the heart of many AI applications. By focusing on computation of new consequences, one can perform efficient online computation of interesting consequences, an essential feature in dynamic contexts. On top of it, some problems specifically require to compute only new consequences, such as abduction by the principle of inverse entailment. Indeed, the set of abductive hypotheses is exactly the set of the negation of new consequences of the negated observation wrt the background theory. The computation of new interesting consequences is thus a very important challenge.

Of course, one can always compute new consequences by computing all consequences of the theory with and without the observations, and making the difference. But focusing only on new consequences is much more efficient, which can be especially interesting in context when information is accessed progressively. The research question we address in this paper is the following: does it still hold in a distributed setting? There exist methods for computing new consequences in a distributed setting [1], but restricted to the propositional case. On the other hand, some recent work [2] focus on computing interesting consequences of a first-order theory, but not only the new ones. We propose here a method that can deal with first order clausal theories while focusing on interesting new consequences. We introduce, in Section 2, (distributed) new consequence finding, and then present our algorithm (Section 3). Section 4 then concludes.

2. FINDING NEW CONSEQUENCES
A clause is a disjunction of literals. A clause C subsumes another clause D if there is a substitution θ such that Cθ ⊆ D. A clausal theory is a set of clauses, interpreted as the conjunction of all clauses in it. A consequence of Σ is a clause entailed by Σ. A clause C belongs to a production field P = ⟨L⟩, where L is a set of literals closed under instantiation, iff every literal in C belongs to L. The set of all subsumption-minimal consequences of a theory Σ that belongs to a production field P is called the characteristic clauses of Σ wrt P, and denoted by Carch(Σ, P). When some observations O are added to a clausal theory Σ, further consequences are derived due to this new information. Such new and interesting consequences are called new characteristic clauses. It is formally defined as the set of all subsumption-minimal consequences of Σ ∪ O belong to P that are not consequences of Σ, and denoted by Newarch(Σ, C, P).

We now consider a system of nA agents a0, . . . , anA−1, each having a clausal theory Σi. I = {0, . . . , nA − 1} denotes the set of indexes of all agents in the system. These agents make some new observations (or acquire new information), represented as a set of clauses Oi, possibly empty. The objective is to determine all the new consequences of those new observations O = ∪i∈I Oi, wrt the whole theory Σ = ∪i∈I Σi, belonging to the shared target production field P = ⟨L⟩, that is, to compute Newarch(∪i∈I Σi, ∪i∈I Oi, ⟨L⟩). This specifies a distributed new consequence finding problem. We emphasize that agents do not share their theories, though for better efficiency, they share their respective languages.

Example 1. Consider a system of 4 agents, whose knowledge (theory and new observations) is defined as follows:
\[
\begin{align*}
a_0: & \quad \Sigma_0 = \{f \lor g, a \lor g,\} \\
a_1: & \quad \Sigma_1 = \{-a \lor b, \neg g \lor h\}, \quad O_1 = \emptyset \\
a_2: & \quad \Sigma_2 = \{-b \lor c \lor d, \neg d \lor \neg e\}, \quad O_2 = \emptyset \\
a_3: & \quad \Sigma_3 = \{-c \lor \neg f\}, \quad O_3 = \emptyset.
\end{align*}
\]

The target production field is \( P = \{\{h\}\} \) (i.e. \( L_P = \{h\}\)).

3. DISTRIBUTED ALGORITHM

The main principle of our algorithm is to compute locally all relevant new consequences (and only those ones) and forward them to agents that can resolve them. Relevant consequences either (i) are new characteristics clauses of the problem, or (ii) can be used by one or more other agents to build such a new characteristic clause. Those latter ones, called bridge consequences, necessarily contains literals that can be resolved by other agents. We thus define, for each agent \( a_i \), the output language \( L_{o\rightarrow i} \), the set of all literals that (i) \( a_i \) might produce\(^1\) and (ii) can be resolved with a clause from another agent\(^2\) and an input language \( L_{i\rightarrow a_i} \), of an agent

\[ a_i \text{ as the set of all literals that (i) might be produced by another agent and (ii) can be resolved by some clause in its knowledge.} \]

Agents do not know each other theories, but they know each other input languages. Agents can focus their computations by using \( L_{o\rightarrow i} \) and \( L_P \). Though a bridge consequence \( C \) could have literals that are not in these production fields, such literals can only appear if they were in a received clause. We thus define the reduction of \( C \) wrt some language \( L(\text{reduc}(C, L)) \) as the set of all literals that appear in \( C \), but do not appear in positive nor negative form in \( L \).

To achieve better efficiency, we apply a prune function to the received clauses, which checks them against \( \Sigma_i \cup \text{listCsq}_i \), removing any subsumed clause.

Algorithm 1 Asynchronous algorithm

```plaintext
Global variables of agent \( a_i \): 
\[ \Sigma_i, O_i \] initialized by problem, constant
\[ \text{firstRes} \leftarrow \text{true} \]
\[ \text{listCsq}_i \leftarrow \emptyset \]
// Whenever agent \( a_i \) receive \( \text{sentCl} \) from an agent
\[ \text{Receive}(\text{sentCl}) \]
if \( \text{firstRes} \) then \( \text{sentCl} \leftarrow \text{sentCl} \cup O_i \) and if \( \text{firstRes} \leftarrow \text{false} \)
// Computing new consequences
\[ \text{prune}(\text{sentCl}) \]
\[ \text{pField} \leftarrow (\Sigma_P \cup L_{o\rightarrow i} \cup \text{reduc}(\text{sentCl}, \Sigma_i)) \]
\[ \text{newCarC} \leftarrow \text{reduc}(\text{pField} \cup \text{listCsq}_i, \text{sentCl}, \text{pField}) \]
\[ \text{listCsq}_i \leftarrow \text{listCsq}_i \cup \text{newCarC} \]
// Sending relevant new consequences to neighbors
for all agents \( a_j \) do
\[ \text{toSend}(j) \leftarrow \emptyset \]
for all \( c \in \text{newCarC} \) do
If \( c \) contains literals from \( L_{o\rightarrow i} \) then
\[ \text{toSend}(j) \leftarrow \text{toSend}(j) \cup \langle c \rangle \]
end if
end for
if \( \text{toSend}(j) \neq \emptyset \) then
\[ \text{send}(a_j, \text{toSend}(j)) \]
end if
end for
// Check new consequences as output
for all \( c \in \text{newCarC} \) do
If belongs(\( c, L_P \)) then
\[ \text{Output}(c) \]
end if
end for
End
```

Example 2. (ex. 1 ctd.) Figure 1 illustrates the unfolding of the asynchronous algorithm. Each box represents an agent (its index) applying the receive procedure. Arrows between two boxes correspond to the communication of some meaning that it must appear in at least one clause of \( a_i \), meaning that there is at least one clause in the theory of a different agent that contains the negation of this literal. clauses (given as label) by the first agent to the second one. The process is initiated by \( a_0 \), who send \( e \) to \( a_2 \) (as \( e \) is only in \( L_{\rightarrow 2} \)). Then \( a_2 \) computes the new consequences of \( e \) wrt to \( L_2 \) with production field \( \{(h \lor b, \neg e, \{c\})\} \) getting \( \neg b \lor c \), which partially belongs to \( L_{\text{all} \rightarrow 1} \) (through \( \neg b \)) and \( L_{\rightarrow 3} \) (\( c \)). It is thus sent to these two agents. Then \( a_1 \) computes \( \text{Newcar}(\Sigma_1, \neg b \lor c, \{(h, \neg a, \neg b, \neg g, \{c\})\}) \), and gets \( \neg a \lor \neg c \), which is sent to \( a_0 \) and \( a_3 \), and so on, until \( h \) is sent as output and other branches are closed.

![Figure 1: Asynchronous resolution of pb 1.](image)

4. CONCLUSION

We proposed in this paper a complete asynchronous algorithm to compute the new interesting consequences of some observations with respect to a full clausal theory distributed among a set of agents. Termination is guaranteed in cases where the centralized case also terminates, and soundness is ensured for incremental computations of consequences. Moreover some post processing was proposed to ensure soundness for computation of new consequences.

5. REFERENCES

