Finding new consequences of an observation in a system of agents

(Extended Abstract)

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

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

Nicolas Maudet
LIP6, 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 to remove them.

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.This can be especially interesting in contexts where information is accessed progressively. The research question we address here 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] allows computation of all interesting consequences of a distributed first-order theory, but it cannot focus on the new ones. We propose here a method that can deal with first order clausal theories while focusing on interesting new consequences.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

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 $C\theta \subseteq D$. A *clausal theory* is a set of clauses, interpreted conjunctively. A *consequence* of Σ is a clause entailed by Σ. A clause C *belongs to* a *production field* $\mathcal{P} = \langle \mathcal{L} \rangle$, where $\mathcal{L}$ is a set of literals closed under instantiation, iff every literal in C belongs to $\mathcal{L}$. The set of all subsumption-minimal consequences of a theory Σ belonging to a production field $\mathcal{P}$ is called the *characteristic clauses* of Σ wrt $\mathcal{P}$[3], and denoted by $Carc(\Sigma, \mathcal{P})$. 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 $\Sigma \cup O$ belonging to $\mathcal{P}$ that are not consequences of Σ, and is denoted by $Newcarc(\Sigma, C, \mathcal{P})$.

We now consider a system of n_A agents $I = \{0, \ldots, n_A - 1\}$, each having a clausal theory Σ_i. Some of these agents make new observations (or acquire new information), represented as a set of clauses O_i. The objective is to determine all the new consequences of those new observations $O = \bigcup_{i \in I} O_i$ wrt the whole theory $\Sigma = \bigcup_{i \in I} \Sigma_i$ belonging to the shared *target production field* $\mathcal{P} = \langle \mathcal{L}_P \rangle$, that is, to compute $Newcarc(\bigcup_{i \in I} \Sigma_i, \bigcup_{i \in I} O_i, \langle \mathcal{L}_P \rangle)$. 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:

0:	$\Sigma_0 = \{f \vee g, a \vee g\}$,	$O_0 = \{e\}$.
1:	$\Sigma_1 = \{\neg a \vee b, \neg g \vee h\}$,	$O_1 = \emptyset$.
2:	$\Sigma_2 = \{\neg b \vee c \vee d, \neg d \vee \neg e\}$,	$O_2 = \emptyset$.
3:	$\Sigma_3 = \{\neg c \vee \neg f\}$,	$O_3 = \emptyset$.

The target production field is $\mathcal{P} = \langle\{h\}\rangle$ (*i.e.* $\mathcal{L}_P = \{h\}$).

3. DISTRIBUTED ALGORITHM

As in [4, 2], 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 are either (i)new characteristics clauses of the problem, or (ii) *bridge* consequences, that is, consequences that can be used by one or more other agents to build such a new characteristic clause. In that latter case, they necessarily contains literals that can be resolved by other agents. We thus define, for each agent i the *output language* $\mathcal{L}_{i\rightarrow}$ as the set of all literals that (i) i might produce and (ii) can be resolved with a clause from another agent. Likewise, the *input language* $\mathcal{L}_{\rightarrow i}$ of an agent i is 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 $\mathcal{L}_{\rightarrow i}$ and $\mathcal{L}_P$. 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 $\mathcal{L}$ (reduc$(C, \mathcal{L})$) as the set of all literals that appear in C, but do not appear in positive nor negative form in $\mathcal{L}$. To achieve better efficiency, we apply a prune function to the received clauses, which checks them against $\Sigma_i \cup listCsq_i$, removing any subsumed clause.

Algorithm 1 Asynchronous algorithm

```
Global variables of agent $i$:
   $\Sigma_i, O_i$: initialized by problem, constant
   $firstRun \leftarrow true$
   $listCsq_i \leftarrow \emptyset$
// Whenever agent $i$ receive $sentCl$ from an agent
Receive($sentCl$)
   if $firstRun$ then $sentCl \leftarrow sentCl \cup O_i$ end if
   $firstRun \leftarrow false$
   // Computing new consequences
   prune($sentCl$)
   $pField \leftarrow \langle \mathcal{L}_P \cup \mathcal{L}_{i\rightarrow} \cup reduc(sentCl, \mathcal{L}_{i\rightarrow})\rangle$
   $newCsq_i \leftarrow newcarc(\Sigma_i \cup listCsq_i, sentCl, pField)$
   $listCsq_i \leftarrow listCsq_i \cup newCsq_i$
   // Sending relevant new consequences to neighbours
   for all agents $j$ do
      $toSend[j] \leftarrow \emptyset$
      for all $c \in newCsq_i$ do
         if $c$ contains literals from $\mathcal{L}_{\rightarrow j}$ then
            $toSend[j] \leftarrow toSend[j] \cup \{c\}$
         end if
      end for
      if $toSend[j] \neq \emptyset$ then
         $send(j, toSend[j])$
      end if
   end for
   //Check new consequences as output
   for all $c \in newCsq_i$ do
      if belongs($c$, $\mathcal{L}_P$) then
         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 applying the *receive* procedure. Arrows between two boxes correspond to the communication of some clauses (given as label) by the first agent to the second one. The process is initiated by 0, who send e to 2 (as e is only in $\mathcal{L}_{\rightarrow 2}$). Then 2 computes the new consequences of e wrt to Σ_2 with production field $\langle\{h, \neg b, \neg e, c\}\rangle$ getting $\neg b \vee c$, which partially belongs to $\mathcal{L}_{\rightarrow 1}$ (through $\neg b$) and $\mathcal{L}_{\rightarrow 3}$ (c). It is thus sent to these two agents. Then 1 computes $Newcarc(\Sigma_1, \neg b \vee c, \langle\{h, \neg a, b, \neg g, c\}\rangle)$, and gets $\neg a \vee \neg c$, which is sent to 0 and 3, and so on, until h is sent as output and other branches are closed.

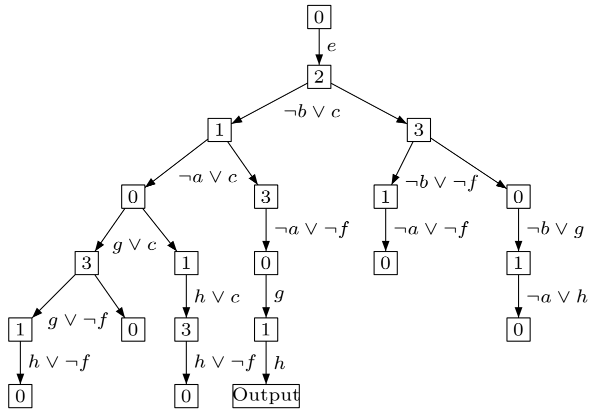

Figure 1: Asynchronous resolution of pb 1.

Termination is guaranteed for non-recursive theories. Otherwise, we need to enforce termination by fixing a limit to the number of resolve operations that can be applied to get a consequence. This algorithm is complete for multi-agent new consequence finding, meaning that it outputs all new consequences of $\bigcup_{i \in I} O_i$ wrt $\bigcup_{i \in I} \Sigma_i$ and $\langle \mathcal{L}_P \rangle$. It also ensures that each output is indeed a consequence of $\bigcup_{i \in I} O_i \cup \bigcup_{i \in I} \Sigma_i$. However, it might also be a consequence of the theory alone (and thus not strictly a new consequence). If our purpose is to incrementally compute all characteristic clauses, this is not a problem at all, but in some other cases, such as the computation of abductive hypothesis, we should only output *new* characteristic clauses. This can be ensured by computing, for each candidate consequence C, $N_C = Newcarc(\bigcup_{i \in I} \Sigma_i, \neg C, \langle\emptyset\rangle)$. If $N_C = \{\emptyset\}$, C is not new, otherwise, it can be kept as a solution.

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

[1] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *JAIR*, 25:269–314, 2006.

[2] G. Bourgne and K. Inoue. Partition-based consequence finding. In *Proc. of ICTAI'2011*, pages 641–648, 2011.

[3] K. Inoue. Linear resolution for consequence finding. *Art. Intel.*, 56(301–353), 1992.

[4] S. McIlraith and E. Amir. Theorem proving with structured theories. In *Proc. of IJCAI'01*, pages 624–634, 2001.