Finding new consequences of an observation in a system of agents

Gauvain Bourgne, Katsumi Inoue, Nicolas Maudet

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 exaclty 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.

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

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 $C\theta \subseteq D$. 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* $\mathcal{P} = \langle \mathcal{L} \rangle$, 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 that belongs 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 charactestic clauses. It is formally defined as the set of all subsumption-minimal consequences of $\Sigma \cup O$ belong to $\mathcal{P}$ that are not consequences of Σ, and denoted by $Newcarc(\Sigma, C, \mathcal{P})$.

We now consider a system of n_A agents $a_0, \ldots, a_{n_A-1}$, each having a clausal theory Σ_i. $I = \{0, \ldots, n_A - 1\}$ 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 O_i, possibly empty. 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:

a_0:	$\Sigma_0 = \{f \vee g, a \vee g\}$,	$O_0 = \{e\}$.
a_1:	$\Sigma_1 = \{\neg a \vee b, \neg g \vee h\}$,	$O_1 = \emptyset$.
a_2:	$\Sigma_2 = \{\neg b \vee c \vee d, \neg d \vee \neg e\}$,	$O_2 = \emptyset$.
a_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

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* $\mathcal{L}_{i\rightarrow}$ 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* $\mathcal{L}_{\rightarrow i}$ of an agent a_i 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 $\mathcal{L}_{\rightarrow i}$ 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 a_i:
  Σ_i, O_i: initialized by problem, constant
  firstRun ← true
  listCsq_i ← ∅
// Whenever agent a_i receive sentCl from an agent
Receive(sentCl)
  if firstRun then sentCl ← sentCl ∪ O_i end if
  firstRun ← false
  // Computing new consequences
  prune(sentCl)
  pField ← ⟨L_P ∪ L_{i→} ∪ reduc(sentCl, L_{i→})⟩
  newCsq_i ← newcarc(Σ_i ∪ listCsq_i, sentCl, pField)
  listCsq_i ← listCsq_i ∪ newCsq_i
  // Sending relevant new consequences to neighbours
  for all agents a_j do
    toSend[j] ← ∅
    for all c ∈ newCsq_i do
      if c contains literals from L_{→j} then
        toSend[j] ← toSend[j] ∪ {c}
      end if
    end for
    if toSend[j] ≠ ∅ then
      send(a_j, toSend[j])
    end if
  end for
  //Check new consequences as output
  for all c ∈ newCsq_i do
    if belongs(c, 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 (its index) 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 a_0, who send e to a_2 (as e is only in $\mathcal{L}_{\rightarrow 2}$). Then a_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 $\mathcal{L}_{all\rightarrow 1}$ (through $\neg b$) and $\mathcal{L}_{\rightarrow 3}$ (c). It is thus sent to these two agents. Then a_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 a_0 and a_3, and so on, until h is sent as output and other branches are closed.

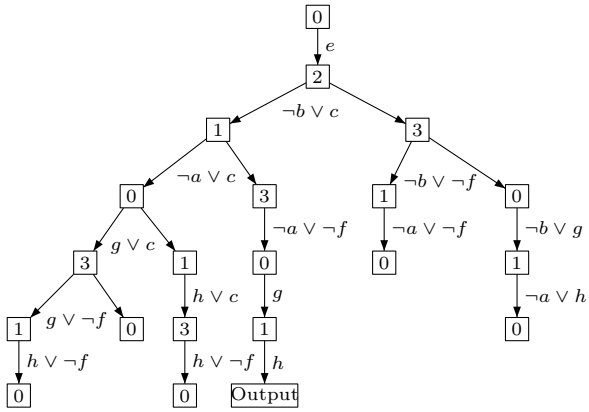

Figure 1: Asynchronous resolution of pb 1.

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 as follows. We remove all new observations, and for each candidate consequence C , compute $N_C = Newcarc(\bigcup_{i\in I} \Sigma_i, negC, \langle\emptyset\rangle)$. 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.

[1]meaning that it must appear in at least one clause of a_i.

[2]meaning that there is at least one clause in the theory of a different agent that contains the negation of this literal.