Reasoning with Inconsistencies in Propositional Peer-to-Peer Inference Systems

Philippe Chatalic, Gia Hien Nguyen, Marie-Christine Rousset

Reasoning with Inconsistencies in Propositional Peer-to-Peer Inference Systems [1]

Ph. Chatalic [2] and **G.H. Nguyen**[3] and **M.Ch. Rousset** [3]

Abstract.

In a peer-to-peer inference system, there is no centralized control or hierarchical organization: each peer is equivalent in functionality and cooperates with other peers in order to solve a collective reasoning task. Since peer theories model possibly different viewpoints, even if each local theory is consistent, the global theory may be inconsistent. We exhibit a distributed algorithm detecting inconsistencies in a fully decentralized setting. We provide a fully distributed reasoning algorithm, which computes only *well-founded* consequences of a formula, i.e., with a consistent set of support.

1 Introduction

Recently peer-to-peer (P2P) systems have received considerable attention because their underlying infrastructure is appropriate to scalable and flexible distributed applications over Internet. In P2P systems, there is no centralized control or hierarchical organization: each peer is equivalent in functionality and cooperates with other peers in order to solve a collective task. P2P systems have evolved from simple keyword-based file sharing systems like Napster [2] and Gnutella [1] to semantic data management systems like EDUTELLA [22], PIAZZA [18] or SOMEWHERE [5]. Reasoning in P2P Inference Systems (P2PIS) has been considered very recently (e.g., [3, 12, 4]). The dynamicity of P2PIS imposes to revisit many reasoning problems in order to address them in a decentralized manner. In particular, it is neither feasible to bring all the information to a single server and use standard reasoning algorithms, nor to compute its best partitioning for using partition-based reasoning like in [15, 8].

The local theories of the P2PIS considered in [3, 4] are sets of clauses defined upon sets of propositional variables (the local vocabularies of the peers). A new peer joins an existing P2PIS by establishing *mappings* with other peers, called its *acquaintances*. Mappings are clauses involving variables of distinct peers that state semantic correspondences between different vocabularies. The decentralized algorithm DECA [6] computes the consequences (in some target language) of an input formula w.r.t. the global theory of the P2PIS (i.e. the union of all peer theories). The point is to compute those consequences, without having access to the global theory. DECA is anytime, sound, and complete (under some conditions). It has been implemented in the SOMEWHERE platform and experiments on synthetic data have shown its scalability [4, 6].

This paper focuses on the problem of reasoning with inconsistencies in a P2PIS. Since peer theories model possibly different viewpoints, the global theory may be inconsistent even if each local theory is consistent. Given the lack of centralized control, all peers should be treated equally. It would be unfair to refuse the join of a new peer just because the resulting P2PIS becomes inconsistent. Our choice is to accept the presence of inconsistency. The problem is first to *detect* inconsistencies, second to *reason* in spite of them in a satisfactory way. We thus compute only *well-founded* consequences of a formula, i.e., consequences of the formula w.r.t. to a consistent subset of the global theory. Such an approach is not novel in the centralized case but raises new algorithmic issues in the decentralized case because the computation and the storage of *nogoods* (accounting for inconsistencies) are distributed. One has to be able to check the consistency of distributed sets of formulas, w.r.t. distributed sets of nogoods.

We assume each local theory to be consistent. Therefore, the possible inconsistencies result from interactions between local theories and are caused by *mappings*. Before adding a mapping, a peer checks whether this mapping (possibly with other mappings) can be the cause of some inconsistency, i.e., if the empty clause can be produced as one of its consequences. In that case, the peer stores locally as a *nogood* the set of mappings involved in the corresponding proof. At reasoning time, the concerned distributed nogoods must be collected to check whether the proof under construction is well-founded.

In Section 2, we formally define the considered P2PIS. In Section 3, we present a decentralized algorithm for detecting inconsistencies and computing the corresponding nogoods. Section 4 provides a decentralized reasoning algorithm which is well-founded despite possible inconsistencies. In Section 5, we conclude by relating our approach w.r.t. existing work.

2 Peer-to-peer inference systems

The P2PIS that we consider are networks of peer clausal theories $\mathcal{P} = \{P_i\}_{i=1..n}$, such that each peer has a proper *vocabulary* (denoted by $\mathcal{V}_{P_i}$). We suppose that each peer has a unique identifier (for example, its IP address) and that variable names use in some way this identifier. For simplicity, we just use the index i as the peer identifier and denote a variable A of the peer P_i by A_i. Each local theory of a peer P_i is a set of clauses without duplicated literals. We denote by $\mathcal{L}_{P_i}$ the language of clauses involving only variables of $\mathcal{V}_{P_i}$.

Definition 1 (Mappings, shared variables and acquaintances) *A* mapping *is a clause of a peer involving at least a variable of the vocabulary of another peer. A variable of some peer is said to be* shared *if it appears in a mapping of another peer. The* acquaintances *of a peer are the peers in the network with which it shares variables.*

The *global theory* $\mathcal{T}(\mathcal{P})$ of a P2PIS $\mathcal{P} = \{P_i\}_{i=1..n}$ is: $\bigcup_{i=1..n} P_i$. Since mappings play a special role in our approach, we distinguish $\mathcal{M} = \bigcup_{i=1..n} M_i$, where M_i is the set of mappings of P_i from

[1] This work is supported by France Telecom

[2] LRI-PCRI - Université Paris-Sud 11, Orsay, France. chatalic@lri.fr

[3] LSR-IMAG - Université Joseph Fourrier, Grenoble, France. {Gia-Hien.Nguyen,Marie-Christine.Rousset}@imag.fr

$\mathcal{O} = \bigcup_{i=1..n} O_i$, where each O_i is the complementary of M_i. In addition, we assume that each mapping of P_i has a unique identifier prefixed by P_i.

A shared literal is a shared variable or its negation. For a clause c, we denote by $S(c)$ the disjunction of its shared literals and by $L(c)$ the disjunction of its other literals. We suppose that each peer P knows its acquaintances, and given a shared literal l we denote by $\text{ACQ}(l, P)$ the set of peers with which P shares the variable of l.

In contrast with other approaches [17, 12], we do not adopt an epistemic or modal semantics for interpreting a P2PIS but we interpret it with the standard semantics of propositional logic. In particular, a variable shared by two peer theories of a given P2PIS is interpreted by the same value in the two peers.

Definition 2 (Semantics of a P2PIS) *Let $\mathcal{P} = \{P_i\}_{i=1..n}$ be a P2PIS, an* interpretation *I of $\mathcal{P}$ is an assignment of the variables of $\mathcal{T}(\mathcal{P})$ to $true$ or $false$. I is a* model of a clause *c iff one of the literals of c is evaluated to $true$ in I. I is a* model of a set of clauses *iff it is a model of all the clauses of the set.*

- $\mathcal{P}$ is consistent *iff $\mathcal{T}(\mathcal{P})$ has a model.*

- The consequence relation *is the standard one: $\mathcal{P} \models c$ iff every model of $\mathcal{P}$ is a model of c. We say that c is an implicate of $\mathcal{P}$.*

Example

Let us consider the P2PIS corresponding to Figure 1. P_1 can be asked by researchers for choosing where to submit their results (demos or papers). For instance, part of its knowledge can model that: $PODS06$ is open for submission; submitting to $PODS06$ entails submitting to $PODS$; only theoretical results are submitted to $PODS$; a demo cannot be submitted to JAIR. P_2 distinguishes proceedings from journals and knows that: submitting to PODS entails submitting to a conference with proceedings; submitting to JAIR entails submitting to a journal; a same result cannot be submitted in a conference and in a journal; patented results cannot be submitted to a journal. P_3 has some knowledge about research valorization policy: software should be patented or presented as demos; theoretical results should be submitted to journals. The knowledge expressed separately by P_1, P_2 and P_3 using their respective vocabularies can be respectively modelled by the set of clauses O_1, O_2 and O_3.

The set M_2 of mappings stored at P_2 states the equivalence between $PODS_1$ and $PODS_2$ (reps. $JAIR_1$ and $JAIR_2$) through the mappings identified by $P_2.1$, $P_2.2$, $P_2.3$ and $P_2.4$. The set M_3 of mappings stored at P_3 also establishes equivalences between variables of P_3 and variables of the two other peers. Note however that mappings clauses do not necessarily result from equivalences.

The reasoning problem

For each peer P_i, we consider a set $\mathcal{TV}_{P_i} \subseteq \mathcal{V}_{P_i}$ of *target variables*, supposed to represent the variables of interest for the application, (e.g., observable facts in a model-based diagnosis application, or classes storing data in an information integration application). For a set SP of peers of a P2PIS, we define its *target language* $\mathcal{T}arget(SP)$ as the language of clauses (including the empty clause) involving only variables of $\bigcup_{P \in SP} \mathcal{TV}_P$.

Definition 3 (P2PIS consequence finding problem) *Given a P2PIS $\mathcal{P}$, a peer P of $\mathcal{P}$ and a clause $q \in \mathcal{L}_P$, the* P2PIS consequence finding problem *is to find the set of* proper prime implicates *of q w.r.t. $\mathcal{T}(\mathcal{P})$ that belong to $\mathcal{T}arget(\mathcal{P})$.*

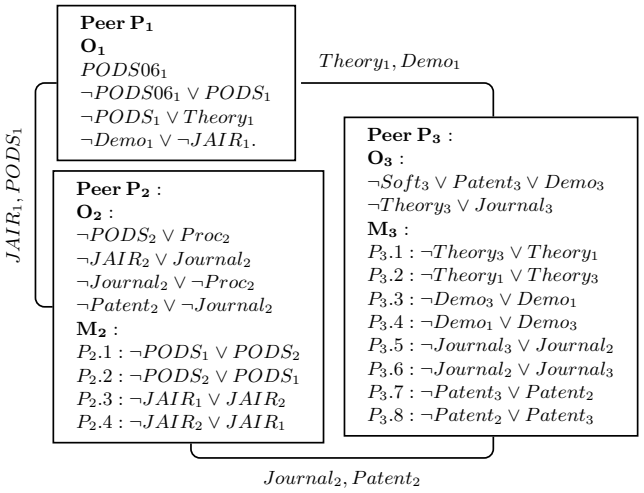

Figure 1. Example P2PIS network (edges labelled by shared variables).

A *proper prime implicate* of q w.r.t. a (distributed) theory $\mathcal{T}$ is a prime implicate of $\mathcal{T} \cup \{q\}$, which is not a prime implicate of $\mathcal{T}$.

DECA [3, 6] is the first fully decentralized algorithm being able to solve this problem without having a global view of the system. Details and illustrations of DECA behavior may be found in [6]. Roughly summarized, when running at a peer P and asked to compute the proper prime implicates of a literal q, it first computes all proper prime implicates of this literal w.r.t. the local theory of P, selects those of interest w.r.t. the target language and then split each clause c obtained in this way in two subclauses : $L(c)$ and $S(c)$. $S(c)$ is in turn splitted and for each shared literal l of $S(c)$ DECA asks its appropriate acquaintances (which are running the very same algorithm) to further propagate l in the P2PIS and to return the corresponding results. As soon as consequences are obtained for all literals of $S(c)$ they are recombined together, as well as with $L(c)$, to produce consequences of the splitted clause c. Different branches of reasoning are thus developed throughout the network, following the shared variables, and thus the mappings. In order to ensure termination in presence of cycles as well as to handle the transmission of the results back to the appropriate peers, an *history* is associated to each propagated literal and updated each time that the corresponding reasoning branch goes out from one peer into another peer.

3 P2P detecting inconsistencies and nogoods

As outlined in the introduction, even if each P_i is locally consistent, this is not necessarily the case for $\mathcal{T}(\mathcal{P})$. We assume that the causes of inconsistencies are only due to mappings and define a **nogood** ng as a set of mappings such that $\mathcal{O} \cup ng$ is inconsistent. Note that in a nogood ng there necessarily exists some mapping m from which one can derive the empty clause (i.e., with a proof rooted in m). We exploit this property in the decentralized algorithm P2P-NG to detect nogoods accounting for inconsistencies. P2P-NG runs at each peer and is used before adding a new mapping m to a peer P, to check whether the propagation of m into the existing P2PIS produces the empty clause. If that is the case, P stores locally as new nogoods the mappings (including m) involved in the corresponding derivations. We call the set of mappings used in a derivation, its **mapping support**. The P2P-NG algorithm computes the (possibly empty) set of mapping supports of the derivations of the empty clause rooted in

m, starting at the peer P. Since mappings have a unique identifier, mapping supports can efficiently be encoded as sets of mapping identifiers (and similarly for nogoods computed from mapping supports).

P2P-NG is an adaptation of the DECA consequence finding algorithm. It follows the same split-recombine strategy but it has the following significant differences:

- The stopping conditions **must** be changed. The ones in the original DECA algorithm were designed for the computation of *proper prime* implicates only. Here we need to find all the possible ways of deriving the empty clause, we cannot stop the reasoning as soon as we produce the empty clause, or as soon as we find a unit clause in a local peer which is the same as the literal under processing.

- Because we are only looking for $\square$ as a consequence, locally produced consequences c such that $L(c) \neq \square$ can be filtered out.

- While DECA would return $\{\square\}$ as its result if there exists a derivation of the empty clause, P2P-NG returns as many mapping supports as different ways of deriving $\square$.

We use the following notations:

- For a clause c, $Resolvent{+}SMS(c, P)$ computes the couples of pairs $(r\ SMS(r, P))$ such that r is a local consequence of c w.r.t. P and $SMS(r, P))$ is its corresponding set of local mapping supports.

- For a literal q, $\bar{q}$ denotes its complementary literal.

- A history $hist$ is a sequence of tuples (l, P, c) where l is a literal, P a peer, and c a clause which is a consequence of l on the peer P. A history $[(l_n, P_n, c_n), \ldots, (l_1, P_1, c_1), (l_0, P_0, c_0)]$ represents a branch of reasoning initiated by the propagation of the literal l_0 within the peer P_0, which either has produced locally the clause c_0 in P_0 (in that case, c_0 may have been splitted into its different literals among which l_1 is propagated in P_1), or not (in that case l_0 is simply propagated from P_0 to P_1 and $l_0 = c_0 = l_1$). For every $i \in [0..n{-}1]$, c_i is a consequence of l_i and P_i, and l_{i+1} is a literal of c_i, which is propagated in P_{i+1}.

- $\otimes$ is the distribution union operator on sets of sets: $SS_1 \otimes \cdots \otimes SS_n = \{S_1 \cup \cdots \cup S_n \,|S_1 \in SS_1, \ldots, S_n \in SS_n\}$. If $L = \{l_1, \ldots, l_p\}$, $\otimes_{l \in L} SS_l$ denotes $SS_{l_1} \otimes \cdots \otimes SS_{l_p}$.

Each literal resulting from the splitting of a clause (Line (4) of the P2P-NG algorithm) is processed independently by the P2P-NG$_l$ algorithm. P2P-NG$_l(q, SP, hist)$ checks whether there exists a derivation of $\square$ rooted in the literal q, starting with the computation of local consequences of q, and then recursively following the acquaintances of the visited peers. To ensure termination, it is necessary to keep track of the literals already processed by peers. This is done thanks to $hist$, where $hist$ is the history of the reasoning branch ending up to the propagation of the literal q in SP, which is the set of acquaintances of the last peer added to the history.

Theorem 1 states that the result of P2P-NG(m, P) can be used to characterize nogoods involving the mapping m (by assimilating mappings to their corresponding index).

Theorem 1 *Let m be a mapping and P be a peer such that* P2P-NG$(m, P) \neq \emptyset$. $\forall ms \in$ P2P-NG(m, P), $ms \cup \{m\}$ *is a nogood.*

Before adding a new mapping m to its local theory, each peer P first computes P2P-NG(m, P) and stores locally all the nogoods $\{m\} \cup ms$, such that $ms \in$ P2P-NG(m, P) is minimal (for inclusion).

Theorem 2 states that the P2P-NG algorithm is complete, i.e., it enables to find the mapping supports of all the irredundant derivations of the empty clause from a given mapping added to a P2PIS. The proof of Theorem 2 relies on Lemma 1.

Definition 4 (Irredundant derivation) *A derivation is irredundant if it does not involve two identical applications of the resolution rule.*

Algorithm 1: Detection of the nogoods caused by adding a mapping P2P-NG(m, P)

(1) LOCAL$(P) \leftarrow Resolvent{+}SMS(m, P)$
(2) RESULT $\leftarrow \emptyset$
(3) **foreach** $(c\ sms) \in$ LOCAL(P) s.t. $S(c) \neq \square$ and $L(c) = \square$
(4) **foreach** literal $q \in S(c)$
(5) NOGOODS$(q) \leftarrow$ P2P-NG$_l(q,$ ACQ$(q, P), \emptyset)$
(6) **if** for every $q \in S(c)$ NOGOODS$(q) \neq \emptyset$
(7) UNIONCOMB $\leftarrow sms \otimes (\otimes_{q \in S(c)}$NOGOODS$(q))$
(8) RESULT $\leftarrow$ RESULT $\cup$ UNIONCOMB
(9) **return** RESULT

P2P-NG$_l(q, SP, hist)$

(1) **if** for every $P \in SP$, $(q, P, \_) \in hist$
(2) **return** $\emptyset$
(3) **else**
(4) $SMS(q) \leftarrow \{\emptyset\}$
(5) RESULT $\leftarrow \emptyset$
(6) **if** $(\bar{q}, \_, \_) \in hist$
(7) RESULT $\leftarrow$ RESULT $\cup\ \{\emptyset\}$
(8) **foreach** $P \in SP$
(9) LOCAL$(P) \leftarrow \{(q\ \{\emptyset\})\} \cup Resolvent{+}SMS(q, P)$
(10) RESULT $\leftarrow$ RESULT $\cup \bigcup_{P \in SP} SMS(\square, P)$
(11) **foreach** $P \in SP$ and $(c\ sms) \in$ LOCAL(P) s.t. $S(c) \neq \square$ and $L(c) = \square$
(12) **foreach** literal $l \in S(c)$
(13) SMS$(l) \leftarrow$ P2P-NG$_l(l,$ ACQ$(l, P), [(q, P, c)|hist])$
(14) **if** for every $l \in S(c)$, SMS$(l) \neq \emptyset$
(15) UNIONCOMB $\leftarrow sms \otimes (\otimes_{l \in S(c)}SMS(l))$
(16) RESULT $\leftarrow$ RESULT $\cup$ UNIONCOMB
(17) **return** RESULT

Theorem 2 *Let $\mathcal{P}$ be a P2PIS and m a mapping of a given peer P of $\mathcal{P}$. Let ms be a mapping support of an irredundant derivation of $\square$ rooted in m. It will be returned by* P2P-NG(m, P).

Lemma 1 *Let ms be the mapping support of an irredundant derivation of $\square$ rooted in a clause c: $c_1 \vee \ldots \vee c_n$ where every c_i is a clause (which can be a unit clause or not) such that there is no literal common to c_i and c_j for $i \neq j$. There exists $ms_1, \ldots, ms_n$ where, for every i, ms_i is a mapping support of an irredundant derivation of $\square$ rooted in c_i, such that: $ms = ms_1 \cup \ldots \cup ms_n$.*

Corollary 1 results directly from Theorem 2. It guarantees that all the minimal nogoods are computed ans stored in the P2PIS. It is the key for proving that the reasoning algorithm presented in the next section is well-founded.

Corollary 1 *Let $\mathcal{T}(\mathcal{P}) = \mathcal{O} \cup \mathcal{M}$ be the global theory of a P2PIS and let S be a set of mappings of $\mathcal{M}$. If S is a minimal nogood then it is stored at some peer P of $\mathcal{P}$.*

Example (cont.): In the example of Section 2, let us suppose that the different peers join in the following order: P_1, then P_2, then P_3 and that their respective mappings are added according to their numbering. At the join of P_2, the successive adding of the 4 mappings causes no inconsistency. When P_3 joins, the 4 first mappings cause no inconsistency. Let us focus on the 5th one. P2P-NG$(\neg Journal_3 \vee Journal_2, P_3)$ is triggered where P_3 is the theory containing the clauses of P_3 that are not mappings, and the 4 first mappings (which have been added since they have been checked as not deriving inconsistencies). $\neg Theory_1 \vee Journal_2$ is produced

locally at P_3 as a local consequence of $\neg Journal_3 \vee Journal_2$, with a set of local mapping support equal to $\{\{P_3.2\}\}$. Then, $\neg Theory_1 \vee Journal_2$ is splitted (Line (4) of P2P-NG): $\neg Theory_1$ is processed by P_1, while $Journal_2$ is processed by P_2. The propagation of $\neg Theory_1$ produces $\Box$ as a local consequence in P_1 with a local set of mapping support equal to $\{\emptyset\}$. Thus $\{\emptyset\}$ is returned to P_3 as the set of mapping supports of the derivation of $\Box$ from $Theory_1$. The propagation of $Journal_2$ produces $\neg PODS_1$ as a local consequence in P_2, with a local set of mapping supports equal to $\{\{P_2.1\}\}$, as well as $\neg Patent_2$ with a local set of mapping support equal to $\{\emptyset\}$. $\neg PODS_1$ is in turn propagated in P_1, where it produces $\Box$ as a local consequence with a local set of mapping supports equal to $\{\emptyset\}$. It is transmitted back to P_2 which, after combination of $\{\emptyset\}$ and $\{\{P_2.1\}\}$ (Line 15) of P2P-NG$_l(Journal_2, \{P_2\}, \emptyset)$), transmits back to P_3 the set of mapping supports $\{\{P_2.1\}\}$ for the derivation $\Box$ from $Journal_2$. By combination (Line 7) P2P-NG$(\neg Journal_3 \vee Journal_2, P_3)$ returns $\{\{P_3.2, P_2.1\}\}$. The nogood $\{P_3.5, P_3.2, P_2.1\}$ is thus obtained and stored at P_3. No other nogood is obtained from the last mappings of P_3.

4 Peer-to-peer well-founded reasoning

First, we have to define the notion of *well-founded* consequences that can be derived from a given input clause and a possibly inconsistent P2PIS. Many semantics have been proposed and studied for reasoning with (centralized) inconsistent theories ([14] for a survey). We adopt the following one (which is one of the simplest ones), because it makes sense in a decentralized and dynamic setting.

Definition 5 (P2P well-founded implicate) *Let $\mathcal{P}$ be an inconsistent $P2PIS$: r is a* well-founded *implicate of c w.r.t. $\mathcal{P}$ if r is an implicate of c w.r.t. a* consistent subset *of $\mathcal{T}(\mathcal{P})$.*

The WF-DECA(q, P) algorithm computes well-founded consequences of the (unit) clause q, starting at the peer P. This algorithm extends the original DECA consequence finding algorithm [3, 6] by computing the set of mapping supports of the derivations for each consequence, and by collecting the nogoods encountered during the reasoning. Because of the split/recombination technique used by the algorithm, mapping supports of derivations are only known after the recombination step, and the set of possibly relevant nogoods must be available at this step: if some mapping support includes a nogood, it is discarded; consequences that get an empty set of mapping supports after nogoods filtering are discarded as well.

We use the following notations :

- $LocalConsSSNG(q, P)$ is a local procedure that computes the set of triples $(c\ sms\ sng)$ such that c is a local consequence of q w.r.t. P, sms is its corresponding set of local mapping supports, and sng is the set of nogoods stored at the peer P that contain a mapping m of some mapping support ms of c.
- $\uplus$ denotes the *merged union* of sets of consequences, i.e. the union of sets of triples of the form $(c\ sms\ sng)$, where triples corresponding to a same consequence c are merged together, by computing the union of their respective sms and sng.
- $\circledcirc$ is the distribution union operator on sets of triples of the form $(c\ \ sms\ \ sng)$: $S_1 \circledcirc \cdots \circledcirc S_n = \{(c_1 \vee \cdots \vee c_n\ \ sms_1 \otimes \cdots \otimes sms_n\ \ sng_1 \cup \cdots \cup sng_n)/(c_1\ sms_1\ sng_1) \in S_1, \ldots, (c_n\ sms_n\ sng_n) \in S_n\}$.

Theorem 3 states that the WF-DECA(q, P) algorithm terminates and returns only well-founded consequences. Its proof relies on showing that each triple $(c\ \ sms\ \ sng)$ returned by WF-

Algorithm 2: Well Founded Distributed Consequence Finding Algorithm

WF-DECA(q, P)
(1) WF-DECAH$(q, \{P\}, \emptyset)$

WF-DECAH$(q, SP, hist)$
(1) **if** for every $P \in SP$, $(q, P, \_) \in hist$
(2) **return** $\emptyset$
(3) **else if** $(\bar{q}, \_, \_) \in hist$
(4) **return** $\{(\Box\ \{\emptyset\}\ \emptyset)\}$
(5) **else**
(6) RESULT $\leftarrow \emptyset$
(7) **foreach** $P \in SP$
(8) LOCAL$(P) \leftarrow \{(q\ \{\emptyset\}\ \emptyset)\} \uplus LocalConsSSNG(q, P)$
(9) **foreach** $P \in SP$ and $(c\ sms\ sng) \in$ LOCAL(P) such that $L(c) \in \mathcal{T}arget(P))$
(10) **if** $S(c) \in \mathcal{T}arget(P)$
(11) RESULT $\leftarrow$ RESULT $\uplus \{(c\ sms\ sng)\}$
(12) **if** $S(c) \neq \Box$
(13) **foreach** literal $l \in S(c)$
(14) ANSWER$(l) \leftarrow$
(15) WF-DECAH$(l,$ ACQ$(q, P), [(q, P, c)|hist])$
(16) **if** for every $l \in S(c)$ ANSWER$(l) \neq \emptyset$
(17) UNIONCOMB $\leftarrow$
(18) $\{(L(c)\ sms\ sng)\} \circledcirc (\circledcirc_{l \in S(c)}$ANSWER$(l))$
(19) **foreach** $(c\ sms\ sng) \in$ UNIONCOMB
(20) $nsms \leftarrow \{ms \in sms / \forall ng \in sng, ng \not\subseteq ms\}$
(21) **if** $nsms \neq \emptyset$
(22) RESULT $\leftarrow$ RESULT $\uplus \{(c\ nsms\ sng)\}$
(23) **return** RESULT

DECA$(q, SP, hist)$ is such that either c is a local consequence of q w.r.t. some peer $P \in SP$, or for every $ms \in sms$, $\mathcal{O} \cup ms$ is consistent (using Corollary 1) and c is the result of a derivation rooted in q that only uses clauses from $\mathcal{O} \cup \{l_0, l_1, \ldots, l_n, q\} \cup ms$, where $l_0, l_1, \ldots, l_n$ are the literals in $hist$.

Theorem 3 *Let P be a peer of a P2PIS $\mathcal{P}$ and q a literal belonging to the vocabulary of P.* WF-DECA(q, P) *terminates and for all triples $(c\ sms\ sng)$ returned by* WF-DECA(q, P), *c is a well-founded consequence of q w.r.t. $\mathcal{P}$.*

Example (cont.): Let us illustrate the behaviour of WF-DECA$(Soft_3, P_3)$, assuming that the only target variables are $PODS_1$ and $JAIR_1$. $Patent_2 \vee Demo_1$ is the only clause produced locally on P_3 with a local part of which (i.e. $\Box$) in $\mathcal{T}arget(P_3)$. Its local sms is $\{\{P_3.7, P_3.3\}\}$. The only nogood stored at P_3 contains neither $P_3.7$ nor $P_3.3$. The corresponding sng returned by $LocalConsSSNG(Soft, P_3)$ is thus empty. $Patent_2 \vee Demo_1$ is then splitted. When $Patent_2$ is transmitted to P_2, $\neg JAIR_1$ is the only clause produced locally with a local part (i.e. $\Box$) in $\mathcal{T}arget(P_2)$. Its local sms is $\{\{P_2.3\}\}$ and its sng is empty. The further propagation of $\neg JAIR_1$ returns an empty result. So the triple $(\neg JAIR_1\{\{P_2.3\}\}\emptyset)$ is sent back to P_3 as a consequence of $Patent_2$. When $Demo_1$ is transmitted to P_1 the only clause produced locally is $\neg JAIR_1$, which is in $\mathcal{T}arget(P_1)$. Its local sms is empty, as well as its sng. So the triple $(\neg JAIR_1\{\emptyset\}\emptyset)$ is sent back to P_3 as a consequence of $Demo_1$. P_3 then combines these two triples obtained from P_2 and P_1 giving $(\neg JAIR_1\ \{\{P2.3\}\}\emptyset)$, which is the only final consequence of $Soft_3$ being in the target language. Since the corresponding sng is empty, it is trivially a well-founded consequence.

5 Conclusion and Related Work

We have presented a fully decentralized approach for reasoning with inconsistencies in propositional P2PIS. Nogoods are discovered each time a new mapping is added. Though these are stored in a completely distributed way, the WF-DECA algorithm guarantees that all consequences it returns are well-founded. For lack of place, many optimization details have been omitted. In particular, calculating *all* possible mapping supports of each consequence is not necessary. Computing only the minimal ones is sufficient. The implementation of this approach is currently under process. An extensive experimental study in the spirit of that of [4, 6] is planned for the near future.

For efficiency reasons our approach for P2P reasoning with more powerful languages is to approximate the peer theories and the queries into propositional logic and to exploit the current infrastructure to compute the corresponding propositional answers. The actual answers are then computed by focusing on the relevant peers.

Reasoning under inconsistency has been widely studied in artificial intelligence, but mainly in the centralized case. There are two kinds of strategies for dealing with inconsistent knowledge bases. The first one is to restore consistency as in belief revision [7, 16]. In the context of peer-to-peer interconnected databases, although not strictly restoring consistency, the work of [9] characterizes consistent answers as those that can be drawn from all minimal repairs of all peer databases and that satisfy as well a set of data exchange constraints (similar to mappings) relating the different peer schemes.

The alternative is just to tolerate inconsistency [10]. In a fully decentralized setting, all peers playing the same role, this seems far more preferable. A wide range of paraconsistent logics have been proposed [20] to avoid the trivialization problem of classical logic. Our approach is closer to coherence systems and argumentative approaches [14], since we consider consequences that can be produced from consistent subsets of $\mathcal{T}(\mathcal{P})$. Since $\mathcal{O}$ is known to be consistent, mapping supports can be considered as justifications supporting the consequences. Our approach is a credulous one, since any subset of $\mathcal{M}$ consistent with $\mathcal{O}$ is considered. Well-founded consequences can also be viewed as either local consequences or as the formulas of some extension of the supernormal default theory [11] $\Delta=(\mathcal{O}, \mathcal{M})$.

The work of [19] on reasoning from inconsistent ontologies is another kind of coherence-based approach. For a given query, it aims at finding a specific consistent subset of the global theory, in which the query (or its negation) classically holds. If it is not possible, the answer is undetermined. The set is constructed by successive consistent expansions, according to some selection function measuring some relevance criterion with the query. They use a syntactical criteria, that selects only formulas sharing some variable with those already selected during the previous iterations. Limiting the choice for the consistent subset clearly reduces the set of accepted consequences.

An epistemic semantics has been proposed in [13] to deal with possible inconsistencies in a P2P Data Inference Systems formalized in a first order multi modal language. It considers the case of local inconsistency (a point not addressed by our approach) as well as global inconsistency. Mapping are formalised in such a way that they cannot be used to propagate information from some locally inconsistent theory. Moreover mappings can only be used to propagate information to a peer, as far as they do not contradict either local information or other non-local information that may be deduced on that peer.

Distributed CSP techniques [24, 23] aim at finding consistent assignments of a set of variables, satisfying a set of distributed constraints. They also propagate nogoods corresponding to invalid partial affectations, that are further stored on the peer that receives them. This tends to replicate among all agents some global knowledge. This is also the case in distributed ATMS [21]. In comparison, we exploit nogoods only at reasoning time. Only those that may interfere with some mapping support of a related sms are transmitted in the sng.

REFERENCES

[1] GNUTELLA. http://www.gnutella.com.

[2] NAPSTER. http://www.napster.com.

[3] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon, 'Distributed reasoning in a peer-to-peer setting', in *Proc. ECAI'04*, pp. 945–946. IOS Press, (August 2004).

[4] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon, 'Scalability study of peer-to-peer consequence finding', in *IJCAI'05*, pp. 351–356. IJCAI, (August 2005).

[5] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon, 'Somewhere in the semantic web', in *Proc. of Principles and Practice of Semantic Web Reasoning*, eds., Francois Fages and Sylvain Soliman, volume 3703 of *LNCS*, pp. 1–16. Springer, (Sept. 2005).

[6] 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', *Journal of Artificial Intelligence Research*, **25**, (January 2006).

[7] Carlos Alchourron, Peter Gardenfors, and David Makinson, 'On the logic of theory change: Partial meet contraction and revision functions.', *Journal of Symbolic Logic,*, **50**, 510–530, (1985).

[8] E. Amir and S. McIlraith, 'Partition-based logical reasoning', in *Proc. of KR'00*, pp. 389–400, Breckenridge, Colorado, USA, (April 11-15 2000). Morgan Kaufmann Publishers.

[9] Leopoldo Bertossi and Loreto Bravo, 'Query answering in peer-to-peer data exchange systems', *eprint arXiv:cs/0401015*, (January 2004).

[10] Leopoldo E. Bertossi, Anthony Hunter, and Torsten Schaub, eds. *Inconsistency Tolerance*, volume 3300 of *LNCS*. Springer, 2005.

[11] Gerhard Brewka, 'Preferred subtheories: An extended logical framework for default reasoning.', in *Proc. of IJCAI*, pp. 1043–1048, (1989).

[12] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati, 'Logical foundations of peer-to-peer data integration', in *Proc. of PODS'04*, ed., Alin Deutsch, pp. 241–251, Paris, France, (june 14-16 2004). ACM.

[13] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati, 'Inconsistency tolerance in p2p data integration: an epistemic logic approach', in *Proc. of the 10th Int. Workshop on Database Programming Languages (DBPL 2005)*, (2005).

[14] Claudette Cayrol, 'On the relation between argumentation and non-monotonic coherence-based entailment.', in *Proc. of IJCAI*, pp. 1443–1448, (1995).

[15] R. Dechter and I. Rish, 'Directed resolution: the davis-putnam procedure revisited', in *Proc. of KR'94*, pp. 134–145, Bonn, Germany, (May 24-27 1994). Morgan Kaufmann.

[16] *Handbook of defeasible reasoning and uncertainty management*, eds., D. Dubois and H. Prade, Kluwer Academic Publishers, 1998.

[17] Chiara Ghidini and Luciano Serafini, *Frontiers Of Combining Systems 2*, chapter Distributed First Order Logics, 121–139, number 7 in Studies in Logic and Computation, Research Studies Press Ltd., 2000.

[18] Alon Y. Halevy, Zachary Ives, Igor Tatarinov, and Peter Mork, 'Piazza: data management infrastructure for semantic web applications', in *Proc. of WWW'03*, pp. 556–567. ACM Press, (2003).

[19] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije, 'Reasoning with inconsistent ontologies.', in *IJCAI'05*, pp. 454–459, (2005).

[20] Anthony Hunter, *Paraconsistent logics*, 11–36, Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[21] C.L. Mason and R.R. Johnson, *Distributed Artificial Intelligence II*, chapter DATMS: a framework for distributed assumption based reasoning, 293–317, Pitman, 1989.

[22] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, and al., 'Edutella: a p2p networking infrastructure based on rdf', in *Proc. of WWW'02*, pp. 604–615. ACM, (May 2002).

[23] Marius-Calin Silaghi, Djamila Sam-Haroud, and Boi Faltings, 'Asynchronous search with aggregations', in *Proc. of AAAI'00*, pp. 917–922. AAAI Press / The MIT Press, (2000).

[24] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara, 'The distributed constraint satisfaction problem: Formalization and algorithms', *IEEE Transactions on Knowledge and Data Engineering*, **10**(5), 673–685, (1998).