Exploiting Preferences over Information Sources to Efficiently Resolve Inconsistencies in Peer-to-peer Query Answering

Arnold Binas and **Sheila A. McIlraith**
Departement of Computer Science, University of Toronto, Toronto, Ontario, Canada
{abinas,sheila}@cs.toronto.edu

Abstract

Decentralized reasoning is receiving increasing attention due to the distributed nature of knowledge on the Web. We address the problem of answering queries to distributed propositional reasoners which may be mutually inconsistent. This paper provides a formal characterization of a prioritized peer-to-peer query answering framework that exploits a preference ordering over the peers as well as a distributed entailment relation. We develop decentralized algorithms for computing answers to a restricted class of queries according to distributed entailment and prove their soundness and completeness. A heuristic exploiting the peers' preference ordering for a more efficient computation of query answers from prioritized, inconsistent systems is also provided and its effectiveness empirically verified. We furthermore investigate global closed-world reasoning in our framework and show how our techniques can be used to achieve global generalized closed-world query answering from local closed-world reasoners in a restricted class of distributed peer-to-peer query answering systems.

1 Introduction

With the advent of the Web has come a significant increase in the availability of information from a variety of information sources, not all of which are mutually consistent or equally reliable. These information sources are often databases, but many foresee a future in which some of them will be deductive databases, logic programs, or even full-fledged logical reasoners. Motivated by this general problem, this paper addresses the problem of peer-to-peer (P2P) query answering over distributed propositional information sources that may be mutually inconsistent. We assume the existence of a preference ordering over the peers to discriminate between peers with conflicting information. This preference ordering may reflect an individual's level of trust in an information source or it may be obtained from an objective third party rating. We provide a formal characterization of a prioritized P2P query answering framework along with a distributed entailment relation based on argued entailment [3]. To realize the specification of our problem, we develop decentralized algorithms based on [1, 2] for computing answers to a restricted class of queries according to distributed entailment and prove their soundness and completeness. To improve the efficiency of reasoning, we propose heuristic and pruning techniques that exploit the preference ordering over peers and the resulting priority ordering over the peers' knowledge and empirically illustrate their effectiveness. A common problem with distributed query answering is that individual information sources may make a *local* closed-world assumption, and yet we wish to answer queries under a *global* closed-world assumption. We show how this interesting and important problem can be resolved as a special case of our framework for a restricted but compelling class of information sources.

There has been significant previous work on distributed logical reasoning. For example, Amir and McIlraith introduced partition-based logical reasoning for propositional and first-order logic (FOL) to reason with multiple knowledge bases (KBs) and to improve the efficiency of reasoning with large KBs [2]. Their approach was limited to consistent KBs connected in a tree topology. Adjiman et al. introduced the first consequence-finding algorithm for distributed propositional theories connected by graphs of arbitrary topology, but limited such systems to be globally consistent [1]. Chatalic et al. extended this approach to allow for mutually inconsistent peers that are connected by mapping clauses [7]. However, their approach allows for a formula and its negation to be derived as a consequence at the same time. Our work builds on this work, addressing a different general problem. We discuss other related work in further detail in the final section.

2 P2P Query Answering Systems

In this section we formalize a P2P query answering framework and provide a distributed entailment relation for such systems. We illustrate concepts via a running example.

2.1 Framework

A P2P query answering system (PQAS) consists of multiple peers, all of which are assigned user-specific priorities that reflect the user's preferences over the peers and are distributed to the peers when the user joins the network. Priorities are drawn from a totally ordered set of comparable elements (such as the set of integers), and a priority is *better* than another priority if its value is *lower*. The peers' priorities define a total or partial preference ordering over the peers. A peer with better priority is preferred over a peer with worse priority. Each peer furthermore hosts a consistent propositional local KB and consequence relation, which

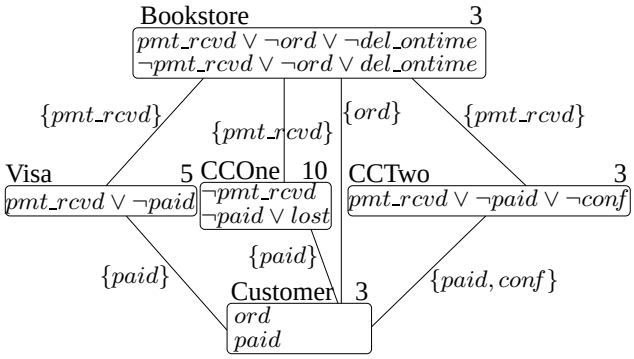

Figure 1: Bookstore example

may be classical entailment or any other consequence relation. Multiple peers' KBs may be mutually inconsistent. Peers are connected by labeled edges. An edge label is a set of variables and determines the language in which formulas can be passed between two peers.

Definition 1 (P2P query answering system (PQAS)). *A P2P query answering system $\mathcal{P}$ is a tuple (P, G) where $P = \{P_i\}_{i=1}^n$ is a set of n peers and G is a graph (V, E) describing the communication connections between those peers. A peer P_i is a triple $(KB_i, Cons_i, I_i)$ comprising a local propositional knowledge base KB_i, its local consequence relation $Cons_i$, and the peer's priority I_i. We call $\bigcup_{i=1}^n KB_i$ the global theory of $\mathcal{P}$. We use the notation $\Sigma \models_i \phi$ to denote $\phi \in Cons_i(\Sigma)$, where Σ is a set of formulas. We say that peer P_i has better priority than peer P_j iff $I_i < I_j$. A peer's signature L_i is the set of propositional symbols in its knowledge base. Each peer's signature L_i is a subset of the global signature $L = \bigcup_{j=1}^n L_j$ of $\mathcal{P}$. V is the set $\{1, \ldots, n\}$ of vertices in G, with vertex i corresponding to peer P_i. E is the set of labeled edges (i, j, L_{ij}) in G, where L_{ij} is the edge label (link signature) between peers P_i and P_j and $L_{ij} \subseteq L_i \cap L_j$.*

Figure 1 is a small example of a PQAS consisting of the peers $P = \{Bookstore, Visa, CCOne, CCTwo, Customer\}$. The priority I_i of peer i is displayed on its upper right corner. The bookstore's KB, KB_1, contains the axioms $\neg pmt_rcvd \wedge ord \rightarrow \neg del_ontime$ and $pmt_rcvd \wedge ord \rightarrow del_ontime$, stating that the book is delivered on time if the payment is received and an order placed and not delivered on time if the payment is not received. The Visa peer contains the KB $KB_2 = \{paid \rightarrow pmt_rcvd\}$, stating that the payment is received if money is paid. CCTwo (KB_4) additionally requires confirmation ($conf$). The unreliable CCOne has worse priority than the Visa peer and contains the KB $KB_3 = \{\neg pmt_rcvd, paid \rightarrow lost\}$, stating that the payment will not be received. The edges are labeled by the propositions which pairs of peers may use to communicate. An edge label between two peers only contains propositions which are mentioned in both peers' local KBs, but not necessarily all such propositions (perhaps for reasons of confidentiality).

2.2 Distributed Entailment

The distributed entailment we wish to achieve is best introduced through our example in Figure 1. Consider the case where Customer has asserted ord and $paid$ and poses the query del_ontime. According to Bookstore, the book is delivered on time if an order has been placed and the payment received. If the book is ordered and the payment not received, the book will not be delivered on time ($\neg del_ontime$). According to Visa, the payment is received ($paid$ and $pmt_rcvd \vee \neg paid$ imply pmt_rcvd), and del_ontime is derived. However, according to credit card peer CCOne, the payment will not be received by the bookstore ($\neg pmt_rcvd$) and $\neg del_ontime$ is derived. Clearly we have a contradiction. But since Visa is preferred over CCOne due to its better priority, we would like to receive the answer it supports. Hence we need del_ontime to be entailed according to *distributed entailment*.

In the following, we formally define this new notion of entailment. Here we employ the notion of *argued entailment* for prioritized KBs [3] in our definitions. Benferhat et al. defined a formula to be entailed by argued entailment if a better *reason* exists for it than for its negation. A reason for a formula is a *consistent* subset of the KB that classically entails the formula (since an inconsistent one could derive any formula and would thus be meaningless). In our distributed setting, we extend this definition to account for several, distributed KBs, possibly different local consequence relations, and restricted sharing of information between KBs.

Definition 2 (Support of a formula by a reason). *The formula ϕ is supported in a PQAS $\mathcal{P}$ by the reason Σ, denoted $\mathcal{P} \models^{\Sigma} \phi$, iff there exists a peer P_i s.t. $\mathcal{P} \models_i^{\Sigma} \phi$. $\mathcal{P} \models_i^{\Sigma} \phi$ iff $\Sigma \not\models \bot$ and one of the following conditions holds: (1) $\Sigma \subseteq KB_i$, $\Sigma \models_i \phi$, and for no $\Sigma^* \subset \Sigma$, $\Sigma^* \models_i \phi$ or (2) there exists a set of formulas $\Sigma' \subseteq KB_i$ and a set of formulas $\psi_1, \ldots, \psi_m$ s.t. $\Sigma' \cup \bigcup_{j=1}^m \{\psi_j\} \models_i \phi$, for no $\Sigma'^* \subset \Sigma'$, $\Sigma'^* \cup \bigcup_{j=1}^m \{\psi_j\} \models_i \phi$, and for each ψ_j, there exists a peer P_k and a set of formulas Σ_j s.t. $\mathcal{P} \models_k^{\Sigma_j} \psi_j$ and $(i, k, L_{ij}) \in E$ with $L_{ij} \supseteq sig(\psi_j)$, and $\Sigma = \Sigma' \cup \bigcup_{j=1}^m \Sigma_j$.*

Thus a reason for a formula ϕ exists in a PQAS $\mathcal{P}$ if some consistent subset of the peers' local KBs derives ϕ given the peers' local consequence relations and given that whenever two formulas of different peers need to interact, an edge between the peers exists and is labeled by all variables mentioned in the formula. Since a reason may exist for both a formula ϕ and its negation $\neg\phi$, we need to weigh the priorities of the formulas in both reasons in order to decide whether to believe ϕ or $\neg\phi$. The priority of a formula is that of its host peer. The rank of a reason is the priority of the worst-priority formula in it. Under distributed entailment, we will believe the result with the best rank-reason (lowest value), i.e. the result supported by the most preferred peers. The definitions are altered and adapted to the distributed case from [3].

Definition 3 (Rank of a reason). *The rank of a reason Σ is $R(\Sigma) = max_{\psi \in \Sigma}(prio(\psi))$, where the priority $prio(\psi)$ of a formula ψ is the priority I_i of the peer P_i it originated from.*

Definition 4 (Distributed entailment in a prioritized PQAS). *A formula ϕ is entailed by the PQAS $\mathcal{P}$, denoted $\mathcal{P} \models_D \phi$, iff there exists a reason Σ^+ supporting ϕ and for all reasons Σ^- supporting $\neg\phi$, $R(\Sigma^+) < R(\Sigma^-)$.*

While in the general case, a PQAS's peers may be mutually inconsistent, an inconsistency may not always be derived. This could be because all peers are mutually consistent or because the inconsistency is not derivable with the restricted communication imposed by peer connectivity. To distinguish between PQASs that can derive contradictions and those that cannot, we introduce $\mathcal{P}$-consistency.

Definition 5 ($\mathcal{P}$-consistency). *A PQAS $\mathcal{P}$ is $\mathcal{P}$-inconsistent iff there exists a formula ϕ s.t. there exists both a reason for ϕ and a reason for $\neg\phi$ in $\mathcal{P}$. Otherwise $\mathcal{P}$ is $\mathcal{P}$-consistent.*

Since in a $\mathcal{P}$-consistent system no contradiction can be derived, the existence of a reason for a formula ϕ guarantees that no reason for $\neg\phi$ exists and thus that ϕ is entailed by distributed entailment in $\mathcal{P}$.

3 Query Answering in a PQAS

Given the PQAS framework, we want to pose a formula as a query and have the system determine whether the query is true, false, or unknown according to our definition of distributed entailment, using individual peers' local knowledge and reasoning capabilities. The goal of this section is to develop a message-passing algorithm that solves this problem for single-literal queries in a possibly $\mathcal{P}$-inconsistent PQAS in which all peers use classical entailment as their local consequence relation.

A consequence-finding algorithm for systems of mutually consistent peers already exists [1]. Adjiman et al.'s algorithm computes consequences of a single-literal query for arbitrary peer topologies and shall serve as the basis for our query answering algorithm. In this section, we modify and extend Adjiman et al.'s algorithm to answer single-literal queries in possibly $\mathcal{P}$-inconsistent PQASs according to distributed entailment, which supports prioritized peers and generates best-priority answers. Our focus on prioritized peers enables a search space pruning technique and search heuristic that drastically improves query processing. A query is posed by a query peer that may be one of the peers of $\mathcal{P}$ or an additional peer connected to $\mathcal{P}$.

3.1 Algorithm

To answer queries according to distributed entailment, we first need an algorithm to compute reasons for a query. Reasons are computed by the message-passing Algorithms 2, 3, 4, and 5. In order to compute reasons for a query literal q by refutation, a *forth* message containing $\neg q$ is sent to a peer whose signature contains q. Each way to derive the empty clause from a consistent set of original peer clauses constitutes a reason for q. A history is used to keep track of which literals and clauses the currently processed literal depended on, enabling the algorithm to detect if a literal has been processed by the same peer before or whether it depends on its own negation in the history (and thus derives the empty clause). Each derived clause has an associated OA set (for "original ancestor") which contains all original parent clauses except the negation of the query. OA sets are used to verify that a derived clause depends only on a *consistent* set of parent clauses. The following notation, proposed by [1], is used in the algorithms.

Definition 6 (History ([1])). *A history $hist$ is a list of tuples (l, P, c) of a literal l, a peer P, and a clause c. c is a consequence of l in P, and l is a literal of the clause of the previous tuple in the $hist$ list.*

Definition 7 (Local consequences ([1])). *$Resolvent(l, P_i)$ is the set $\{c | KB_i \models_i c \vee \neg l\}$ of local consequences in peer P_i of the literal l.*

Definition 8 (Acquainted peers by literal ([1])). *$ACQ(l, P)$ is the set $\{P' \in V | (P, P', L_{ij}) \in E, l \in L_{ij}\}$ of peers sharing an edge labeled by L_{ij} with peer P, where $l \in L_{ij}$.*

Definition 9 (Distributed disjunction ([1])). *The distributed disjunction operator $\otimes$ in $A \otimes B$ forms clauses by disjoining all combinations of clauses in the sets A and B . For an indexed set of clause sets $\{A_i\}_i$, the notation $\otimes_{i \in \{1,2,3\}} A_i$ means $A_1 \otimes A_2 \otimes A_3$.*

Each peer can send and receive four message types during the search for a reason for the query. *Forth* messages (Algorithm 2) request the search for the empty clause $\square$ by neighboring peers. The receiving peer computes all local consequences of the literal p of the message (line 14). When two clauses are resolved, their original ancestors are added to the resolvent's OA set. An exception is the negation of the query, which is not part of the OA set. Any derivations of $\square$ with a consistent OA set are returned to the sender peer via a *back* message containing $\square$ (line 21). Local consequence clauses all of whose variables are shared with other peers are split into their individual literals and each sent to the connected peers via another *forth* message (line 34). *Back* messages (Algorithm 3) are sent back to the sender peer when $\square$ is derived. Upon receiving a *back* message, a peer stores $\square$ as a consequence of the literal of the corresponding *forth* message (line 3). If all other literals of this literal's parent clause also have a $\square$ consequence associated with them and their respective OA sets are mutually satisfiable, one back message for each combination of per-literal empty clauses is sent to the last peer in the history (line 11). *Final* messages (Algorithm 4) indicate that the exploration of a particular search branch is completed (Algorithm 2, lines 2, 4, and 35). *Prio* messages (Algorithm 5) are sent when a new reason of better priority than any reason known so far is found (Algorithm 2, lines 13 and 24 and Algorithm 3, line 13). *Prio* messages serve to update each peer's local value of $best$, which determines the priority at which peers and clauses can be safely pruned from the search space. This pruning of the search space deserves highlighting and results in a significant improvement in performance of the system.

A new query q is posed by the query peer $User$ for the computation of reasons to peer P by sending the message m($User, P, forth, \emptyset, \neg q$). Algorithm 1 runs at the query peer and computes query answers from reasons for the query and its negation by returning the answer corresponding to the

Algorithm 1 Initiating a query q to peer P

Answer_Query(q, P**)**
send m($Self, P, forth, \emptyset, \neg q$)
send m($Self, P, forth, \emptyset, q$)
$pos \leftarrow \infty$; $neg \leftarrow \infty$
while !received(m($P, Self, final, [(\neg q, Self, true)]$)) or !received(m($P, Self, final, [(q, Self, true)]$)) **do**
  for all m($P, Self, back, [(\neg q, Self, \Box)], \Box$) received **do**
    $pos \leftarrow$ min($pos, \Box.prio$)
  for all m($P, Self, back, [(q, Self, \Box)], \Box$) received **do**
    $neg \leftarrow$ min($pos, \Box.prio$)
if $pos < neg$ **then return** YES
if $neg < pos$ **then return** NO
return UNK

best-rank reason. UNK (for unknown) is returned if the best reasons for and against the query have equal rank or no reason for either exists.

3.2 Example

Returning to the bookstore example, Customer asserts ord and $paid$ and asks at the bookstore whether the book will be delivered on time. Algorithm 1 poses the query (as its negation, $\neg del_ontime$) as well as the opposite query (also as its negation, del_ontime). $\neg del_ontime$ generates the local consequence $\neg pmt_rcvd \vee \neg ord$ which is split into its individual literals. A *forth* message containing $\neg pmt_rcvd$ is sent to Visa, CCOne, and CCTwo. Visa generates $\neg paid$ which resolves with $paid$ to the empty clause at Customer. The empty clause with its OA set is passed back to Visa and then to Bookstore via *back* messages. $\neg pmt_rcvd$ generates no consequences in CCOne or CCTwo, and final messages are sent for these branches. The $\neg ord$ of the consequence in Bookstore is sent as a *forth* message to Customer where it resolves with ord to the empty clause and is passed back as a *back* message. The empty clauses for $\neg pmt_rcvd$ and $\neg ord$ are merged back together at the bookstore and the resulting OA set $\{\neg pmt_rcvd \vee \neg ord \vee del_ontime, pmt_rcvd \vee \neg paid, paid, ord\}$ is determined to be consistent. Thus a reason for del_ontime is found with rank 5 (max over the three peers contributing clauses to the reason). Similarly the empty clause can be derived from del_ontime using Bookstore, CCOne, and Customer, resulting in a reason of rank 10 for $\neg del_ontime$. Since del_ontime is inferred with better rank than $\neg del_ontime$, YES is returned as the answer.

3.3 Analysis and Discussion

We proved various soundness and completeness results for the algorithm and restate the most important theorems here. Detailed proofs and further theorems can be found in [5]. To prove the soundness and completeness of Algorithm 1 with respect to distributed entailment, we first establish that the message-passing algorithm finds the best-rank reasons for a query. To this end, Theorems 1 and 2 establish that the message-passing algorithm without priority-based pruning is sound and complete for computing reasons.

Theorem 1 (Soundness wrt. computing reasons). *Given a $\mathcal{P}$-inconsistent PQAS $\mathcal{P}$ and a peer P, for every $\Box$ in a*

Algorithm 2 *Forth* message algorithm

ReceiveForthMessage(m($Sender, Self, forth, hist, p$**))**
if $(p, Self, _) \in hist$ or $p \in KB_{Self}$ **then**
  send m($Self, Sender, final, [(p, Self, true)|hist]$)
else if $p.prio > best$ or $Self.prio > best$ **then**
  send m($Self, Sender, final, [(p, Self, true)|hist]$)
else
  if $(\neg p, _, _) \in hist$ **then**
    $hist$ is of the form $[(l', _, c')|hist']$
    if $c'.OA$ is SAT **then**
      $\Box$ a new empty clause; $\Box.OA \leftarrow c'.OA$; $\Box.prio \leftarrow c'.prio$
      send m($Self, Sender, back, [(p, Self, \Box)|hist], \Box$)
      if $hist = \emptyset$ and $\Box.prio < best$ **then**
        send m($Self, Sender, prio, \Box.prio$)
  LOCAL($Self$) $\leftarrow \{p\} \cup Resolvent(p, Self)$
  for all $c \in$ LOCAL($Self$) **do**
    let $\{c_i^*\}_i$ be the set of clauses from KB_{Self} that went into c
    $c.OA \leftarrow \bigcup_i c_i^*.OA$; $c.prio \leftarrow \max_i(p.prio, Self.prio)$
  LOCAL($Self$) $\leftarrow \{c \in$LOCAL($Self$)$|c.prio \leq best\}$
  $temp_min \leftarrow \infty$
  for all $\Box \in$ LOCAL($Self$) s.t. $\Box.OA$ is SAT **do**
    send m($Self, Sender, back, [(p, Self, \Box)|hist], \Box$)
    $temp_min \leftarrow$ min($temp_min, \Box.prio$)
  if $hist = \emptyset$ and $temp_min < best$ **then**
    send m($Self, Sender, prio, temp_min$)
  LOCAL($Self$) $\leftarrow \{c \in$ LOCAL($Self$) $|c \neq \Box$, all literals in c shared$\}$
  if LOCAL($Self$) $= \emptyset$ **then**
    send m($Self, Sender, final, [(p, Self, true)|hist]$)
  for all $c \in$ LOCAL($Self$) **do**
    for all $l \in c$ **do**
      CONS($l, [(p, Self, c)|hist]$) $\leftarrow \emptyset$
      $ACQ^* \leftarrow \{P' \in ACQ(l, Self)|P'.prio \leq best\}$
      for all $RP \in ACQ^*$ **do**
        FINAL($l, [(p, Self, c)|hist], RP$) $\leftarrow false$
        send m($Self, RP, forth, [(p, Self, c)|hist], l$)
  if no *forth* message sent **then**
    send m($Self, Sender, final, [(p, Self, true)|hist]$)

Algorithm 3 *Back* message algorithm

ReceiveBackMessage(m($Sender, Self, back, hist, \Box^*$**))**
$hist$ is of the form $[(l', Sender, \Box^*), (p, Self, c)|hist']$
CONS($l', [(p, Self, c)|hist']$) $\leftarrow$ CONS($l', [(p, Self, c)|hist']$) $\cup \Box^*$
RESULT $\leftarrow (\otimes_{l \in c \setminus \{l'\}}$CONS($l, [(p, Self, c)|hist']$)$) \otimes \{\Box^*\}$
for all $\Box \in$ RESULT s.t. $\Box$ contains an empty clause consequence for each $l \in c$ **do**
  let $\{\Box_i^*\}_i$ be the set of empty clauses that went into $\Box$
  $\Box.OA \leftarrow \bigcup_i \Box_i^*.OA$; $\Box.prio \leftarrow \max_i(\Box_i^*.prio)$
RESULT $\leftarrow \{\Box \in$ RESULT$|\Box.prio \leq best$ and $\Box.OA$ is SAT$\}$
if $hist' = \emptyset$ **then** $U \leftarrow User$ **else** $U \leftarrow$ the first peer P' of $hist'$
for all $\Box' \in$ RESULT **do**
  send m($Self, U, back, [(p, Self, c)|hist'], \Box'$)
if $hist = \emptyset$ and $\min_{\Box \in \mathrm{RESULT}}(\Box.prio) < best$ **then**
  send m($Self, Sender, prio, \min_{\Box \in \mathrm{RESULT}}(\Box.prio)$)

Algorithm 4 *Final* message algorithm

ReceiveFinalMessage(m($Sender, Self, final, hist$**))**
$hist$ is of the form $[(l', Sender, true), (p, Self, c)|hist']$
FINAL($l', [(p, Self, c)|hist'], Sender$) $\leftarrow true$
if $\forall c^* \in$ LOCAL($Self$) and $\forall l \in c^*$, FINAL($l, [(p, Self, c^*)|hist'], _$) $= true$ **then**
  if $hist' = \emptyset$ **then** $U \leftarrow User$ **else** $U \leftarrow$ the first peer P' of $hist'$
  send m($Self, U, final, [(p, Self, true)|hist']$)

Algorithm 5 *Prio* message algorithm

```
ReceivePrioMessage(m(Sender, Self, prio, x))
if x < best then
    best ← x
    for all RP ∈ ACQ(_, Self) s.t. RP ≠ Sender do
        m(Self, RP, prio, best)
```

message $m(P', P, back, [(\neg q, P', \square)], \square)$ *that* P *receives after sending* $m(P, P', forth, \emptyset, \neg q)$*, there is a set of formulas* Σ *s.t.* $\mathcal{P} \models_i^{\Sigma} q$*.*

Theorem 2 (Completeness wrt. computing reasons). *The following holds for Algorithms 2, 3, and 4 without priority-based pruning. Given a* $\mathcal{P}$*-inconsistent PQAS* $\mathcal{P}$ *and a peer* P_i*, if* $\mathcal{P} \models_i^{\Sigma} q$ *for some set of formulas* Σ*, then sending* $m(P, P', forth, \emptyset, \neg q)$ *results in receiving* $m(P', P, back, [(\neg q, P', \square)], \square)$*, where* P' *is a peer of* $\mathcal{P}$ *whose signature contains* q*.*

Clearly, if the message-passing algorithm is sound and complete for finding reasons, it finds the best-rank reason for the query. It remains to show that the branch containing the best-rank reason is never pruned due to its priority.

Theorem 3 (Pruning by priority limit). *The following hold for a PQAS running Algorithms 2, 3, 4, and 5. (i) Whenever a peer updates its local value of* $best$*, there exists a reason for the original query or its negation with priority* $best$*. (ii) Forth messages and acquainted peers ignored by a peer* P *cannot result in a reason for the original query or its negation with better or equal rank than the current value of* $best$*.*

Theorem 4 (Termination (following [1])). *If a peer* P *sends a message* $m(P, P', forth, \emptyset, \neg q)$*, it will eventually receive a message* $m(P', P, final, [(\neg q, P', true)])$*.*

Since Algorithm 1 picks the best-rank reason among the reasons for the query and its negation and the message-passing algorithm terminates, and since the message-passing algorithm is guaranteed to find the best-rank reasons for the query and its negation, Algorithm 1 is sound and complete with respect to distributed entailment.

Theorem 5 (Soundness and completeness wrt. distributed entailment). *Given a PQAS* $\mathcal{P}$ *and a peer* P*, Answer_Query*(q, P) *of Algorithm 1 returns* YES *iff* $\mathcal{P} \models_D q$*,* NO *iff* $\mathcal{P} \models_D \neg q$*, and* UNK *otherwise.*

Pruning and Ordering Heuristic: Time can be saved by the message-passing algorithm when searching for the best-rank reasons for a query and its negation simultaneously by exploiting the preference ordering over the peers and the resulting priority ordering over formulas. Since we are only interested in the best-rank reason, and since clauses and peers with worse priority than the currently best known reason are pruned away, it generally pays off for each peer to process messages with consequences of better priority first. This technique ensures that better-rank reasons are found earlier and the priority limit $best$ is updated more quickly, resulting in a larger part of the search space being pruned. Furthermore, the best-rank reason will be one of the first reasons found (although not necessarily the very first one as the message-passing algorithm runs concurrently across distributed peers). In Section 5, we present empirical results illustrating the effectiveness of this priority-ordering heuristic.

Consistency: Before sending an empty clause representing a reason in a *back* message, the algorithm needs to check the satisfiability of the clause's OA set, since this set could be inconsistent. This may be done by either a call to a SAT solver or by comparing the clauses in the OA set against cached *nogoods* in the framework of [7]. Both approaches have their merits—one needs no precomputed nogoods and the other may amortize their computation over multiple queries.

In the special case of a $\mathcal{P}$-consistent PQAS the computation of query answers is less costly. Since no contradiction can be derived in such a system, a reason only exists either for the query, its negation, or neither, and the message-passing algorithm for finding reasons is sound and complete for query answering according to distributed entailment. Furthermore, the satisfiability of clause sets is guaranteed in a $\mathcal{P}$-consistent system and OA sets need not be tracked nor their consistency checked. Priorities need not be tracked if trying to find the answer to a query only. In this case, the first reason for the query found will suffice and the algorithm may terminate. If the best-rank reason for the answers is of interest (for example, if priorities represent reliability), priorities must be tracked, but only the query and not its negation asked in a $\mathcal{P}$-consistent system.

4 Global Closed-world Reasoning

Our definition of a PQAS allows for arbitrary local consequence relations. For example, a database or logic-program peer might have a non-classical consequence relation that makes a local closed-world assumption (CWA) [14]. Nevertheless, in many distributed reasoning applications, we may wish to perform *global* closed-world reasoning. Our challenge is to perform global closed-world reasoning in a P2P setting with peers some or all of which employ some form of local closed-world reasoning.

Consider again the bookstore example. CCTwo has better priority than Visa and now makes a local CWA. CCTwo states that payment is received only if the customer paid and her credit card is confirmed ($conf$). This condition is not met, so CCTwo entails $\neg pmt_rcvd$ following CWA. Although Visa has worse priority, we would still like to entail del_ontime at Bookstore by *global closed-world entailment* since the customer would choose to pay using her Visa card if doing so allows the payment to go through.

This behavior in the bookstore example could be characterized as closed-world reasoning from the global theory. However, regular CWA entailment results in an inconsistent set of entailed formulas in the presence of positive prime implicates, i.e. minimal clauses of only positive literals that are globally entailed. Consider, for example, the theory that only consists of the clause $p \vee q$. Under Reiter's regular CWA, both $\neg p$ and $\neg q$ are entailed by the CWA and together contradict the original clause of the theory. In order to resolve this artificial inconsistency, Minker introduced the generalized closed-world assumption (GCWA), under which

only those literals are entailed negatively which are not entailed positively and occur in no minimal positive clause entailed by the theory [12]. In the following, we review the formal definitions of CWA and GCWA and use them to define *generalized global closed-world entailment* for PQASs.

4.1 Definitions

We define generalized global closed-world entailment (GGCW) in a PQAS in which each peer makes a local CWA as GCWA entailment from the global theory $\bigcup_{i=1}^{n} KB_i$ in order to avoid problems with minimal positive clauses. Definition 10 reviews regular closed-world entailment which each peer uses as its local consequences relation. Definition 11 reviews generalized closed-world entailment as needed to define GGCW entailment in Definition 12.

Definition 10 (Closed-world entailment ($\models_{CW}$)). *Let KB be a consistent propositional theory. Let $KB^+ = KB \cup \{\neg p | p$ is atomic and $KB \not\models p\}$. KB entails a formula ϕ by closed-world entailment, denoted $KB \models_{CW} \phi$, iff $KB^+ \models \phi$.*

Definition 11 (Generalized closed-world entailment ($\models_{GC}$)). *Let KB be a consistent propositional theory. Let $KB^* = KB \cup \{\neg p | p$ is atomic and p occurs in no positive prime implicate of $KB\}$. KB entails a formula ϕ by generalized closed-world entailment, denoted $KB \models_{GC} \phi$, iff $KB^* \models \phi$.*

Given a PQAS with a consistent global theory and in which each peers makes a local CWA, we define generalized global closed-world entailment as entailment according to the GCWA wrt. the PQAS's global theory.

Definition 12 (Generalized global closed-world entailment ($\models_{GGC}$)). *Let $\mathcal{P}$ be a PQAS in which each peer locally makes the closed-world assumption (CWA) and the global theory is satisfiable. Then $\mathcal{P}$ entails a formula ϕ by the global generalized closed-world assumption, denoted $\mathcal{P} \models_{GGC} \phi$, iff $\bigcup_{i=1}^{n} KB_i \models_{GC} \phi$.*

4.2 GGCW Query Answering

We can achieve GGCW query answering with *no* modification to our algorithms. For the purposes of this paper (as in Definition 12), we assume that each peer performs reasoning that is equivalent to classical entailment with a local CWA. We also assume maximal connectivity of peers—that if they share variables, they are reflected in their links. Both conditions hold in our example of Figure 1.

The trick that makes this work is to set the priority of any local closed-world consequence (such as $\neg pmt_rcvd$ in CCTwo) to a priority $\mathcal{C}$ that is *worse* than that of *every* peer in the system. Applying our existing message-passing algorithms now results in query answering according to GGCW entailment. Consider what happens in the bookstore example. The first clause of Bookstore, together with the CWA consequence $\neg pmt_rcv$ from CCTwo and ord from Customer, still produces a reason for $\neg del_ontime$. The rank of this reason will be worse than the priority of any peer due to the CWA consequence from CCTwo. A reason for del_ontime with rank 5 is still produced and will override the reason for $\neg del_ontime$ due to its better rank. More generally, *any* reason which does not contain local closed-world consequences will override *any* reason which relies on a local CWA by design. Theorem 6 formalizes this main result.

Theorem 6 (Global generalized closed-world query answering from consistent theories). *Let $\mathcal{P} = (P, G)$ be a PQAS with a satisfiable global knowledge base $\bigcup_{i=1}^{n} KB_i$ such that if for any two peers P_i and P_j, $L_{ij} = L_i \cap L_j$ is non-empty, $(i, j, L_{ij}) \in E$ and $Cons_i$ is the CWA entailment relation $\models_{CW}$ s.t. CWA-consequences are assigned the priority $\mathcal{C}$, which is lower than any peer's priority but higher than the priority represented by infinity. For a literal q, Answer_Query(q, P) of Algorithm 1 returns YES iff $\mathcal{P} \models_{GGC} q$, NO iff $\mathcal{P} \models_{GGC} \neg q$, and UNK otherwise.*

Notice how the algorithm avoids issues with the literals of positive prime implicates and thereby adheres to the GCWA from the global theory. Consider a positive prime implicate $p \vee q$ of the global theory and the query q. Since all peers make a local CWA, some peers will produce $\neg p$ and $\neg q$ with priority $\mathcal{C}$. The $\neg p$ is used with $p \vee q$ to derive q positively also with priority $\mathcal{C}$ and UNK is the answer for the query q as required by generalized global closed-world entailment.

It follows from Theorem 6 that in a PQAS in which all peers' local KBs only contain Horn clauses, the algorithm achieves query answering according to the regular CWA from the global theory (denoted $\mathcal{P} \models_{GCW} q$ in Corollary 1 below). More generally one can imagine arbitrary user-defined policies for ranking specific peers' local closed-world consequences relative to other peers, for example adhering to the partial preference ordering over peers. Such arbitrary policies are equally realizable within our framework.

Corollary 1 (Global closed-world query answering from consistent Horn theories). *Let $\mathcal{P} = (P, G)$ be a PQAS which satisfies the conditions of Theorem 6 and for all of whose peers P_i, KB_i only contains Horn clauses. Answer_Query(q, P) of Algorithm 1 returns YES iff $\mathcal{P} \models_{GCW} q$ and NO iff $\mathcal{P} \not\models_{GCW} q$, where q is a literal.*

5 Implementation and Experiments

Our implementation simulates a PQAS on a single machine. There is a message queue for each peer, and peers take turns to process one message each. The default message queue is a first-in first-out (FIFO) queue, ensuring that messages are processed in the order received. The ordering heuristic uses a priority message queue, which is sorted by the priorities of clauses and peers of the messages and returns better-priority messages first. We executed experiments to evaluate the effectiveness of our priority-ordering heuristic and our pruning strategy. Our pruning strategy prunes consequences and peers of worse priority than the currently best known reason. We ran three variations of the algorithm querying each proposition of each of a set of several hundred randomly generated PQAS instances. Each instance had 4–20 peers with 4–8 clauses per peer, 1–6 shared propositions per peer, and a total of 8–96 distinct propositions. An incomplete attempt to answer a query by a given variation of the algorithm was aborted after ten minutes. Out

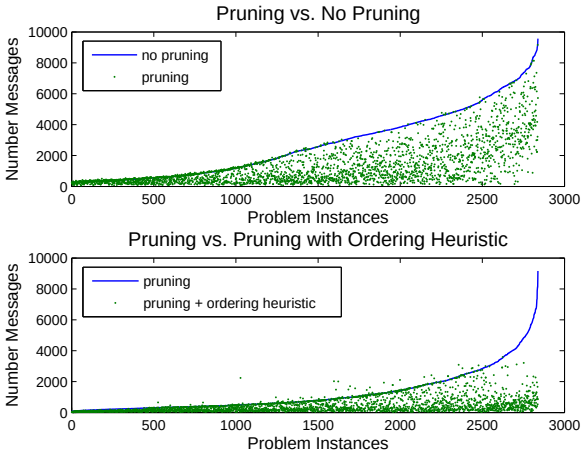

Figure 2: Total number of messages sent

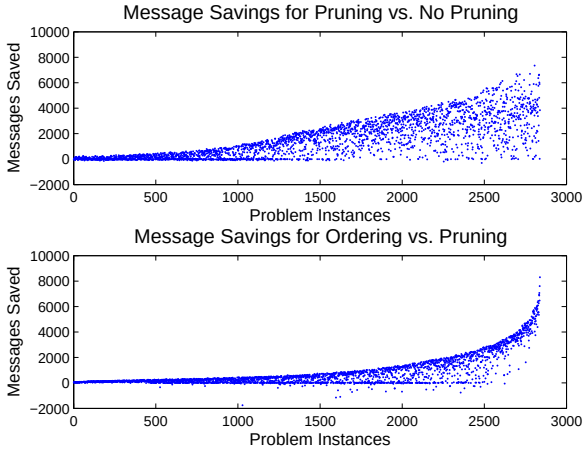

Figure 3: Number of messages saved

of all queries given, the naive, pruning, and ordering methods solved 1341, 2292, and 2842 queries, respectively. The average number of messages required to solve a query for the three methods were 2735, 1266, and 456 messages, respectively. Figures 2 and 3 show the results for individual queries in terms of the number of messages passed throughout the system to solve a query. The top graph of Figure 2 contrasts the total number of messages sent until termination of the naive algorithm (without pruning) with the number of messages used when pruning. Problem instances are sorted by number of messages used by the naive approach. The bottom graph of Figure 2 compares the number of messages sent by the heuristic version with the pruning-only version and is sorted by the hardness of problems for the latter. We chose the number of messages sent to answer a query as a performance measure as in a real-world distributed system, messages would be sent over a network, creating a bottleneck for the system. In both graphs only non-trivial queries that have been solved by at least one of the three methods are shown. A non-trivial query is one that took the naive algorithm at least 300 messages to solve. Queries that took the naive algorithm less than 300 messages to solve took on average 52, 52, and 46 messages to solve for the naive, pruning, and ordering versions of the algorithm, respectively, so ignoring such trivial queries does not hurt the evaluation of the naive approach in the comparison with the other two approaches. Figure 3 shows the number of messages saved by the pruning and priority-ordering techniques compared to not using them for the same set of queries (and in the same order) as in Figure 2.

The graphs show that pruning can lead to a drastic reduction in the number of messages generated over the naive algorithm, and in turn, that the ordering heuristic in conjunction with pruning often beats pruning alone. In a large number of cases, the savings due to both pruning and ordering are exponential. In some cases there are no or almost no savings, and in most cases the savings are somewhere in-between. In instances where no consequences or peers can be pruned, our pruning strategy may require slightly more messages than the naive approach (*prio* message overhead). In a few degenerate cases, the heuristic version of the algorithm can use more messages than the pruning-only version. This occurs when a medium-priority message generating a reason and $best$ update is at the front of the FIFO queue in the pruning-only algorithm, but the heuristic version processes better-priority messages that lead to no reason first, potentially generating messages of bad priority before the pruning cutoff variable $best$ is updated in Algorithm 5.

6 Discussion and Related Work

In this paper we provided a formal characterization of a P2P query answering system and a distributed entailment relation. Our system includes a user-defined preference ordering over the peers and allows for inconsistent information among them. The preference ordering is exploited not only for finding the best-rank reason for or against a query, but also for finding this reason computationally efficiently. While a preference specification as simple as the one presented in this paper is all that is necessary for many applications, our framework can be easily extended to richer specifications of preferences as well as aggregation schemes. We provided an extension to Adjiman et al.'s message-passing algorithm that computes query answers in the presence of inconsistent knowledge and, optionally, a preference ordering over peers and their knowledge. We proved our algorithm sound and complete with respect to our specification. A safe pruning technique and ordering heuristic were provided. Empirical analysis demonstrated a dramatic improvement in the algorithm using these techniques. We formally characterized global generalized closed-world query answering for PQASs with consistent global theories and provided a solution to this important problem within our PQAS framework.

This work provides the foundation for many interesting extensions. A trend in Semantic Web research is the development of rule languages such as SWRL and RuleML. We would like to apply our techniques to such Semantic Web peers. Our PQAS assumes a global signature. An inter-

esting extension of this work is to augment our techniques to address information integration. We also wish to extend our algorithms to deal with local-global closed-world reasoning for arbitrary peers and context-specific prioritization policies. Benferhat et al.'s argued entailment relation in [3], which served as a basis for our distributed entailment, satisfies the important intuition that no formula should be believed on the basis of a consistent subset of the global theory if its negation can be shown from a consistent subset with better rank. A future extension of our work may provide for an alternative distributed entailment relation which ensures that a query follows only then by distributed entailment, if the reasons it relies on cannot be defeated with better priority, even if the negation of the query itself cannot be shown at all. This idea is strongly related to Dung's work on the acceptability of arguments that are "attacked" by other arguments [10]. Finally there are some extensions that we wish to complete, including algorithms for more complex queries and for richer priority or trust aggregation models.

There is significant related work, most prominently [1, 2, 7] as discussed in the introduction. There is also related work on distributed description logics [15] and on distributed database peers [16]. Work on inconsistent knowledge concentrates on the centralized case. E.g., in addition to [7], [4] uses query rewriting to consistently answer queries to inconsistent databases with integrity constraints, while [3] compares several entailment relations for prioritized inconsistent KBs. (We exploited their notion of prioritized argued entailment in this paper.) In related work, [13] explores syntax-based approaches to belief revision. Grosof provides a framework of prioritized, inconsistent logic programs with a unique consistent answer set [11]. Priorities have also been used in [6] to achieve extended default reasoning in centralized theories. In the area of local closed-world reasoning, [8] formalized the concept of a local closed-world assumption of data-sources, allowing for limited global closed-world reasoning with databases. [9] addressed the issue of open and closed-world reasoning with rule bases or logic programs for the Semantic Web using logic program transformations. While related, none of these lines of work address the general problem that we addressed in this paper.

Acknowledgments

We are grateful to Michael Gruninger and Gerhard Brewka for valuable feedback on this work. We furthermore thank Philippe Adjiman for providing an early version of his P2PIS code which we did not end up using. We also thank Eric Hsu and Jorge Baier for their comments on an earlier draft of this paper. We gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada (NSERC).

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. *Journal of Artificial Intelligence Research*, 25:269–314, 2006.

[2] E. Amir and S. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artificial Intelligence*, 162(1-2):49–88, 2005.

[3] S. Benferhat, D. Dubois, and H. Prade. Some syntactic approaches to the handling of inconsistent knowledge bases: a comparative study part 2: the prioritized case. In E. Orłowska, editor, *Logic at Work: Essays Dedicated to the Memory of Helen Rasiowa*, volume 24, pages 437–511. Physica-Verlag, 1999.

[4] L. E. Bertossi, J. Chomicki, A. Cortés, and C. Gutiérrez. Consistent answers from integrated data sources. In *Proceedings of the 5th International Conference on Flexible Query Answering Systems (FQAS)*, pages 71–85, London, UK, 2002.

[5] A. Binas and S. McIlraith. Distributed query answering in peer-to-peer reasoning systems. Technical report, Department of Computer Science, University of Toronto, 2007.

[6] G. Brewka and T. Eiter. Prioritizing default logic. In *Intellectics and Computational Logic*, pages 27–45. Kluwer Academic Publishers, 2000.

[7] P. Chatalic, G.-H. Nguyen, and M.-C. Rousset. Reasoning with inconsistencies in propositional peer-to-peer inference systems. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, pages 352–357, 2006.

[8] A. Cortés-Calabuig, M. Denecker, O. Arieli, B. Van Nuffelen, and M. Bruynooghe. On the local closed-world assumption of data-sources. In *BNAIC*, pages 333–334, 2005.

[9] C. V. Damásio, A. Analyti, G. Antoniou, and G. Wagner. Supporting open and closed world reasoning on the web. In *PPSWR*, pages 149–163, 2006.

[10] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[11] B. N. Grosof. Courteous logic programs: Prioritized conflict handling for rules. Technical Report RC 20836, IBM, 1997.

[12] J. Minker. On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science*, volume 138, pages 292–308. Springer, Berlin, 1982.

[13] B. Nebel. Syntax-based approaches to belief revision. In P. Gärdenfors, editor, *Belief Revision*, volume 29, pages 52–88. Cambridge University Press, 1992.

[14] R. Reiter. On closed world data bases. In Herve Gallaire and Jack Minker, editors, *Logic and data bases*, pages 55–76. Plenum Press, New York, NY, 1978.

[15] L. Serafini and A. Tamilin. DRAGO: Distributed reasoning architecture for the semantic web. Technical Report T04-12-05, ITC-irst, 2004.

[16] I. Zaihrayeu. *Towards Peer-to-Peer Information Management Systems*. PhD thesis, University of Trento, 2006.