P2P Deductive Databases: Well Founded Semantics and Distributed Computation

L. Caroprese and E. Zumpano(✉)

DIMES, University of Calabria, 87036 Rende, Italy
{l.caroprese,e.zumpano}@dimes.unical.it

Abstract. This paper stems from previous works of the same authors in which a declarative semantics for *Peer-to-Peer* (P2P) systems, defined in terms of *Preferred Weak Models*, is proposed. Under this semantics only facts not making the local databases inconsistent can be imported. As in the general case a P2P system may admit many preferred weak models whose computational complexity is prohibitive, the paper looks for a more pragmatic solution. It assigns to a P2P system its *Well Founded Model*, a partial deterministic model that captures the intuition that if an atom is true in a preferred weak model, but it is false in another one, then it is undefined in the well founded model. The paper presents a distributed algorithm for the computation of the well founded model and a system prototype.

1 Introduction

Several proposals considering the issue of managing the coordination, the integration of information [4–6,9,13,18,20] as well as the computation of queries in a P2P system have been proposed in the literature [2,11]. This paper follows the proposal in [6–10] in which a different interpretation of mapping rules has led to the proposal of a semantics for a P2P system defined in terms of *Preferred Weak Models*. Under this semantics only facts not making the local databases inconsistent can be imported, and the preferred weak models are the consistent scenarios in which peers import maximal sets of facts not violating constraints.

Example 1. Consider a P2P in which the peer: (i) $\mathcal{P}_3$ contains two atoms: $r(a)$ and $r(b)$; (ii) $\mathcal{P}_2$ imports data from $\mathcal{P}_3$ using the (mapping) rule $q(X) \hookleftarrow r(X)$[1]. Moreover imported atoms must satisfy the constraint $\leftarrow q(X), q(Y), X \neq Y$ stating that the relation q may contain at most one tuple; (iii) $\mathcal{P}_1$ imports data from $\mathcal{P}_2$, using the (mapping) rule $p(X) \hookleftarrow q(X)$. $\mathcal{P}_1$ also contains the rules $s \leftarrow p(X)$ stating that s is *true* if the relation p contains at least one tuple, and $t \leftarrow p(X), p(Y), X \neq Y$, stating that t is *true* if the relation p contains at least two distinct tuples.

The intuition is that, with $r(a)$ and $r(b)$ *true* in $\mathcal{P}_3$, either $q(a)$ or $q(b)$ could be imported in $\mathcal{P}_2$ and, consequently, only one tuple is imported in the relation

[1] Please, note the special syntax we use for mapping rules.

© Springer International Publishing AG 2017
M. Kirikova et al. (Eds.): ADBIS 2017, CCIS 767, pp. 91–99, 2017.
DOI: 10.1007/978-3-319-67162-8_11

p of the peer $\mathcal{P}_1$. Note that whatever is the derivation in $\mathcal{P}_2$, s is derived in $\mathcal{P}_1$ while t is not derived. Therefore, s and t are, respectively, *true* and *false* in $\mathcal{P}_1$. $\square$

A P2P system may admits many preferred weak models and the computational complexity is prohibitive. Therefore, a more pragmatic solution is needed. The paper first introduces a rewriting technique that allows modeling a P2P system $\mathcal{PS}$ as a unique logic program, $Rew_t(\mathcal{PS})$, that can be used as a computational vehicle to calculate the semantics of the P2P system; then presents the *Well Founded Model Semantics*, that allows obtaining a deterministic model whose computation is polynomial time.

Moreover, the paper presents a distributed algorithm for the computation of the well founded model and provides some details on the implementation of a system prototype for query answering in P2P network based on the proposed semantics.

2 P2P Systems: Syntax and Semantics

Familiarity is assumed with deductive database [1], logic programming, stable models, head cycle free (HCF) program, well founded model and computational complexity [3,12,16,17,19]. A *(peer) predicate symbol* is a pair $i : p$, where i is a *peer identifier* and p is a predicate symbol. A *(peer) atom* is of the form $i : A$, where i is a *peer identifier* and A is a standard atom. A *(peer) literal* is a peer atom $i : A$ or its negation *not* $i : A$. A conjunction $i : A_1, \dots, i : A_m, not\ i : A_{m+1}, \dots, not\ i : A_n, \phi$, where ϕ is a conjunction of built-in atoms, will be also denoted as $i : \mathcal{B}$, with $\mathcal{B}$ equals to $A_1, \dots, A_m, not\ A_{m+1}, \dots, not\ A_n, \phi$.

A *(peer) rule* can be of one of the following three types:

- STANDARD RULE. It is of the form $i : H \leftarrow i : \mathcal{B}$, where $i : H$ is an atom and $i : \mathcal{B}$ is a conjunction of atoms and built-in atoms.
- INTEGRITY CONSTRAINT. It is of the form $\leftarrow i : \mathcal{B}$, where $i : \mathcal{B}$ is a conjunction of literals and built-in atoms.
- MAPPING RULE. It is of the form $i : H \hookleftarrow j : \mathcal{B}$, where $i : H$ is an atom, $j : \mathcal{B}$ is a conjunction of atoms and built-in atoms and $i \neq j$.

$i : H$ is called *head* while $i : \mathcal{B}$ (resp. $j : \mathcal{B}$) is called *body*. Negation is allowed just in the body of integrity constraints. The definition of a predicate $i{:}p$ consists of the set of rules in whose head the predicate symbol $i{:}p$ occurs. A predicate can be of three different kinds: *base predicate*, *derived predicate* and *mapping predicate*. A base predicate is defined by a set of ground facts; a derived predicate is defined by a set of standard rules and a mapping predicate is defined by a set of mapping rules. An atom $i : p(X)$ is a *base atom* (resp. *derived atom*, *mapping atom*) if $i : p$ is a base predicate (resp. standard predicate, mapping predicate). Given an interpretation M, $M[\mathcal{D}]$ (resp. $M[\mathcal{LP}]$, $M[\mathcal{MP}]$) denotes the subset of base atoms (resp. derived atoms, mapping atoms) in M.

Definition 1. P2P System. A *peer* $\mathcal{P}_i$ is a tuple $\langle \mathcal{D}_i, \mathcal{LP}_i, \mathcal{MP}_i, \mathcal{IC}_i \rangle$, where (i) $\mathcal{D}_i$ is a set of facts (*local database*); (ii) $\mathcal{LP}_i$ is a set of standard rules; (iii) $\mathcal{MP}_i$ is a set of mapping rules and (iv) $\mathcal{IC}_i$ is a set of constraints over predicates defined by $\mathcal{D}_i$, $\mathcal{LP}_i$ and $\mathcal{MP}_i$. A *P2P system* $\mathcal{PS}$ is a set of peers $\{\mathcal{P}_1, \dots, \mathcal{P}_n\}$. $\Box$

Given a P2P system $\mathcal{PS} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$, where $\mathcal{P}_i = \langle \mathcal{D}_i, \mathcal{LP}_i, \mathcal{MP}_i, \mathcal{IC}_i \rangle$, $\mathcal{D}, \mathcal{LP}, \mathcal{MP}$ and $\mathcal{IC}$ denote, respectively, the global sets of ground facts, standard rules, mapping rules and integrity constraints, i.e. $\mathcal{D} = \bigcup_{i \in [1..n]} \mathcal{D}_i$, $\mathcal{LP} = \bigcup_{i \in [1..n]} \mathcal{LP}_i$, $\mathcal{MP} = \bigcup_{i \in [1..n]} \mathcal{MP}_i$ and $\mathcal{IC} = \bigcup_{i \in [1..n]} \mathcal{IC}_i$. In the rest of this paper, with a little abuse of notation, $\mathcal{PS}$ will be also denoted both with the tuple $\langle \mathcal{D}, \mathcal{LP}, \mathcal{MP}, \mathcal{IC} \rangle$ and the set $\mathcal{D} \cup \mathcal{LP} \cup \mathcal{MP} \cup \mathcal{IC}$.

We now review the *Preferred Weak Model* semantics in [6,7]. For each peer $\mathcal{P}_i = \langle \mathcal{D}_i, \mathcal{LP}_i, \mathcal{MP}_i, \mathcal{IC}_i \rangle$, the set $\mathcal{D}_i \cup \mathcal{LP}_i$ is a *positive normal program*, thus it admits just *one minimal model* that represents the *local knowledge* of $\mathcal{P}_i$. It is assumed that each peer is *locally consistent*, i.e. its local knowledge satisfies $\mathcal{IC}_i$ (i.e. $\mathcal{D}_i \cup \mathcal{LP}_i \models \mathcal{IC}_i$). Therefore, inconsistencies may be introduced just when the peer imports data. The intuitive meaning of a mapping rule $i : H \hookleftarrow j : \mathcal{B} \in \mathcal{MP}_i$ is that if the body conjunction $j : \mathcal{B}$ is *true* in the source peer $\mathcal{P}_j$ the atom $i : H$ can be imported in $\mathcal{P}_i$ only if it does not imply (directly or indirectly) the violation of some constraint in $\mathcal{IC}_i$.

Given a mapping rule $r = H \hookleftarrow \mathcal{B}$, the corresponding standard logic rule $H \leftarrow \mathcal{B}$ will be denoted as $St(r)$. Analogously, given a set of mapping rules $\mathcal{MP}$, $St(\mathcal{MP}) = \{St(r) \mid r \in \mathcal{MP}\}$ and given a P2P system $\mathcal{PS} = \mathcal{D} \cup \mathcal{LP} \cup \mathcal{MP} \cup \mathcal{IC}$, $St(\mathcal{PS}) = \mathcal{D} \cup \mathcal{LP} \cup St(\mathcal{MP}) \cup \mathcal{IC}$.

Given an interpretation M, an atom H and a conjunction of atoms $\mathcal{B}$:

- $val_M(H \leftarrow \mathcal{B}) = val_M(H) \geq val_M(\mathcal{B})$,
- $val_M(H \hookleftarrow \mathcal{B}) = val_M(H) \leq val_M(\mathcal{B})$.

Therefore, if the body is *true*, the head of a standard rule *must* be *true*, whereas the head of a mapping rule *could* be *true*. Intuitively, a *weak model* M is an interpretation that satisfies all standard rules, mapping rules and constraints of $\mathcal{PS}$ and such that each atom $H \in M[\mathcal{MP}]$ (i.e. each mapping atom) is *supported* from a mapping rule $H \hookleftarrow \mathcal{B}$ whose body $\mathcal{B}$ is satisfied by M. A *preferred weak model* is a weak model containing a maximal subset of mapping atoms.

Definition 2. (Preferred) Weak Model. Given a P2P system $\mathcal{PS} = \mathcal{D} \cup \mathcal{LP} \cup \mathcal{MP} \cup \mathcal{IC}$, an interpretation M is a *weak model* for $\mathcal{PS}$ if $\{M\} = \mathcal{MM}(St(\mathcal{PS}^M))$, where $\mathcal{PS}^M$ is the program obtained from *ground*$(\mathcal{PS})$ by removing all mapping rules whose head is *false* w.r.t. M. Given two weak models M and N, M is said to *preferable* to N, and is denoted as $M \sqsupseteq N$, if $M[\mathcal{MP}] \supseteq N[\mathcal{MP}]$. Moreover, if $M \sqsupseteq N$ and $N \not\sqsupseteq M$, then $M \sqsupset N$. A weak model M is said to be *preferred* if there is no weak model N such that $N \sqsupset M$.

The set of weak models for a P2P system $\mathcal{PS}$ will be denoted by $\mathcal{WM}(\mathcal{PS})$, whereas the set of preferred weak models will be denoted by $\mathcal{PWM}(\mathcal{PS})$. $\Box$

Theorem 1. *For every consistent P2P system $\mathcal{PS}$, $\mathcal{PWM}(\mathcal{PS}) \neq \emptyset$.*

Example 2. Consider a P2P system $\mathcal{PS}$ in which the peer: $\mathcal{P}_2$ contains the facts $q(a)$ and $q(b)$, whereas $\mathcal{P}_1$ contains the mapping rule $p(X) \hookleftarrow q(X)$ and the constraint $\leftarrow p(X), p(Y), X \neq Y$. $\mathcal{WM}(\mathcal{PS})$ are: $M_0 = \{q(a), q(b)\}$, $M_1 = \{q(a), q(b), p(a)\}$ and $M_2 = \{q(a), q(b), p(b)\}$, whereas $\mathcal{PWM}(\mathcal{PS})$ are M_1 and M_2 as they import maximal sets of atoms from $\mathcal{P}_2$. □

3 Computing the Preferred Weak Model Semantics

This section presents an alternative characterization of the preferred weak model semantics, that allows to model a P2P system $\mathcal{PS}$ with a single logic program $Rew_t(\mathcal{PS})$. Let's firstly introduce some preliminaries. Given an atom $A = i : p(x)$, A^t denotes the atom $i : p^t(x)$ and A^v denotes the atom $i : p^v(x)$. A^t will be called the *testing atom*, whereas A^v will be called the *violating atom*.

Definition 3. Given a conjunction

$$\mathcal{B} = A_1, \ldots, A_h, not\ A_{h+1}, \ldots, not\ A_n, B_1, \ldots, B_k, not\ B_{k+1}, \ldots, not\ B_m, \phi \quad (1)$$

where A_i ($i \in [1..\ n]$) is a mapping atom or a derived atom, B_i ($i \in [1..\ m]$) is a base atom and ϕ is a conjunction of built in atoms, we define

$$\mathcal{B}^t = A_1^t, \ldots, A_h^t, not\ A_{h+1}^t, \ldots, not\ A_n^t, B_1, \ldots, B_k, not\ B_{k+1}, \ldots, not\ B_m, \phi \quad (2)$$

Therefore, given a negation free conjunction $\mathcal{B} = A_1, \ldots, A_h, B_1, \ldots, B_k, \ldots, \phi$, then $\mathcal{B}^t = A_1^t, \ldots, A_h^t, B_1, \ldots, B_k, \phi$. In the following, the rewriting of a P2P system is reported.

Definition 4. Rewriting of an integrity constraint. Given an integrity constraint $i = \ \leftarrow \mathcal{B}$ (that is of the form (1)) its rewriting is defined as $Rew_t(i) = \{A_1^v \vee \cdots \vee A_h^v \leftarrow \mathcal{B}^t\}$. □

If $\mathcal{B}^t$ (of the form (2)), is *true*, at least one of the violating atoms $A_1^v, \ldots, A_h^v$ is *true*. Therefore, at least one of the atoms $A_1, \ldots, A_h$ cannot be inferred.

Definition 5. Rewriting of a standard rule. Given a standard rule $s = H \leftarrow \mathcal{B}$, its rewriting is defined as $Rew_t(s) = \{H \leftarrow \mathcal{B};\ H^t \leftarrow \mathcal{B}^t;\ A_1^v \vee \cdots \vee A_h^v \leftarrow \mathcal{B}^t, H^v\ \}$. □

In order to find the mapping atoms that, if imported, generate some inconsistencies (i.e. in order to find their corresponding violating atoms), all possible mapping testing atoms are imported and the derived testing atoms are inferred. In the previous definition, if $\mathcal{B}^t$ is *true* and the violating atom H^v is *true*, then the body of the disjunctive rule is *true* and therefore it can be deduced that at least one of the violating atoms $A_1^v, \ldots, A_h^v$ is *true* (i.e. to avoid such inconsistencies at least one of atoms $A_1, \ldots, A_h$ cannot be inferred).

Definition 6. Rewriting of a mapping rule. Given a mapping rule $m = H \hookleftarrow \mathcal{B}$, its rewriting is defined as $Rew_t(m) = \{H^t \leftarrow \mathcal{B};\ H \ \leftarrow H^t, not\ H^v\ \}$. □

Intuitively, to check whether a mapping atom H generates some inconsistencies, if imported in its target peer, a testing atom H^t is imported in the same peer. Rather than violating some integrity constraint, it (eventually) generates, by rules obtained from the rewriting of standard rules and integrity constraints, the atom H^v. In this case H, cannot be inferred and inconsistencies are prevented.

Definition 7. Rewriting of a P2P system. Given a P2P system $\mathcal{PS} = \mathcal{D} \cup \mathcal{LP} \cup \mathcal{MP} \cup \mathcal{IC}$, then: $Rew_t(\mathcal{MP}) = \bigcup_{m \in \mathcal{MP}} Rew_t(m)$, $Rew_t(\mathcal{LP}) = \bigcup_{s \in \mathcal{LP}} Rew_t(s)$, $Rew_t(\mathcal{IC}) = \bigcup_{i \in \mathcal{IC}} Rew_t(i)$ and $Rew_t(\mathcal{PS}) = \mathcal{D} \cup Rew_t(\mathcal{LP}) \cup Rew_t(\mathcal{MP}) \cup Rew_t(\mathcal{IC})$ □

Definition 8. Total Stable Model. Given a P2P system $\mathcal{PS}$ and a stable model M for $Rew_t(\mathcal{PS})$, the interpretation obtained by deleting from M its violating and testing atoms, denoted as $\mathcal{T}(M)$, is a *total stable model* of $\mathcal{PS}$. The set of total stable models of $\mathcal{PS}$ is denoted as $\mathcal{TSM}(\mathcal{PS})$. □

Example 3. Consider the P2P system $\mathcal{PS}$ in Example 2. From Definition (7) we obtain:

$$\begin{aligned} Rew_t(\mathcal{PS}) = \{&q(a);\ q(b);\ p^t(X) \leftarrow q(X);\ p(X) \leftarrow p^t(X), not\ p^v(X);\\ &p^v(X) \vee p^v(Y) \leftarrow p^t(X), p^t(Y), X \neq Y\} \end{aligned}$$

The stable models of $Rew_t(\mathcal{PS})$ are: $M_1 = \{q(a), q(b), p^t(a), p^t(b), p^v(a), p(b)\}$, $M_2 = \{q(a), q(b), p^t(a), p^t(b), p(a), p^v(b)\}$. Then, the total stable models of $\mathcal{PS}$ are $\mathcal{TSM}(\mathcal{PS}) = \{\{q(a), q(b), p(b)\}, \{q(a), q(b), p(a)\}\}$. □

Theorem 2. *For every P2P system $\mathcal{PS}$, $\mathcal{TSM}(\mathcal{PS}) = \mathcal{PWM}(\mathcal{PS})$.* □

4 Well Founded Semantics and Distributed Computation

A P2P system may admit many preferred weak models whose computational complexity has been shown to be prohibitive [6,7]. Therefore, a deterministic model whose computation is guaranteed to be polynomial time is needed. In more details, the rewriting presented in Sect. 3 allows modeling a P2P system by a single disjunctive logic program. By assuming that this program is HCF, it can be rewritten into an equivalent normal program for which a *Well Founded Model Semantics* can be adopted. Such a semantics allows to compute in polynomial time a deterministic model describing the P2P system, by capturing the intuition that if an atom is *true* in a total stable (or preferred weak) *model* of $\mathcal{PS}$ and is *false* in another one, then it is *undefined* in the well founded model.

Theorem 3. *Let $\mathcal{PS} = \mathcal{D} \cup \mathcal{LP} \cup \mathcal{MP} \cup \mathcal{IC}$ be a P2P system, then $Rew_t(\mathcal{PS})$ is HCF iff there are not two distinct atoms occurring in the body of a rule in $ground(\mathcal{LP} \cup \mathcal{IC})$ mutually dependent by positive recursion.* □

we assume each P2P system $\mathcal{PS}$ is s.t. $Rew_t(\mathcal{PS})$ is HCF. $\mathcal{PS}$ will be called as *HCF P2P system*. From previous hypothesis, it follows that $Rew_t(\mathcal{PS})$ can be normalized as $\mathcal{SM}(Rew_t(\mathcal{PS})) = \mathcal{SM}(Normalized(Rew_t\ (\mathcal{PS})))$.

Definition 9. Rewriting of an HCF P2P system. Given an HCF P2P system $\mathcal{PS}$, $Rew_w(\mathcal{PS}) = Normalized(Rew_t(\mathcal{PS}))$. □

Therefore, the preferred weak models of an HCF P2P system $\mathcal{PS}$ corresponds to the stable models of the normal program $Rew_w(\mathcal{PS})$. Next step is to adopt for $Rew_w(\mathcal{PS})$ a three-valued semantics that allows computing *deterministic models* and in particular the *well founded model.*

Definition 10. Well Founded Semantics. Given an HCF P2P system $\mathcal{PS}$ and the well founded model of $Rew_w(\mathcal{PS})$, say $\langle T, F\rangle$, the *well founded model semantics* of $\mathcal{PS}$ is given by $\langle \mathcal{T}(T), \mathcal{T}(F)\rangle$. □

Example 4. The rewriting of the HCF P2P system $\mathcal{PS}$ in Example 3 is:

$$\begin{aligned} Rew_w(\mathcal{PS}) = \{& q(a);\ q(b);\ p^t(X) \leftarrow q(X);\ p(X) \leftarrow p^t(X), not\ p^v(X);\\ & p^v(X) \leftarrow p^t(X), p^t(Y), X \neq Y, not\ p^v(Y);\\ & p^v(Y) \leftarrow p^t(X), p^t(Y), X \neq Y, not\ p^v(X)\} \end{aligned}$$

The well founded model of $Rew_w(\mathcal{PS})$ is $\langle\{q(a), q(b), p^t(a), p^t(b)\}, \emptyset\rangle$ and the well founded semantics of $\mathcal{PS}$ is given by $\langle\{q(a), q(b)\}, \emptyset\rangle$. The atoms $q(a)$ and $q(b)$ are *true*, while the atoms $p(a)$ and $p(b)$ are *undefined.* □

Theorem 4. *Let $\mathcal{PS}$ be a HCF P2P system, then deciding: (i) whether an interpretation M is a preferred weak model of $\mathcal{PS}$ is P-time; (ii) whether an atom A is* true *in some preferred weak model of $\mathcal{PS}$ is $\mathcal{NP}$-complete; (iii) whether an atom A is* true *in every preferred weak model of $\mathcal{PS}$ is co$\mathcal{NP}$-complete; (iv) whether an atom A is* true *in the well founded model of $\mathcal{PS}$ is P-time.* □

$Rew_w(\mathcal{PS})$ allows to compute the well founded semantics of $\mathcal{PS}$ in polynomial time. In this section we present a technique allowing to compute the well founded model in a distributed way. The basic idea is that each peer computes its own portion of the "unique logic program", sending to the other peers the result.

Definition 11. Let $\mathcal{PS}$= be an HCF P2P system, where $\mathcal{P}_i = \langle \mathcal{D}_i, \mathcal{LP}_i, \mathcal{MP}_i, \mathcal{IC}_i\rangle$, for $i \in [1..n]$. Then, $Rew_w(\mathcal{P}_i) = Rew_w(\mathcal{D}_i \cup \mathcal{LP}_i \cup \mathcal{MP}_i \cup \mathcal{IC}_i)$.

Previous definition allows to derive a single normal logic program for each peer. Observe that, $Rew_w(\mathcal{PS}) = \bigcup_{i\in[1..n]} Rew_w(\mathcal{P}_i)$.

Example 5. The rewritings located on peers $\mathcal{P}_1$ and $\mathcal{P}_2$ of Example 2 are:

$$\begin{aligned} Rew_w(\mathcal{P}_1) &= \{p^t(X) \leftarrow q(X);\ p(X) \leftarrow p^t(X), not\ p^v(X);\\ &\quad p^v(X) \leftarrow p^t(X), p^t(Y), not\ p^v(Y), X \neq Y;\\ &\quad p^v(Y) \leftarrow p^t(X), p^t(Y), not\ p^v(X), X \neq Y\}.\\ Rew_w(\mathcal{P}_2) &= \{\text{q(a); q(b)}\}. \end{aligned}$$
□

The idea is the following: if a peer receives a query, then it will recursively query the peer to which it is connected through mapping rules, before being able to calculate its answer. Formally, a local query submitted by a user to a peer does not differ from a remote query submitted by another peer. Once retrieved the necessary data from neighbor peers, the peer computes its well founded model and evaluates the query (either local or remote) on that model; then if the query is a remote query, the answer is sent to the requesting peer. For the sake of presentation, a partial model reports, using the syntax $[T, U]$, the sets of true and undefined atoms, instead of the sets of true and false atoms.

Example 6. Consider the P2P system in Example 2. If $\mathcal{P}_1$ receives the query $p(X)$, it submits the query $p(X)$ to the peer $\mathcal{P}_2$. Once $\mathcal{P}_2$ receives the query $q(X)$, it computes its well founded model $W_2 = [\{q(a), q(b)\}, \emptyset]$. $\mathcal{P}_1$ receives the data, populates its local database and computes its well founded model $W_1 = [\{p^t(a), p^t(b)\}, \{p^v(a), p^v(b), p(a), p(b)\}]$ and finally, evaluates the query $p(X)$ over W_1. The answer will be $[\emptyset, \{p(a), p(b)\}]$. Observe that, the peer replies providing two undefined atoms: $p(a)$ and $p(b)$. □

```
Algorithm: $ComputeAnswer_i$
input: 1) $i : q(X)$ - a query
       2) $s$ - a sender which is a peer $\mathcal{P}_k$ (remote query) or null (local query)
output: $[T, U]$ where $T$ (resp. $U$) is the set of true (resp. undefined) atoms
begin
  while true
    wait for an input $i : q(X)$ and $s$;
    $P = Rew_w(\mathcal{P}_i)$;
    for each $(i : h(X) \hookleftarrow j : b(X)) \in \mathcal{MP}_i$
       $[T, U] = ComputeAnswer_j(j : b(X), \mathcal{P}_i)$;
       $P = P \cup T \cup \{a \leftarrow not\ a \mid a \in U\}$;
    end for
    $[T_w, U_w] = ComputeWellFoundedModel(P)$;
    $answer = [\{i : q(x) \mid i : q(x) \in T_w\}, \{i : q(x) \mid i : q(x) \in U_w\}]$;
    if $isNull(s)$ then $show(answer)$;
    else $send(answer, \mathcal{P}_k)$;
    end if
  end while
end
```

We associate to a P2P system, $\mathcal{PS}$, a graph $g(\mathcal{PS})$ whose nodes are the peers of $\mathcal{PS}$ and whose edges are the pairs $(\mathcal{P}_i, \mathcal{P}_j)$ s.t. $(i : H \hookleftarrow j : \mathcal{B}) \in \mathcal{PS}$. We say that $\mathcal{PS}$ is acyclic if $g(\mathcal{PC})$ is acyclic. In order to guarantee the termination of the computation, from now on we assume that our P2P systems are acyclic. Without loss of generality, we assume that each mapping rule is of the form $i : p(X) \hookleftarrow j : q(X)$. The behavior of the peer $\mathcal{P}_i = \langle \mathcal{D}_i, \mathcal{LP}_i, \mathcal{MP}_i, \mathcal{IC}_i \rangle$ within the P2P system can be modeled by the algorithm $ComputeAnswer_i$, running on the peer $\mathcal{P}_i$. It receives in input a query $i : q(X)$ submitted by a sender s which can be another peer or a user. For each mapping rule $i : h(X) \hookleftarrow j : b(X)$, the peer $\mathcal{P}_i$ queries the peer $\mathcal{P}_j$ asking for $j : b(X)$. $\mathcal{P}_i$ will receive true and undefined tuples that will enrich the local knowledge. Observe that, true atoms will be simply inserted into P while for each undefined atom a, a rule $a \leftarrow not\ a$ allowing to infer the correct truth value for a (undefined) in the well founded model, is inserted. Once the local knowledge has been updated, the local well founded model is computed by means of the function *ComputeWellFoundedModel* and the answer for the query is extracted. If the sender is a user then the answer is simply shown, otherwise (if it is a peer) it is sent back to it.

A system prototype for query answering in a P2P network based on the proposed semantics has benn implemented. The system has been developed using Java.

The communication among peers is performed by using JXTA libraries [15]. A peer is a node in the JXTA network. JXTA defines a set of protocols providing implementation for basic and complex P2P functionalities allowing each device in the network to communicate and interact as a peer and guarantees some advantages in the development of P2P applications (e.g. it can be run on different digital devices (PCs, PDAs)). *XSB* [21] is a logic programming and deductive database system used for computing the answer to a given query using the well founded semantics. InterProlog [14] is an open source front-end that provides Java with the ability to interact with XSB engine.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Boston (1994)
2. Bertossi, L., Bravo, L.: Query answering in peer-to-peer data exchange systems. In: Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268, pp. 476–485. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30192-9_47
3. Ben-Eliyahu, R., Dechter, R.: Propositional Semantics for Disjunctive Logic Programs. In: JICSLP, pp. 813–827 (1992)
4. Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability and complexity of query answering over inconsistent and incomplete databases. In: PODS, pp. 260–271 (2003)
5. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Logical foundations of peer-to-peer data integration. In: PODS, pp. 241–251 (2004)
6. Caroprese, L., Greco, S., Zumpano, E.: A logic programming approach to querying and integrating P2P deductive databases. In: FLAIRS, pp. 31–36 (2006)
7. Caroprese, L., Molinaro, C., Zumpano, E.: Integrating and querying P2P deductive databases. In: IDEAS, pp. 285–290 (2006)
8. Caroprese, L., Zumpano, E.: Consistent data integration in P2P deductive databases. In: Prade, H., Subrahmanian, V.S. (eds.) SUM 2007. LNCS (LNAI), vol. 4772, pp. 230–243. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75410-7_17
9. Caroprese, L., Zumpano, E.: Modeling cooperation in P2P data management systems. In: An, A., Matwin, S., Raś, Z.W., Ślęzak, D. (eds.) ISMIS 2008. LNCS (LNAI), vol. 4994, pp. 225–235. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68123-6_25
10. Caroprese, L., Zumpano, E.: Handling preferences in P2P systems. In: Lukasiewicz, T., Sali, A. (eds.) FoIKS 2012. LNCS, vol. 7153, pp. 91–106. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28472-4_6
11. Franconi, E., Kuper, G., Lopatenko, A., Serafini, L.: A robust logical and computational characterisation of peer-to-peer database systems. In: Aberer, K., Koubarakis, M., Kalogeraki, V. (eds.) DBISP2P 2003. LNCS, vol. 2944, pp. 64–76. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24629-9_6

12. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of Fifth Conference on Logic Programming, pp. 1070–1080 (1998)
13. Halevy, A., Ives, Z., Suciu, D., Tatarinov, I.: Schema mediation in peer data management systems. In: Interenational Conference on Database Theory, pp. 505–516 (2003)
14. InterProlog. http://www.declarativa.com/interprolog/
15. Project JXTA. http://www.jxta.org/
16. Lone, Z., Truszczyński, M.: On the problem of computing the well-founded semantics. In: Lloyd, J., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Palamidessi, C., Pereira, L.M., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS, vol. 1861, pp. 673–687. Springer, Heidelberg (2000). doi:10.1007/3-540-44957-4_45
17. Lloyd, J.W.: Foundations of Logic Programming. Springer-Verlag, Heidelberg (1987)
18. Madhavan, J., Halevy, A.Y.: Composing mappings among data sources. In: VLDB, pp. 572–583 (2003)
19. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Boston (1994)
20. Tatarinov, I., Halevy., A.: Efficient Query reformulation in peer data management systems. In: SIGMOD, pp. 539–550 (2004)
21. XSB Project. http://xsb.sourceforge.net