ONLINE APPENDIX
for the paper

Consistency and Trust in Peer Data Exchange Systems

published in Theory and Practice of Logic Programming

LEOPOLDO BERTOSSI
Carleton University, School of Computer Science, Ottawa, Canada.
bertossi@scs.carleton.ca

LORETO BRAVO
Universidad del Desarrollo, Facultad de Ingeniería, Santiago, Chile.
bravo@udd.cl

Appendix A Discussion

A.1 On cycles and their assumptions

In this section, unless stated otherwise, we refer to the special semantics introduced in Section 4.

We have assumed that $\mathcal{G}(\mathfrak{P})$ is acyclic. However, the peers, not being aware of being in a cycle in $\mathcal{G}(\mathfrak{P})$, could attempt to do data exchange as described above. In order to detect an infinite loop, for a query $\mathcal{Q}$ posed by a peer P, a unique identifier $id(\mathtt{P}, \mathcal{Q})$ can be created and kept in all the queries that have origin in $\mathcal{Q}$. If an identifier comes back to a peer, it will realize that it is in a cycle and act accordingly.

The assumption of acyclicity of the accessibility graph is quite cautious, in the sense that it excludes cases where a reasonable semantics could still be given and the logic programs would work correctly. This is because the cycles in $\mathcal{G}(\mathfrak{P})$ are not necessarily relevant.

Example 1
Consider $\mathcal{S}(\mathtt{P1}) = \{R^1(\cdot), S^1(\cdot)\}$, $\mathcal{S}(\mathtt{P2}) = \{R^2(\cdot),\ S^2(\cdot)\}$, $\Sigma(\mathtt{P1},\ \mathtt{P2}) = \{\forall x(R^2(x) \rightarrow R^1(x))\}$, $\Sigma(\mathtt{P2}, \mathtt{P1}) = \{\forall x(S^1(x) \rightarrow S^2(x))\}$, $Trust = \{(\mathtt{P1},\ less,\ \mathtt{P2}),\ (\mathtt{P2},\ less,\ \mathtt{P1})\}$. If a query is posed to P1, it will request from P2 the PCAs to query $R^2(x)$, but not those to query $S^2(x)$. Peer P2 can realize it does not need data from P1 and will simply return $D(\mathtt{P2})\lfloor\{R^2\}$ to P1, who will run its solution program and

answer the original query. Even though there is a cycle in $\mathcal{G}(\mathfrak{P})$, there is no infinite loop in the query answering process. □

As we mentioned before, if there are ref-cycles in $\Sigma(\mathtt{P})$, the stable models of the solution program for $\mathtt{P}$ may correspond to a strict superset of the solutions. This is shown in the next example. In such a case, post-processing that deletes models corresponding to non-minimal "solutions" is necessary.

Example 2
Consider $D(\mathtt{P1}) = \{R^1(a,b)\}$, $D(\mathtt{P2}) = \{R^2(a,c)\}$, $\Sigma(\mathtt{P1},\mathtt{P2}) = \{\forall x\forall z(R^1(x,z) \rightarrow \exists y\, R^2(x,y)), \forall x\forall z(R^2(x,z) \rightarrow \exists y\, R^1(x,y)\}$, which is ref-cyclic; $\Sigma(\mathtt{P2}) = \Sigma(\mathtt{P1},\mathtt{P1}) = \emptyset$; and $(\mathtt{P1}, \mathit{same}, \mathtt{P2}) \in \mathit{Trust}$. Here, $\mathtt{P1}$ has only one solution, namely $\{R^1(a,b)\}$.

However, $\Pi(\mathtt{P1})$ has two models. The most relevant part of the program consists of the facts $R^1(a,b)$, $R^2(a,c)$, and the following rules:

$$
\begin{aligned}
R^1\_(x,y,\mathbf{f}) \vee R^2\_(x,null,\mathbf{t}) &\leftarrow R^1\_(x,y,\mathbf{t}^\star), \mathit{not}\ \mathit{aux}_1(x), x \neq null\\
\mathit{aux}_1(x) &\leftarrow R^2(x,null), \mathit{not}\, R^2\_(x,null,\mathbf{f})\\
\mathit{aux}_1(x) &\leftarrow R^2(x,y,\mathbf{t}^\star), \mathit{not}\, R^2\_(x,y,\mathbf{f}), x \neq null, y \neq null\\
R^2\_(x,y,\mathbf{f}) \vee R^1\_(x,null,\mathbf{t}) &\leftarrow R^2\_(x,y,\mathbf{t}^\star), \mathit{not}\ \mathit{aux}_2(x), x \neq null\\
\mathit{aux}_2(x) &\leftarrow R^1(x,null), \mathit{not}\, R^1\_(x,null,\mathbf{f})\\
\mathit{aux}_2(x) &\leftarrow R^1(x,y,\mathbf{t}^\star), \mathit{not}\, R^1\_(x,y,\mathbf{f}), x \neq null, y \neq null
\end{aligned}
$$

The two models correspond to the neighborhood "solutions" $\{R^1(a,b), R^2(a,c)\}$ and $\emptyset$, producing in their turn, the instances $\{R^1(a,b)\}$ and $\emptyset$, resp., for $\mathtt{P1}$. Only the former is a solution instance. The second model is the result of cycles through weak negation (*not*). The cycle creates the self justification of facts as follows: (i) If we choose $R^2\_(a,c,\mathbf{f})$ to be true, then by the second and third rules above, $\mathit{aux}_1(a)$ is false. (ii) Then, the first rule can be satisfied, by making $R^1\_(a,b,\mathbf{f})$ true. (iii) By the fifth and sixth rules, $\mathit{aux}_2(a)$ is false. (iv) This justifies making $R^2\_(a,b,\mathbf{f})$ true, thus, closing the cycle. Notice, that in the whole justification the changes where not determined by inconsistencies. □

There are also cases with an acyclic $\mathcal{G}(\mathfrak{P})$, but with ref-cycles in the DECs, where the logic programming counterpart of the semantics is correct due to the role of the trust relationships.

Example 3
(example 2 continued) If we replace $(\mathtt{P1}, \mathit{same}, \mathtt{P2}) \in \mathit{Trust}$ by $(\mathtt{P1}, \mathit{less}, \mathtt{P2}) \in \mathit{Trust}$, the relevant part of $\Pi(\mathtt{P1})$ now is: $R^1(a,b)$, $R^2(a,c)$, plus

$$
\begin{aligned}
R^1\_(x,y,\mathbf{f}) &\leftarrow R^1\_(x,y,\mathbf{t}^\star), \mathit{not}\ \mathit{aux}_1(x), x \neq null\,\cdot\\
\mathit{aux}_1(x) &\leftarrow R^2(x,null), \mathit{not}\, R^2\_(x,null,\mathbf{f})\,\cdot\\
\mathit{aux}_1(x) &\leftarrow R^2(x,y,\mathbf{t}^\star), \mathit{not}\, R^2\_(x,y,\mathbf{f}), x \neq null, y \neq null\,\cdot\\
R^1\_(x,null,\mathbf{t}) &\leftarrow R^2\_(x,y,\mathbf{t}^\star), \mathit{not}\ \mathit{aux}_2(x), x \neq null\,\cdot\\
\mathit{aux}_2(x) &\leftarrow R^1(x,null), \mathit{not}\, R^1\_(x,null,\mathbf{f})\,\cdot\\
\mathit{aux}_2(x) &\leftarrow R^1(x,y,\mathbf{t}^\star), \mathit{not}\, R^1\_(x,y,\mathbf{f}), x \neq null, y \neq null\cdot
\end{aligned}
$$

Since P1 trusts more peer P2 than itself, it will modify only its own data. This program computes exactly the solutions for peer P1, i.e. $\{R^1(a, b)\}$, even though the DECs exhibit ref-cycles. □

It becomes clear that it is possible to find more relaxed conditions, both on the accessibility graph and ref-cycles, under which a sensible semantics for solutions and semantically corresponding logic programs can be given. Also, for cyclic accessibility graphs, *super peers* (Yang and Garcia-Molina 2003) could be used, to detect cycles and prune certain DECs, making the graph acyclic if necessary; and then our semantics could be applied.

A.2 Query sensitive query answering

Our definition of the solution semantics and the peer consistent answers might suggest that, in order to answer a particular query $\mathcal{Q}$, a peer P has to import the full intersection of the solutions of each of its neighbors, which in their turn have to do the same, etc. If we do this, most likely most of the data imported by P will be irrelevant for the query at hand, and is not needed.

It is possible to design query answering methodologies that are more sensitive to the query at hand, in the sense that only the relevant data is transitively imported into P before answering $\mathcal{Q}$. A full treatment of this subject is beyond the scope of this paper. However, we can give some indications as to how to proceed.

In (Caniupan and Bertossi 2010), the *dependency graph* of database predicates with respect to a set of ICs was introduced and used to capture the notion of possibly transitive relevance of a predicate for another, which is useful in consistent query answering. Here we can use similar graphs for each peer in relationship with its neighbors through a set of DECs, also taking into account the local ICs. These graphs would give us a better upper bound on what to import from other peers (as opposed to bringing the full intersection of solutions).

More precisely, if a query $\mathcal{Q}$ to P contains $\mathcal{S}(\mathtt{P})$-predicates $P_1, \ldots, P_n$, with relevant $\mathcal{S}(\mathtt{Q})$-predicates $Q_1^1, \ldots, Q_{m_1}^1, \ldots, Q_1^n, \ldots, Q_{m_n}^n$, resp., at a neighbor Q, then P will request from Q the PCAs to each of the constant-free, atomic queries $Q_j^i(\bar{x})$. The corresponding sets of answers will form the (most likely smaller) instance provided by Q to P, who will prune its repair program by keeping only the relevant rules, i.e. those that are related to $\mathcal{Q}$, the P_i and the Q_j^i. This idea can be illustrated by means of an example.

Example 4
Consider a schema $\mathfrak{P} = \langle \mathcal{P}, \mathfrak{S}, \Sigma, Trust \rangle$ with $\mathcal{P} = \{\mathtt{P1}, \mathtt{P2}, \mathtt{P3}\}$, $\mathfrak{S} = \{\mathcal{S}(\mathtt{P1}), \mathcal{S}(\mathtt{P2}), \mathcal{S}(\mathtt{P3})\}$, $\mathcal{S}(\mathtt{P1}) = \{R^1(\cdot,\cdot),\ S^1(\cdot,\cdot),\ T^1(\cdot,\cdot)\}$, $\mathcal{S}(\mathtt{P2}) = \{R^2(\cdot,\cdot),\ S^2(\cdot,\cdot), T^2(\cdot,\cdot)\}$, $\mathcal{S}(\mathtt{P3}) = \{R^3(\cdot,\cdot), S^3(\cdot,\cdot)\}$, $Trust = \{(\mathtt{P1}, less, \mathtt{P2}),\ (\mathtt{P2}, same, \mathtt{P3})\}$. Let $\mathfrak{D}$ be an arbitrary instance for the PDES.

The sets of DECs are: $\Sigma(\mathtt{P1},\ \mathtt{P1}) = \{\forall x \forall y(R^1(x, y) \wedge S^1(x, y) \rightarrow \mathbf{false})\}$, $\Sigma(\mathtt{P1}, \mathtt{P2}) = \{\forall x \forall y(R^2(x, y) \rightarrow R^1(x, y)), \forall x \forall y(S^2(x, y) \rightarrow S^1(x, y))\}$, and $\Sigma(\mathtt{P2},\ \mathtt{P3}) = \{\forall x \forall y(R^2(x, y) \wedge R^3(x, y) \rightarrow \mathbf{false})\}$.

If P1 is posed the query $\mathcal{Q}_1(x) :\ \exists y R^1(x, y)$, then the relevant predicates in

$\mathcal{S}(\mathtt{P1})$ are R^1, S^1 (due to the DEC in $\Sigma(\mathtt{P1})$). Then, through the DECs also, it follows that the predicates that are relevant to P1 are R^2, S^2 at P2. So, P1 poses to P2 the queries $R^2(x,y), S^2(x,y)$. The only relevant predicate at P3 is S^3. So, P2 will pose to P3 the query $S^3(x,y)$.

In this case, P3 will return $D(\mathtt{P3}) \downharpoonright \{R^3\}$ to P2, which, due to the UDEC in $\Sigma(\mathtt{P2},\mathtt{P3})$, will subtract it from $D(\mathtt{P2}) \downharpoonright \{R^2\}$, because $(\bigcap Sol(\mathtt{P2},\mathfrak{D})) \downharpoonright \{R^2\} = (D(\mathtt{P2}) \downharpoonright \{R^2\} \setminus D(\mathtt{P3}) \downharpoonright \{R^3\})$. Peer P2 will send this difference to P1 as it is the answer to query $R^2(x,y)$. Peer P2 will also return to P1 the entire $D(\mathtt{P2}) \downharpoonright \{S^2\}$ as its answer to the query $S^2(x,y)$.

Finally, P1 will answer the original query with a solution program containing as facts the tuples in $D(\mathtt{P1}) \downharpoonright \{R^1\}, D(\mathtt{P1}) \downharpoonright \{S^1\}, D(\mathtt{P2}) \downharpoonright \{S^2\}, ((D(\mathtt{P2}) \downharpoonright \{R^2\}) \setminus (D(\mathtt{P3}) \downharpoonright \{R^3\}))$. The last set, as an extension for R^2 in the program. □

The methodology sketched in this example will be certainly more efficient than computing and shipping the full intersection of a peer's solutions. It is natural to expect that additional optimizations can be developed.

A particularly appealing, but provably less general, approach to peer-consistent query answering is *first-order query rewriting*, which we illustrate by means of an example.

Example 5
Consider an instance $\mathfrak{D} = \{D(\mathtt{P1}), D(\mathtt{P2}), D(\mathtt{P3})\}$ for the schema $\mathfrak{P} = \langle \mathcal{P}, \mathfrak{S}, \Sigma, Trust \rangle$ with $\mathcal{P} = \{\mathtt{P1}, \mathtt{P2}, \mathtt{P3}\}$, $\mathfrak{S} = \{\mathcal{S}(\mathtt{P1}), \mathcal{S}(\mathtt{P2}), \mathcal{S}(\mathtt{P3})\}$, $\mathcal{S}(\mathtt{Pi}) = \{R^i(\cdot,\cdot)\}$, $Trust = \{(\mathtt{P1}, less, \mathtt{P2}), (\mathtt{P1}, same, \mathtt{P3})\}$, $D(\mathtt{P1}) = \{R^1(a,b),\ R^1(s,\ t)\}$, $D(\mathtt{P2}) = \{R^2(c,d),\ R^2(a,e)\}$, $D(\mathtt{P3}) = \{R^3(a,f),\ R^3(s,u)\}$ and the DECs:

$$\begin{aligned} \Sigma(\mathtt{P1},\mathtt{P2}) &= \{\forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))\}; \\ \Sigma(\mathtt{P1},\mathtt{P3}) &= \{\forall x \forall y \forall z (R^1(x,y) \wedge R^3(x,z) \rightarrow y = z)\}. \end{aligned}$$

We are interested in P1's solutions. Since P2, P3 are sink peers in the graph $\mathcal{G}(\mathfrak{P})$, we have the extended instance $D = \{R^1(a,b), R^1(s,t), R^2(c,d),\ R^2(a,e),\ R^3(a,f),\ R^3(s,u)\}$, from which we have to obtain the solutions for P1.

The solutions for P1 are obtained by first repairing D with respect to $\Sigma(\mathtt{P1},\mathtt{P2})$, obtaining $D_1 = \{R^1(a,b), R^1(s,t),\ R^1(c,d), R^1(a,e),\ R^2(c,d), R^2(a,e),\ R^3(a,f),\ R^3(s,u)\}$. We have only one repair at this stage, which now has to be repaired in its turn with respect to $\Sigma(\mathtt{P1},\mathtt{P3})$ (but keeping the relationship between P1 and P2 satisfied). There are two sets of tuples violating $\Sigma(\mathtt{P1},\mathtt{P3})$ in D_1: $\{R^1(s,t), R^3(s,u)\}$ and $\{R^1(a,b), R^1(a,e),\ R^3(a,f)\}$. The first violation can be repaired by deleting any, but only one, of the two tuples. The second one, by deleting tuple $R^3(a,f)$ only (otherwise we would violate the relationship between P1 and P2).

As a consequence, we obtain two neighborhood solutions: $D' = \{R^1(a,b), R^1(s,t),\ R^1(c,d),\ R^1(a,e), R^2(c,d), R^2(a,e)\}$, and $D'' = \{R^1(a,b), R^1(c,d), R^1(a,e),\ R^2(c,d), R^2(a,\ e), R^3(s,u)\}$. The solutions for P1 are: $D(\mathtt{P1})' = \{R^1(a,b), R^1(s,t),\ R^1(c,d),\ R^1(a,e)\}$ and $D(\mathtt{P1})'' = \{R^1(a,b),\ R^1(c,d),\ R^1(a,e)\}$.

If the query $\mathcal{Q} : R^1(x,y)$ is posed to P1, the PCAs are $\langle a,b\rangle, \langle c,d\rangle, \langle a,e\rangle$, because those are R^1-tuples found in in the intersection of P1's solutions.

Now, let us try an alternative method for peer consistently answering the same

query. We first rewrite the query using the DEC in $\Sigma(\mathtt{P1}, \mathtt{P2})$, obtaining $\mathcal{Q}'$: $R^1(x, y) \vee R^2(x, y)$, with the effect of bringing P2's data into P1. Next, considering $\Sigma(\mathtt{P1}, \mathtt{P3})$, this query is rewritten as

$$\mathcal{Q}'' : [R^1(x, y) \wedge \forall z_1((R^3(x, z_1) \wedge \neg\exists z_2 R^2(x, z_2)) \rightarrow z_1 = y)] \vee R^2(x, y) \cdot \quad \text{(A1)}$$

To answer this query, P1 first issues a query to P2 to retrieve the tuples in R^2, since they will be essentially in R^1 in all the solutions, due to $\Sigma(\mathtt{P1}, \mathtt{P2})$. Next, a query is issued to P3 to leave aside from the answers those tuples of R^1 that have the same first but not the same second argument in R^3. This filtering is performed as long as there is no tuple in R^2 that "protects" the tuple in R^1. For example, the tuple $R^1(a, b)$ is protected by $R^2(a, e)$ because, as $R^1(a, e)$ belongs to all the solutions, the only way to repair a violation with respect to $\Sigma(\mathtt{P1}, \mathtt{P3})$ is by deleting the tuple from R^3. In this case, the R^1-tuple will be in the answer.

We can see that answering query (A1) amounts to issuing from P1 queries to P2, P3 about the contents of their relations, R^2 and R^3, resp., which are answered by the latter by classical query evaluation over their local instances. After those data have been gathered by P1, it proceeds to evaluate (A1), which contains an implicit repair process.

Now, the answers to (A1) are $\langle a, b\rangle, \langle c, d\rangle, \langle a, e\rangle$, precisely the PCAs we obtained above, considering all the explicit solutions for P1. □

The FO rewriting methodology we just illustrated is bound to have limited applicability. If this was a general mechanism, PCAs to conjunctive queries could be obtained in polynomial time in data, i.e. in the size of the union of the instances of a peer and those of its neighbors (or the local intersections of their solutions). However, Corollary 5.2 tells us that the complexity is higher than this (if $P \neq NP$).

A.3 A semantics based on arbitrary data elements

The purpose of this section is twofold. First, we will present an alternative special semantics that fits into the general semantic framework of Section 3. Second, we will show that this general semantics (and also the one in Section 4) can handle data mappings that are more complex that those usually considered in the related work on peer data exchange. All this will be done on the basis of an extended example.

Consider a PDES $\mathfrak{P} = \langle \mathcal{P}, \mathfrak{S}, \Sigma, Trust\rangle$ with $\mathcal{P} = \{\mathtt{P1}, \mathtt{P2}\}$, $\mathfrak{S} = \{\mathcal{S}(\mathtt{P1}), \mathcal{S}(\mathtt{P2})\}$, $\mathcal{S}(\mathtt{P1}) = \{R^1(\cdot, \cdot),\ T^1(\cdot, \cdot)\}$, $\mathcal{S}(\mathtt{P2}) = \{T^2(\cdot, \cdot),\ S^2(\cdot, \cdot)\}$, $Trust = \{(\mathtt{P1}, less, \mathtt{P2})\}$, and $\Sigma(\mathtt{P1}, \mathtt{P2})$ consists of the following DEC:

$$\forall x \forall y \forall z(R^1(x, y) \wedge T^2(z, y) \rightarrow \exists w(T^1(x, w) \wedge S^2(z, w))) \cdot \quad \text{(A2)}$$

This DEC, which falls within our general syntactic class of DECs, mixes tables of the two peers on each side of the implication. This kind of mapping is more general than those typically considered in virtual data integration (Lenzerini 2002) or data exchange (Kolaitis 2005).[25]

[25] Cf. (Bertossi and Bravo 2004b) for some connections between PDESs and virtual data integration under the *local-as-view* approach. Also (De Giacomo et al. 2007), for relationships between PDESs, virtual data integration, and data exchange.

If (A2) is not satisfied by the data in P1 and P2, which happens when the join in the antecedent is satisfied, but not the one in the consequent, then solutions for P1 have to be found, keeping P2's data fixed in the process, due to the trust relationship. Now, in this section we will depart from the solution semantics introduced in Section 4, by restoring consistency with respect to (A2) through the introduction of arbitrary elements of the data domain $\mathcal{U}$.[26] Those elements become witnesses for the existentially quantified variables in the DECs. That is, these values come from the data domain, and are not replaced by *null* or by labeled nulls as in data exchange (Kolaitis 2005). Since we have alternative choices for them, we may obtain different solutions for a peer. However, by definition of solution, they have to stay as close as possible to the original instance. In this case, the general comparison relation $\preceq_D$ between neighborhood instances of Section 3 is given by: $D_1 \preceq_D^{\Delta} D_2$ iff $\Delta(D, D_1) \subseteq \Delta(D, D_2)$.

We will specify the solutions for this example directly in (or using) logic programs. We will also show the most relevant part of the program $\Pi^-(\texttt{P1})$. Since we have to restore consistency with respect to (A2), the main rules are (A3)-(A6) below.

$$R^1\_(x, y, \mathbf{f}) \leftarrow R^1\_(x, y, \mathbf{t}^\star), T^2(z, y), \mathit{not}\ \mathit{aux}_1(x, z), \mathit{not}\ \mathit{aux}_2(z) \cdot \quad \text{(A3)}$$

$$\mathit{aux}_1(x, z) \leftarrow T^1\_(x, w, \mathbf{t}^\star), S^2(z, w) \cdot \quad \text{(A4)}$$

$$\mathit{aux}_2(z) \leftarrow S^2(z, w) \cdot \quad \text{(A5)}$$

That is, $R^1(x, y)$ is deleted if it participates in a violation of (A2) (what is captured by the first three literals in the body of (A3) plus rule (A4)), and there is no way to restore consistency by inserting a tuple into T^1, because there is no possible matching tuple in S^2 for the possibly new tuple in T^1 (what is captured by the last literal in the body of (A3) plus rule (A5)). In case there is such a tuple in S^2, we can either delete a tuple from R^1 or insert a tuple into T^1:

$$R^1\_(x, y, \mathbf{f}) \vee T^1\_(x, w, \mathbf{t}) \leftarrow R^1\_(x, y, \mathbf{t}^\star),\ T^2(z, y),\ \mathit{not}\ \mathit{aux}_1(x, z),\ S^2(z, w),\ \mathit{choice}((x, z), w) \cdot \quad \text{(A6)}$$

That is, in case of a violation of (A2), when there is tuple of the form $S^2(a, t)$ in S^2 for the combination of values $\langle d, a\rangle$, then the *choice operator* (Giannotti et al. 1991) non-deterministically chooses a unique value for t, so that the tuple $T^1(d, t)$ is inserted into T^1 as an alternative to deleting $R^1(d, m)$ from R^1. The *choice* predicate can be eliminated and replaced by another predicate that can be specified by means of extra but standard rules (Giannotti et al. 1991).

If, instead, we had $Trust = \{(\texttt{P1}, \mathit{same}, \texttt{P2})\}$, P2's relations would also be flexible when searching for solution instances. In this case, the program becomes more involved in terms of presentation (but not conceptually), in the sense that more relations can be updated, and corresponding repair rules have to be added.

Notice that in this example, the values that are chosen as witnesses for the existential quantifier in the DEC are taken from the active domain of the database,

[26] This $\mathcal{U}$ could be a finite superset of the union of the active domains involved or infinite. The latter case is also covered by our semantics. The logic programming semantics is also perfectly defined in this case.

namely from the set of values for the second attribute of relation S^2. In other cases, for example with a DEC of the form $\forall x \forall y(T^2(x, y) \rightarrow \exists z R^1(x, z))$, we have to consider arbitrary values from an underlying domain *dom* when inserting tuples into R^1. In this case, *dom* requires a specification as a finite predicate in the program.

Some of the ideas presented above (such as the insertion of elements from the active domain and the use of the choice operator) have been fully developed and applied by the authors (Bertossi and Bravo 2004a; Bravo and Bertossi 2003; Bravo and Bertossi 2005) to inconsistency management in virtual data integration systems under the *local-as-view approach* (Lenzerini 2002).

A.4 Data transport and semantics

The data distributed across different peers has to be appropriately gathered to build solution instances for a peer, and different semantics may emerge as candidates, depending on the granularity of the data sent between peers. In the context of the general semantic framework introduced in Section 3, we developed a particular semantics in Section 4, according to which a peer `Q` passes back to a neighbor `P`, who is building its solutions, this is (part of) its certain data. This is the one that holds in all of `Q`'s solutions.

In (Bravo 2007, chapter 7) also two other alternative semantics are fully developed and compared, in particular establishing some conditions under which they coincide or differ. Those other semantics assume that more detailed information, such as mappings and trust relationships, can be sent between peers. We briefly describe them.

1. Send all. The first one assumes that data, DECs and trust relations can be sent between peers. So, we can think that we have a possibly large database instance that has to be virtually repaired in order to satisfy all the relevant DECs (obtained from the accessibility graph) and at the same time accommodating the trust relationships. In this case, the DECs are treated as traditional ICs on the integrated instance. This semantics is similar to the repair semantics for consistent query answering. Preferences imposed on repairs can be used to capture the trust relationships. In this case, it is not necessary to require the acyclicity of the accessibility graph.

2. Send solutions. The second one assumes that only solutions can be send between neighboring peers. In this case, a peer `P` requests the solutions of the neighbors in order to calculate its own solutions. Here, the database consists of the data at `P` plus all the solutions of `P`'s neighbors; and the constraints are the DECs between `P` and its neighbors. As in Section 3, this is a recursive definition, and assumes an acyclic accessibility graph.

We think that the semantics we developed in Section 3, which could be called *"send cores"*, is more natural (a peer passes over what it is certain about), and also simplifies reasoning at the local level, i.e. at each peer's site, because at most one instance peer neighbor has to be considered.

Now, under our official semantics, if we want to use local solution programs, each neighbor of a peer P has to run its program, and then send the (relevant part of the) intersection of the stable models to P, who runs its local solution program. This means that different programs have to be fed externally.

At least under the *"send solutions"* and *"send cores"* semantics (which assumes an acyclic peer graph), we could imagine having a single program that does all this output/input concatenation, *internally*. Actually, it is possible to build a *single program for a peer*, say $\Pi^{man}(\mathtt{P})$, that acts as a combination of solution programs as given in Section 6. For each peer Q that is accessible from P, the program locally runs a solution program, produces Q's solutions (its stable models), or the intersection thereof; and, without leaving $\Pi^{man}(\mathtt{P})$, passes them to its preceding neighbors Q', who uses them to locally compute its own solutions by means of its own, local solution program, etc., until reaching P. Such a program $\Pi^{man}(\mathtt{P})$ can be a *manifold program* (Faber and Woltran 2011).

Actually, within a manifold program (MP), a program can pass certain or possible query answers as an input to another program. Conceptually, MPs offer a nice logical solution, by means of a single program, to this form of program combination that, otherwise, would require external intervention (the efficient implementation of MPs is still an open problem).

A.5 On trust

We introduced trust relationships in the process of peer data exchange already in (Bertossi and Bravo 2004b); and here we have further developed this idea, in a general semantic framework. However, the notion of trust we have in this work is still rather simple. Actually, it can be represented as an annotated binary relation between peers. It would be interesting to impose a more sophisticated and rich model of trust on top of the DECs-bases network of peers. Our concern is not about computing or updating trust in a P2P overlay network (Xiong and Liu 2004; Jøsang et al. 2006), but about logical specifications of trust. We envision a logic-based model of trust that can be integrated with/into the DECs. Such a model could express some higher level properties, e.g. symmetry or transitivity of trust relationships. A logic-based representation of trust could allow us to infer non-explicit trust relationships whenever necessary.

Trust modeling is an active and important area of research nowadays, most notably in the context of the semantic web. See (Sabater and Sierra 2005; Artz and Gil 2007) for surveys. The integration of trust models, including related notions, like reputation, provenance, etc., into peer data exchange is still an open area of research. This is specially the case for logic-based models of trust (Herzig et al. 2008). See (Hien Nguyen et al. 2008) and references therein for probabilistic approaches.

In Definition 6.1, the trust relationships between peers were implicitly and rigidly captured in the specifications of solutions by means of the disjunctive heads of the repair rules. Although this is a simpler way of presenting things, it has some drawbacks: (a) The approach is not modular in the sense that trust is built-in into the rules; (b) Changes in trust relationships requires changing heads of repair rules;

and (c) In case no solution exists due to the rigid and conflicting trust relationships, no alternative, but possibly less desirable solution can be obtained as a stable model of the solution program.

One way of addressing these issues is through the use of *preference programs*, which are answer set programs that express different forms of preference, which essentially amounts to preferring and keeping only certain stable models of the program. For example, in a disjunctive rule of the form $A \vee B \leftarrow Body$, one could prefer to make A true instead of B. If this is possible, only that stable model would be chosen. However, if that is not possible (due to the other rules and facts in the program), making B true is still good enough. More complex preferences could also be captured. Preferences can be explicitly and declaratively expressed, and the resulting programs can be compiled into usual answer set programs with their usual stable model semantics (cf. (Brewka 2004) and references therein).

Here we briefly outline how *weak program constraints* (Buccafurri et al. 2000; Leone et al. 2006) declaratively capture the kind of preferences that address our needs. (Cf. (Brewka 2004) for connections between preferences in logic programs and weak constraints.).

Example 6
Consider Example 6.2, where $(\mathtt{P1}, less, \mathtt{P2}) \in Trust$ is captured by the non-disjunctive repair rule

$$R^1\_(x, y, \mathbf{t}) \leftarrow R^2\_(x, y, \mathbf{t}^\star), R^1\_(x, y, \mathbf{f}^\star), x \neq null, y \neq null\cdot$$

The same effect, and more, could be obtained by uniformly using disjunctive rules followed by appropriate weak constraints. In this case,

$$\begin{aligned} R^1\_(x, y, \mathbf{t}) \vee R^2\_(x, y, \mathbf{f}) \quad &\leftarrow \quad R^2\_(x, y, \mathbf{t}^\star), R^1\_(x, y, \mathbf{f}^\star), x \neq null, y \neq null\cdot \\ &\Leftarrow \quad R^2\_(x, y, \mathbf{f})\cdot \end{aligned} \tag{A7}$$

Here, the weak constraint (A7) expresses a preference for the stable models of the program that minimize the number of violations of the condition expressed its body, in this case, that, when restoring the satisfaction of the DEC $\forall x \forall y(R^2(x, y) \rightarrow R^1(x, y))$, the tuple $R^2(x, y)$ is not deleted. These weak constraints are used by a peer P to ensure that, if possible, the tuples in the peers that it trusts more than itself are not modified. □

If the original solution program has solutions, then the new program would have the same solutions. However, the latter could have solutions when the former does not. This would make the semantics of the system more flexible with respect to unsatisfiable trust requirements. It is also clear that the weak constraints could be easily derived from the trust relationships and the DECs. The solution program with weak constraints can be run in the *DLV* system (Leone et al. 2006) to obtain the solutions and peer consistent answers of a peer.

Notice that the new repair programs, except for the weak program constraints, are now of the same kind as those for specifying repairs of single databases with respect to local ICs (Bravo and Bertossi 2006). Actually, if in the new program the

weak program constraints are replaced by (hard) program constraints, e.g. (A7) by $\leftarrow R^2\_(x, y, \mathbf{f})$, the solutions coincide with those of the programs in Definition 6.1.

We should mention that in (Arenas et al. 2003), weak constraints were used, as a part of a repair program, to specify the preference for *cardinality repairs*, i.e. repairs that minimize the *number* of tuples that are inserted or deleted to restore consistency, as opposed to minimality (with respect to subset-inclusion) of sets of inserted/deleted tuples.

Appendix B Proofs of Results

Proof of *Proposition 3.1*:

Let D_0 be an empty instance for the schema $\mathcal{S}(\mathcal{N}(P))$. By being empty, D_0 satisfies $\bigcup_{\mathtt{Q}\in\mathcal{N}(\mathtt{P})} \Sigma(\mathtt{P}, \mathtt{Q})$ (condition (i) for a neighborhood solution). Also, since all the trust relationships are of the "same" kind, condition (ii) on neighborhood solutions is satisfied by D_0. Then, either D_0 is a neighborhood solution, or there exists a neighborhood solution D'' such that $D'' \preceq_{\bar{D}} D_0$. □

Proof of *Corollary 3.1*:

All we need is notice that the possibly inconsistent sink peers in the accessibility graph always have local repairs under the kind of DECS considered (ICs in that case). A solution for a peer P can then be obtained by recursively propagating back neighborhood solutions (that always exist by Proposition 3.1) for peers along the paths that contain P. □

Proof of *Proposition 5.1*:

Membership of *coNP* is established by directly appealing to Definition 3.1 of neighborhood solution. In fact, given neighborhood instance J, after having checked (in polynomial time) if $J \subseteq r\text{-}Chase^{null}(\bar{D}, \Sigma^-(\mathtt{P}))$, a non-deterministic algorithm to test that J is *not* a neighborhood solution for P and the neighborhood instance $\bar{D}$ checks if one of the following holds:

1. $J \not\models \Sigma(\mathtt{P})$.
2. $J \restriction \{R\} \neq \bar{D} \restriction \{R\}$, for some $\mathtt{Q} \in \mathcal{N}(\mathtt{P})$ and predicate $R \in \mathcal{S}(\mathtt{Q})$ with $(\mathtt{P}, less, \mathtt{Q}) \in Trust$.
3. There is an instance J' for $\mathcal{S}(\mathcal{N}(\mathtt{P}))$ (the non-deterministic choice) that satisfies conditions (i) and (ii) of Definition 3.1, but $J' <_{\bar{D}}^{\Sigma(\mathtt{P})} J$.

These conditions are the basis for a non-deterministic algorithm: First conditions 1. and 2. can be checked deterministically in polynomial time. If they are passed by J (i.e. the answer is negative), then an instance J' with size polynomially bounded by the size of J is guessed. Next, conditions 1.-2. are checked for J', and 3., for the pair (J, J'). The three tests can be performed in polynomial time in $|J| + |\bar{D}|$. If the answer to any of the tests is *yes*, J is not a neighborhood solution.

Hardness can be proved by reduction of satisfiability of propositional formulas in CNF, which is *coNP*-complete. The reduction is a modification of that used in

Theorem 4.4 of (Chomicki and Marcinkowski 2005) to prove that repairs obtained through deletions are *coNP*-complete. In our case we have to deal with trust relationships and the possible insertion of tuples with *null*. Actually, in our proof the former will be used to exclude the latter.

We now show that the satisfiability of a propositional formula $\varphi: \ \varphi_1 \wedge \varphi_2 \wedge \ldots \varphi_m$ in CNF (i.e. the φ_i are clauses) can be reduced to checking if a particular neighborhood instance is a neighborhood solution for a given peer.

Consider the fixed PDES schema (it does not depend on φ): $\mathfrak{P} = \langle \mathcal{P}, \mathfrak{S}, \Sigma, Trust \rangle$, with $\mathcal{P} = \{\mathtt{P1}, \mathtt{P2}\}$, $\mathfrak{S} = \{S(\mathtt{P1}), S(\mathtt{P2})\}$, $S(\mathtt{P1}) = \{R^1(\cdot,\cdot,\cdot,\cdot)\}$, $S(\mathtt{P2}) = \{R^2(\cdot,\cdot,\cdot,\cdot)\}$, $Trust = \{(\mathtt{P1}, \mathit{same}, \mathtt{P1}), (\mathtt{P1}, \mathit{less}, \mathtt{P2})\}$, and $\Sigma = \{\Sigma(\mathtt{P1}, \mathtt{P1}), \Sigma(\mathtt{P1}, \mathtt{P2})\}$, with:

$$\begin{aligned}\Sigma(\mathtt{P1}, \mathtt{P1}) = \{&\forall x_1 y_1 y_2 z_1 z_2 w_1 w_2 (R^1(x_1, y_1, z_1, w_1) \wedge R^1(x_1, y_2, z_2, w_2) \rightarrow y_1 = y_2),\\ &\forall x_1 y_1 z_1 w_1 (R^1(x_1, y_1, z_1, w_1) \rightarrow \exists x_2 y_2 z_2 R^1(x_2, y_2, z_2, z_1))\}.\\ \Sigma(\mathtt{P1}, \mathtt{P2}) = \{&\forall xyzw (R^1(x, y, z, w) \rightarrow R^2(x, y, z, w))\}.\end{aligned}$$

Now, consider a propositional formula φ as above, on which the instances for the peer system will depend. Let $\mathfrak{D}_\varphi = \{D(\mathtt{P1}), D(\mathtt{P2})\}$ be the instance for $\mathfrak{P}$, with:

$$\begin{aligned} D(\mathtt{P1}) &= \{R^1(p_j, 0, \varphi_i, \varphi_{i+1}) \mid p_j \text{ occurs negatively in } \varphi_i\} \cup \\ &\quad \{R^1(p_j, 1, \varphi_i, \varphi_{i+1}) \mid p_j \text{ occurs positively in } \varphi_i\}\cdot \\ D(\mathtt{P2}) &= \{R^2(a, b, c, d) \mid R^1(a, b, c, d) \in D(\mathtt{P1})\}\cdot \end{aligned}$$

(The addition $i{+}1$ is meant to be modulo the number m of clauses in ϕ.) Notice that tables R^1 and R^2 for peers P1 and P2, respectively, have the same rows. Intuitively, the UDEC in $\Sigma(\mathtt{P1}, \mathtt{P1})$ ensures that, for every proposition in the first attribute of R^1, the truth assignment, if any, is unique; whereas the RDEC in it, ensures that there are assignments that make all clauses in the formula true.

We now show that the neighborhood instance $\bar{D}$, with the empty relation for R^1 plus the original contents of $D(\mathtt{P2})$, is the neighborhood solution for P1 with initial neighborhood instance $D(\mathtt{P1}) \cup D(\mathtt{P2})$ if and only if φ is *not* satisfiable. In this case, $\bar{D}$ would be obtained through the deletion of all tuples from R^1 in $D(\mathtt{P1})$. Notice that due to the trust relations and the DEC in $\Sigma(\mathtt{P1}, \mathtt{P2})$, only tuple deletions from peer P1's instance are admissible updates.

To prove that $\bar{D}$ being a neighborhood solution for P1 implies that φ is not satisfiable, assume by contradiction that φ is satisfiable. Then, there is an assignment σ that makes φ true.

The instance $\bar{D}' := \{R^1(p, 0, \varphi_i, \varphi_{i+1}) \in D(\mathtt{P1}) \mid \sigma(p) = 0\} \cup \{R^1(p, 1, \varphi_i, \varphi_{i+1}) \in D(\mathtt{P1}) \mid \sigma(p) = 1\} \cup D(\mathtt{P2})$ is a subinstance of $D(\mathtt{P1}) \cup D(\mathtt{P2})$, $\bar{D}' \lfloor \mathcal{S}(\mathtt{P1}) \neq \emptyset$, satisfies the DECs, and does not modify the more trusted relations, i.e. $\bar{D}' \lfloor \mathcal{S}(\mathtt{P2}) = D(\mathtt{P2})$. Thus, $\bar{D}$ cannot be a neighborhood solution since $\bar{D}' <^{\Sigma(\mathtt{P1})}_{D(\mathtt{P1}) \cup D(\mathtt{P2})} \bar{D}$.

Now we show that if φ is not satisfiable, then $\bar{D}$ is a neighborhood solution for P1 when starting with neighborhood instance $D(\mathtt{P1}) \cup D(\mathtt{P2})$. Assume by contradiction that $\bar{D}$ is not a neighborhood solution. Since $\bar{D}$ satisfies all the DECs and respects the trust relationships, $\bar{D}$ cannot be a neighborhood solution only if there is a neighborhood instance $\bar{D}'$, such that: $\bar{D}' \models \Sigma(\mathtt{P1})$; $\bar{D}' \lfloor S(\mathtt{P2}) = D(\mathtt{P2})$, and $\bar{D}' <^{\Sigma(\mathtt{P1})}_{D(\mathtt{P1}) \cup D(\mathtt{P2})} \bar{D}$. Since $\bar{D}'$ can be obtained only through tuple deletions, it holds:

$\bar{D} \subsetneqq D' \subseteq D(\mathtt{P1}) \cup D(\mathtt{P2})$. Thus, there is at least one tuple $R^1(\bar{t}) \in (\bar{D}' \cap D(\mathtt{P1}))$. Due to the UDEC in $\Sigma(\mathtt{P1}, \mathtt{P2})$, we conclude that, for every $i \in [1, m]$, there exists a p and v with $R^1(p, v, \varphi_i, \varphi_{i+1}) \in (D' \cap D(\mathtt{P1}))$. Using these tuples we can define the following assignment σ':

$$\sigma'(p) = \begin{cases} 1 & \text{if } R^1(p, 1, \varphi_i, \varphi_{i+1}) \in D' \text{ with } i \in [1, m] \\ 0 & \text{if } R^1(p, 0, \varphi_i, \varphi_{i+1}) \in D' \text{ with } i \in [1, m] \end{cases}$$

The assignment is well defined, because the functional dependency in $\Sigma(\mathtt{P1}, \mathtt{P1})$ ensures that only one value exists for each proposition. By construction, σ' is an assignment that satisfies φ. We have reached a contradiction, which completes the proof. □

Proof of *Proposition 5.2*:

First we prove membership of Π_2^P. An atom $R(\bar{t}) \in \mathit{localCore}(\mathtt{P}, \bar{D})$ if for every $D' \in \mathit{NS}(\mathtt{P}, \bar{D})$, $D' \models R(\bar{t})$. Thus, a non-deterministic algorithm that checks if $R(\bar{t}) \notin \mathit{localCore}(\mathtt{P}, \bar{D})$ guesses an instance J of $\mathcal{S}(\mathcal{N}(\mathtt{P}))$, next checks if it is a neighborhood solution for $\mathtt{P}$ and $\bar{D}$, and finally, if $R(\bar{t}) \notin J$. By Proposition 5.1, the first of these two tests is in *coNP*; and the second one is in polynomial time. Thus, the problem is in Π_2^P.

Hardness holds by a reduction from satisfiability of a quantified propositional formulas (QBF) β of the form $\forall p_1 \cdots \forall p_k \exists q_1 \cdots \exists q_l \psi$, where ψ is a quantifier-free propositional formula in CNF, i.e. of the form $\psi_1 \wedge \ldots \wedge \psi_m$, where the ψ_i are clauses. This problem is Π_2^P-complete (Schaefer and Umans 2008; Papadimitriou 1994). (The reduction is adapted from that for Theorem 4.7 in (Chomicki and Marcinkowski 2005).)

We construct a PDES schema (independent from β) $\mathfrak{P}_0 = \langle \mathcal{P}_0, \mathfrak{S}_0, \Sigma_0, \mathit{Trust}_0 \rangle$, with $\mathcal{P}_0 = \{\mathtt{P1}, \mathtt{P2}\}$, $\mathfrak{S}_0 = \{\mathcal{S}(\mathtt{P1}), \mathcal{S}(\mathtt{P1})\}$, $\mathcal{S}(\mathtt{P1}) = \{R(\cdot,\cdot,\cdot), T(\cdot)\}$, $\mathcal{S}(\mathtt{P2}) = \{\mathit{Clause}(\cdot), \mathit{Var}(\cdot)\}$, $\mathit{Trust}_0 = \{(\mathtt{P1}, \mathit{less}, \mathtt{P2})\}$, and the DECs $\Sigma_0 = \{\Sigma(\mathtt{P1}, \mathtt{P2}), \Sigma(\mathtt{P1}, \mathtt{P1})\}$ with:

$$\begin{aligned} \Sigma(\mathtt{P1}, \mathtt{P2}) &= \{\forall x(\mathit{Clause}(x) \rightarrow \exists yz R(y, z, x)), \\ &\qquad \forall x(\mathit{Var}(x) \rightarrow R(x, 1, a) \vee R(x, 0, a))\}, \\ \Sigma(\mathtt{P1}, \mathtt{P1}) &= \{\forall xy_1y_2z_1z_2(R(x, y_1, z_1) \wedge R(x, y_2, z_2)) \rightarrow y_1 = y_2, \\ &\qquad \forall xyzw(R(x, y, z) \wedge T(w) \rightarrow w \neq \mathbf{sat} \vee \mathit{IsNotNull}(x) \vee \mathit{IsNotNull}(y))\}. \end{aligned}$$

Now, given a QBF β, we construct an instance $\bar{D}_\beta$ for the neighborhood schema $\mathcal{S}(\mathcal{N}(\mathtt{P1}))$ around $\mathtt{P1}$, such that: $T(\mathbf{sat}) \in \mathit{localCore}(\mathtt{P1}, \bar{D}_\beta)$ iff β is true.

Now, for $\beta = \forall p_1 \cdots \forall p_k \exists q_1 \cdots \exists q_l (\psi_1 \wedge \ldots \wedge \psi_m)$, $\bar{D}_\beta := D_\beta(\mathtt{P1}) \cup D_\beta(\mathtt{P2})$, with:

$$\begin{aligned} D_\beta(\mathtt{P1}) &:= \{R(\mathit{var}_j, 1, \psi_i) \mid \mathit{var}_j \text{ occurs positively in } \psi_i\} \cup \\ &\qquad \{R(\mathit{var}_j, 0, \psi_i) \mid \mathit{var}_j \text{ occurs negatively in } \psi_i\} \cup \\ &\qquad \{\mathit{Var}(p_i) \mid p_i \text{ is universally quantified in } \psi\} \cup \{T(\mathbf{sat})\}. \\ D_\beta(\mathtt{P2}) &:= \{\mathit{Clause}(\psi_1), \ldots, \mathit{Clause}(\psi_m)\}. \end{aligned}$$

Intuitively, relation $R(x, y, z)$ is used to provide a truth value y to variable x in conjunct z. This truth value will be unique across ψ due to the integrity constraints

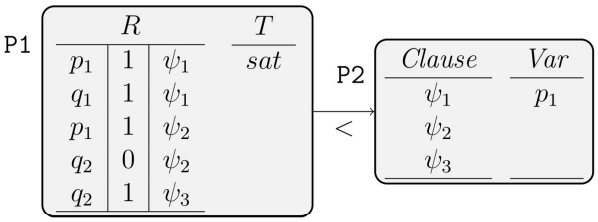

Fig. B 1. Instances for a peer system

on R contained in $\Sigma(\mathtt{P1}, \mathtt{P1})$. The first DEC in $\Sigma(\mathtt{P1}, \mathtt{P2})$ ensures that, for every clause ψ_i, there is, if possible, a literal which is true in it. If it is not possible (the formula is not true), it inserts a tuple of the form $R(null, null, \psi_i)$. The second DEC ensures that all possible assignments for the universally quantifies variables are tested in different solutions. It uses a constant, a, which is different from all ψ_i. The first IC in $\Sigma(\mathtt{P1}, \mathtt{P1})$ enforces that, in each solution, each propositional variable takes a unique value. Finally, the second IC in $\Sigma(\mathtt{P1}, \mathtt{P1})$ ensures that if $R(null, null, \psi_i)$ is true, then predicate $T(\mathbf{sat})$ should not be part of the neighborhood solution. In this way, formula β is true if and only of $T(\mathbf{sat}) \in localCore(\mathtt{P1}, \bar{D}_\beta)$. To conclude the proof, we illustrate the reduction with the following example.

Example 7
Consider the QBF $\forall p_1 \exists q_1 \exists q_2 (\psi_1 \wedge \psi_2 \wedge \psi_3)$, with $\psi_1 \colon\ (p_1 \vee q_1)$, $\psi_2 \colon\ (p_1 \vee \neg q_2)$, and $\psi_3 \colon\ q_2$. Instance $\bar{D}_\beta$ is the union of the instances in Figure B 1.

On this basis, the neighborhood solutions for P1 and $\bar{D}_\beta$ are:
$D_1 = \{R(p_1, 1, a), R(p_1, 1, \psi_1), R(q_1, 1, \psi_1), R(p_1, 1, \psi_2), R(q_2, 1, \psi_3), T(\mathbf{sat})\} \cup D(\mathtt{P2})$,
$D_2 = \{R(p_1, 0, a), R(q_1, 1, \psi_1), R(q_2, 1, \psi_3), R(null, null, \psi_2)\} \cup D(\mathtt{P2})$, and
$D_3 = \{R(p_1, 0, a), R(q_1, 1, \psi_1), R(q_2, 0, \psi_2), R(null, null, \psi_3)\} \cup D(\mathtt{P2})$.

Since $T(\mathbf{sat}) \notin localCore(\mathtt{P1}, \bar{D}_\beta) := (D_1 \cap D_2 \cap D_3) \restriction \mathcal{S}(\mathtt{P1})$, the QBF formula is false.

□

Proof of *Corollary 5.2*:

Membership is established with a test similar to that in the proof of Proposition 5.2. Hardness follows from Proposition 5.2, because it is about a particular kind of conjunctive queries, namely atomic of the form $\mathcal{Q}(\bar{x}) \colon R(\bar{x})$, where R is a predicate for a peer P. The peer consistent answers to this query are exactly the $\bar{c}$s, such that $R(\bar{c})$ is in the local core of P. □

Proof of *Proposition 5.3*:

The existence and uniqueness is straightforward since there are no local restrictions (existence), and there is no non-determinism involved (uniqueness). The unique solution for a peer P can be computed by means of Algorithm ImportSolution shown in Figure B 2. It recursively computes the solutions for all the peers that are accessible from P. The base case occurs when a peer Q has no DECs (line 4). In that case,

1 **Algorithm:** ImportSolution
2 **Input**: An instance $\mathfrak{D}$ for a PDES schema $\mathfrak{P} = \langle \mathcal{P}, \mathfrak{S}, \Sigma, Trust \rangle$ and a peer $\mathtt{P} \in \mathcal{P}$
3 **Output**: The unique solution of $\mathtt{P}$
4 **if** *P has no outgoing edges* **then return** $D(\mathtt{P})$;
5 **else**
6 | **foreach** $\mathtt{Q} \in \mathcal{N}^{\circ}(P)$ **do**
7 | | $Sol(\mathtt{Q}, \mathfrak{D}) \leftarrow$ ImportSolution$(\mathcal{P}, \mathtt{Q})$;
8 | $D' \leftarrow D(\mathtt{P}) \cup \bigcup_{\mathtt{Q} \in \mathcal{N}^{\circ}(\mathtt{P})} Sol(\mathtt{Q}, \mathfrak{D})$;
9 | $NS \leftarrow$ *minimal model of Datalog import program* $\mathfrak{I}(\mathtt{P}, D')$;
10 | **return** $NS \lfloor S(\mathtt{P})$;

Fig. B 2. Computing the solution for a peer in the import case

its unique solution is its own database $D(\mathtt{Q})$. Otherwise (lines 5-10), the algorithm recursively requests the solutions of its neighbors (lines 6-7) and uses them to construct instance D' (line 8). Then, the unique neighborhood solution for the peer consists of the minimal model of $\mathfrak{I}(\mathtt{P}, D')$ (line 9). By restricting $\mathtt{P}$'s neighborhood solution to $\mathtt{P}$'s schema we get $\mathtt{P}$'s solution (line 10). □

Proof of *Proposition 6.1*:

We will prove this result for the case where the central peer trusts its neighbors as much as itself, which is more general in some sense than that where it trusts neighbors' data more, because more alternatives for repairs come up, using the full power of disjunctive programs. Below D is $\mathtt{P}$'s neighborhood instance for which neighborhood solutions are specified by means of the program in Definition 6.1. $\mathcal{C}$ is the set of constraints, i.e. UDECs and RDECs, for the neighborhood.

Given the trust assumptions, the program takes a special form, as follows. For exchange constraints in $\mathcal{C}$ of the forms:

(a) *Universal constraint* (UDEC):

$$\forall \bar{x}(\bigwedge_{i=1}^{m} P_i(\bar{x}_i) \longrightarrow \bigvee_{j=1}^{n} Q_j(\bar{y}_j) \vee \varphi). \tag{B1}$$

(b) *Referential constraint*: (RDEC)

$$\forall \bar{x}\,(P(\bar{x}) \longrightarrow \exists \bar{y}\, Q(\bar{x}', \bar{y}))\,.^{27} \tag{B2}$$

the neighborhood solution program $\Pi(\mathtt{P}, D)$ becomes:

1. $dom(c)$ for every $c \in Adom(D) \setminus \{null\}$.
2. The fact $P(\bar{a})$ for every atom $P(\bar{a}) \in D$.
3. For every UDEC ψ of the form (B1), the rule:
$\bigvee_{i=1}^{n} P_i\text{-}(\bar{x}_i, \mathbf{f}) \vee \bigvee_{j=1}^{m} Q_j\text{-}(\bar{y}_j, \mathbf{t}) \longleftarrow \bigwedge_{i=1}^{n} P_i\text{-}(\bar{x}_i, \mathbf{t}^{\star}), \bigwedge_{j=1}^{m} Q_j\text{-}(\bar{y}_j, \mathbf{f}^{\star}), \bigwedge_{x_l \in RelV(\psi)} dom(x_l), \bar{\varphi}\cdot$
where $RelV(\psi)$ is the set of relevant attributes for ψ, $\bar{x} = \bigcup_{i=1}^{n} x_i$, and $\bar{\varphi}$ is a conjunction of built-ins that is equivalent to the negation of φ.
4. For every RDEC ψ of the form (B2), the rules:[28]

[28] Literal $dom(\bar{x})$ denotes the conjunction of the atoms $dom(x_j)$ for $x_j \in \bar{x}$.

$P_{\_}(\bar{x}, \mathbf{f}) \vee Q_{\_}(\bar{x}', \overline{null}, \mathbf{t}) \leftarrow P_{\_}(\bar{x}, \mathbf{t}^{\star}), \mathit{not}\ \mathit{aux}_{\psi}(\bar{x}'), \mathit{dom}(\bar{x}')\cdot$
And for every $y_i \in \bar{y}$:
$\mathit{aux}_{\psi}(\bar{x}') \leftarrow Q_{\_}(\bar{x}', \bar{y}, \mathbf{t}^{\star}), \mathit{not}\ \ Q_{\_}(\bar{x}', \bar{y}, \mathbf{f}), \mathit{dom}(\bar{x}'), \mathit{dom}(y_i).$
$\mathit{aux}_{\psi}(\bar{x}') \leftarrow Q(\bar{x}', \overline{null}), \mathit{not}\ \ Q_{\_}(\bar{x}', \overline{null}, \mathbf{f}), \mathit{dom}(\bar{x}').$

5. For every predicate $P \in \mathcal{S}(\mathcal{N}(\mathtt{P}))$, the rules:
$P_{\_}(\bar{x}, \mathbf{t}^{\star}) \leftarrow P(\bar{x})\cdot \quad P_{\_}(\bar{x}, \mathbf{t}^{\star}) \leftarrow P_{\_}(\bar{x}, \mathbf{t})\cdot$
$P_{\_}(\bar{x}, \mathbf{f}^{\star}) \leftarrow P(\bar{x}, \mathbf{f})\cdot \quad P_{\_}(\bar{x}, \mathbf{f}^{\star}) \leftarrow \mathit{dom}(\bar{x}), \mathit{not}\ \ P(\bar{x})\cdot$
6. For every predicate $P \in \mathcal{S}(\mathcal{N}(\mathtt{P}))$, the *interpretation rules*:
$P_{\_}(\bar{x}, \mathbf{t}^{\star\star}) \leftarrow P_{\_}(\bar{x}, \mathbf{t})\cdot \qquad P_{\_}(\bar{x}, \mathbf{t}^{\star\star}) \leftarrow P(\bar{x}), \mathit{not}\ \ P_{\_}(\bar{x}, \mathbf{f})\cdot$
7. For every predicate $P \in \mathcal{S}(\mathcal{N}(\mathtt{P}))$, the *coherence constraints*:
$\leftarrow \quad P_{\_}(\bar{x}, \mathbf{t}),\ P_{\_}(\bar{x}, \mathbf{f})\cdot$

The claim is: If $\mathcal{M}$ is a stable model of $\Pi(\mathtt{P}, D)$, then D_M is a neighborhood solution repair of D. Furthermore, the neighborhood solutions obtained in this way are all the neighborhood solutions of D. We recall that for a stable model of $\Pi(\mathtt{P}, D)$,

$$D_M = \{P(\bar{a}) \ \mid \ P \in \mathcal{S}(\mathcal{N}(\mathtt{P})) \text{ and } P_{\_}(\bar{a}, \mathbf{t}^{\star\star}) \in M\}\cdot \tag{B3}$$

The proof follows directly from Propositions 1 and 2 below, which require in their turn some lemmas and intermediate definitions. □

In the following, for a disjunctive program Π and a set of ground atoms M, Π^M is the positive ground program obtained by the Gelfond-Lifschitz transformation (Gelfond and Lifschitz 1991):

$$\Pi^M = \{H \leftarrow B \ \mid \ H \leftarrow B, \mathit{not}\, A_1, \ldots, \mathit{not}\, A_m \in \mathit{ground}(\Pi), \text{ and } A_1, \ldots A_m \notin M\}\cdot$$

Lemma 1
Given an instance D and a RDEC-acyclic set of UDECs and RDECs, if M is a stable model of $\Pi(\mathtt{P}, D)$, then exactly one of the following cases holds:

1. $P(\bar{a})$, $P_{\_}(\bar{a}, \mathbf{t}^{\star})$ and $P_{\_}(\bar{a}, \mathbf{t}^{\star\star})$ belong to M, and no other $P_{\_}(\bar{a}, v)$, for v an annotation, belongs to M.
2. $P(\bar{a})$, $P_{\_}(\bar{a}, \mathbf{t}^{\star})$ and $P_{\_}(\bar{a}, \mathbf{f})$ belong to M, and no other $P_{\_}(\bar{a}, v)$, for v an annotation, belongs to M.
3. $P(\bar{a}) \notin M$, and $P_{\_}(\bar{a}, \mathbf{t})$, $P_{\_}(\bar{a}, \mathbf{t}^{\star})$, $P_{\_}(\bar{a}, \mathbf{t}^{\star\star})$ belong to M, and no other $P_{\_}(\bar{a}, v)$, for v an annotation, belongs to M.
4. $P(\bar{a}) \notin M$, and no $P_{\_}(\bar{a}, v)$, for v an annotation, belongs to M.

Proof**:** For an atom $P(\bar{a})$, there are two possibilities:

(a) $P(\bar{a}) \in M$. Then, from rule 5., $P_{\_}(\bar{a}, \mathbf{t}^{\star}) \in M$. Two cases are possible now: $P_{\_}(\bar{a}, \mathbf{f}) \notin M$ or $P_{\_}(\bar{a}, \mathbf{f}) \in M$. In the first case, since M is minimal, $P_{\_}(\bar{a}, \mathbf{t}) \notin M$ and $P_{\_}(\bar{a}, \mathbf{t}^{\star\star}) \in M$. In the second case, due to rule 7., $P_{\_}(\bar{a}, \mathbf{t}) \notin M$. These cases cover the first two in the statement of the lemma.

(b) $P(\bar{a}) \notin M$. Two cases are possible now: $P_{\_}(\bar{a}, \mathbf{t}) \in M$ or $P_{\_}(\bar{a}, \mathbf{t}) \notin M$. In the first one, it also holds $P_{\_}(\bar{a}, \mathbf{t}^{\star\star})$, $P_{\_}(\bar{a}, \mathbf{t}^{\star}) \in M$, by rules 5. and 6.; and $P_{\_}(\bar{a}, \mathbf{f}) \notin M$ by rule 7. In the second case, $P_{\_}(\bar{a}, \mathbf{t}^{\star}) \notin M$ (because M is minimal), $P_{\_}(\bar{a}, \mathbf{f}) \notin M$ (because $P_{\_}(\bar{a}, \mathbf{t}^{\star}) \notin M$, and M is minimal). These cases cover the last two in the statement of the lemma.

□

From two database instances we can define a structure.

Definition 1
For two database instances D_1 and D_2 over the same schema and domain and a set of constraints $\mathcal{C}$, $M^\star_{\mathcal{C}}(D_1, D_2)$ is the Herbrand structure $\langle \mathcal{U}, I_{\mathcal{P}}, I_{\mathcal{B}} \rangle$, where $\mathcal{U}$ is the underlying domain,[29] and I_P, I_B are the interpretations for the database predicates (extended with annotation arguments), and the built-ins, respectively. I_P is inductively defined as follows:

1. If $P(\bar{a}) \in D_1$ and $P(\bar{a}) \in D_2$, then $P(\bar{a})$, $P\_(\bar{a}, \mathbf{t}^\star)$ and $P\_(\bar{a}, \mathbf{t}^{\star\star}) \in I_P$.
2. If $P(\bar{a}) \in D_1$ and $P(\bar{a}) \notin D_2$, then $P(\bar{a})$, $P\_(\bar{a}, \mathbf{t}^\star)$ and $P\_(\bar{a}, \mathbf{f}) \in I_P$.
3. If $P(\bar{a}) \notin D_1$ and $P(\bar{a}) \notin D_2$, then $P\_(\bar{a}, v) \notin I_{\mathcal{P}}$ for every annotation v.
4. If $P(\bar{a}) \notin D_1$ and $P(\bar{a}) \in D_2$, then $P\_(\bar{a}, \mathbf{t})$, $P\_(\bar{a}, \mathbf{t}^\star)$ and $P\_(\bar{a}, \mathbf{t}^{\star\star}) \in I_{\mathcal{P}}$.
5. For every RDEC $\psi \in \mathcal{C}$ of the form $\forall \bar{x}(P(\bar{x}) \rightarrow \exists \bar{y} Q(\bar{x}', \bar{y}))$: If $P\_(\bar{a}, \mathbf{t}^{\star\star}) \in I_{\mathcal{P}}$ and $Q\_(\bar{a}', \bar{b}, \mathbf{t}^{\star\star}) \in I_{\mathcal{P}}$, with $\bar{a} \neq null$ and at least one $b \in \bar{b}$, $b \neq null$, then $aux_\psi(\bar{a}') \in I_{\mathcal{P}}$.

The interpretation $I_{\mathcal{B}}$ is defined as expected: if Q is a built-in, then $Q(\bar{a}) \in I_{\mathcal{B}}$ iff $Q(\bar{a})$ is true in classical logic, and $Q(\bar{a}) \notin I_{\mathcal{B}}$ iff $Q(\bar{a})$ is false. □

Notice that the database instance associated to $M^\star_{\mathcal{C}}(D_1, D_2)$ through (B3) corresponds exactly to D_2, i.e. $D_{M^\star_{\mathcal{C}}(D_1, D_2)} = D_2$.

Lemma 2
Given an instance D and a set $\mathcal{C}$ of UDECs and RDECs, if $D' \models_N \mathcal{C}$, then there is a model M of the program $\Pi(\mathtt{P}, D)^M$ with $D_M = D'$. Actually, $M^\star_{\mathcal{C}}(D, D')$ is such a model.

PROOF**:** Since $D_{M^\star_{\mathcal{C}}(D,D')} = D'$, we only need to show that $M^\star_{\mathcal{C}}(D, D')$ satisfies all the rules of $\Pi(\mathtt{P}, D)^{M^\star_{\mathcal{C}}(D,D')}$. First, by construction, it is clear that rules 2., 5. and 6. are satisfied by $M^\star_{\mathcal{C}}(D, D')$.

For every UDEC in $\mathcal{C}$, the program has the rule in 3. If its body is satisfied, then the atoms $P_{i\_}(\bar{a}_i, \mathbf{t}^\star) \in M^\star_{\mathcal{C}}(D, D')$ and $Q_{i\_}(\bar{b}_i, \mathbf{f}) \in M^\star_{\mathcal{C}}(D, D')$ or $Q_i(\bar{b}_i) \notin M^\star_{\mathcal{C}}(D, D')$. Also, since the constraint is satisfied, at least one of the $P_i(\bar{a}_i)$ is not in D' or one of the $Q_i(\bar{b}_i)$ is in D'. By construction of $M^\star_{\mathcal{C}}(D, D')$, at least one of $P_{i\_}(\bar{a}_i, \mathbf{f})$ or $Q_{i\_}(\bar{b}_i, \mathbf{t})$ is in $M^\star_{\mathcal{C}}(D, D')$. Therefore, the head of the rule is also satisfied.

For every RDEC in $\mathcal{C}$, there are the rules 4. By construction of $M^\star_{\mathcal{C}}(D, D')$, for every $\psi \in \mathcal{C}$, those that define $aux_\psi(\bar{x})$ are satisfied.

If the body of the first rule in 4 is true in $M^\star_{\mathcal{C}}(D, D')$, it means that the constraint is not satisfied in the initial instance or at some point along the repair process. Since the constraint is satisfied by D', the satisfaction is restored by adding $Q\_(\bar{x}, \overline{null})$ or by deleting $P(\bar{x})$. This implies that $Q\_(\bar{x}, \overline{null}, \mathbf{t}) \in M^\star_{\mathcal{C}}(D, D')$ or $P\_(\bar{x}, \mathbf{f}) \in M^\star_{\mathcal{C}}(D, D')$. As a consequence, the first (or second) rule is satisfied. Then,

[29] In this case it can be restricted to the active domain of the neighborhood instance D (or the union of the active domains of D_1, D_2) plus the constant *null*.

by construction of $M^\star_{\mathcal{C}}(D, D')$, $P\_(\bar{a}, \mathbf{f}) \in M^\star_{\mathcal{C}}(D, D')$, and the head of the rule is satisfied. □

The next lemma shows that if M is a minimal model of the program $\Pi(\mathtt{P}, D)^M$, then D_M satisfies the constraints.

Lemma 3
Given a database D and a set of constraints $\mathcal{C}$, if M is a stable model of the program $\Pi(\mathtt{P}, D)$, then $D_M \models_N \mathcal{C}$.

PROOF**:** We want to show that $D_M \models_N \psi$, for every constraint $\psi \in \mathcal{C}$. There are two cases to consider:

A. IC ψ is a UDEC. Since M is a model of $\Pi(\mathtt{P}, D)^M$, M satisfies rules 3. of $\Pi(\mathtt{P}, D)$. Then, at least one of the following cases holds:
   (a) $P_i\_(\bar{a}, \mathbf{f}) \in M$. Then, $P_i(\bar{a}, \mathbf{t}^{\star\star}) \notin M$ and $P(\bar{a}) \notin D_M$ (by Lemma 1). Hence, $P_i(\bar{a}) \notin D_M$. Since the analysis was done for an arbitrary value $\bar{a}$, $D_M \models_N \bigwedge_{i=1}^n P_i(\bar{x}_i) \rightarrow \bigvee_{j=1}^m Q_j(\bar{y}_j) \vee \varphi$ holds.
   (b) $Q_j\_(\bar{a}, \mathbf{t}) \in M$. It is symmetric to the previous one.
   (c) It is not true that $M \models_N \bar{\varphi}$. Then $M \models_N \varphi$. Hence, φ is true, and $D_M \models_N \bigwedge_{i=1}^n P_i(\bar{x}_i) \rightarrow \bigvee_{j=1}^m Q_j(\bar{y}_j) \vee \varphi$ holds.
   (d) $P_i\_(\bar{a}, \mathbf{t}^\star) \notin M$. Given that M is minimal, just the last item in Lemma 1 holds. This means $P_i(\bar{a}, \mathbf{t}^{\star\star}) \notin M$, $P_i(\bar{a}) \notin D_M$ and $P_i(\bar{a}) \notin D_M$. Since the analysis was done for an arbitrary value $\bar{a}$, $D_M \models_N \bigwedge_{i=1}^n P_i(\bar{x}_i) \rightarrow \bigvee_{j=1}^m Q_j(\bar{y}_j) \vee \varphi$ holds.
   (e) $Q_j\_(\bar{a}, \mathbf{f}) \notin M$ or $Q_j(\bar{a}) \in M$. Given that M is minimal, just the first item in Lemma 1 holds. Then, $Q_j\_(\bar{a}, \mathbf{t}^{\star\star}) \in M$, $Q_j(\bar{a}) \in D_M$ and $D_M \models_N Q_j(\bar{a})$. Since the analysis was done for an arbitrary value $\bar{a}$, $D_M \models_N \bigwedge_{i=1}^n P_i(\bar{x}_i) \rightarrow \bigvee_{j=1}^m Q_j(\bar{y}_j) \vee \varphi$ holds.

B. Formula ψ is a RDEC. Since M is a model of $\Pi(\mathtt{P}, D)^M$, M satisfies rules 4. of $\Pi(\mathtt{P}, D)$. Then, at least one of the following cases holds:
   (a) $P\_(\bar{a}, \mathbf{f}) \in M$. Then, $P_i\_(\bar{a}, \mathbf{t}^{\star\star}) \notin M$ and $P(\bar{a}) \notin D_M$ (by Lemma 1). Hence, $P_i\_(\bar{a}) \notin D_M$. Since the analysis was done for an arbitrary value $\bar{a}$, $D_M \models_N (P(\bar{x}) \rightarrow Q(\bar{x}', y))$ holds.
   (b) $Q\_(\bar{a}', \overline{null}, \mathbf{t}) \in M$. It is symmetric to the previous one.
   (c) $P\_(\bar{a}, \mathbf{t}^\star) \notin M$. Given that M is minimal, just the last item in Lemma 1 holds. This means $P\_(\bar{a}, \mathbf{t}^{\star\star}) \notin M$, $P(\bar{a}) \notin D_M$ and $P(\bar{a}) \notin D_M$. Since the analysis was done for an arbitrary value $\bar{a}$, $D_M \models_N (P(\bar{x}) \rightarrow Q(\bar{x}', y))$ holds.
   (d) $aux_\psi(\bar{a}') \in M$. This means that $P\_(\bar{a}, \mathbf{t}^\star) \in M$ and there exists $\bar{b} \neq \overline{null}$ with $Q\_(\bar{a}', \bar{b}, \mathbf{t}^\star) \in M$, $Q\_(\bar{a}, \mathbf{f}) \notin M$, and then, that $P(\bar{a}) \in D_M$ and $Q(\bar{a}, \bar{b}) \in D_M$. Then, the constraint is satisfied.

□

Lemma 4
Let D and D' be instances over the same schema and domain. If M is a minimal

model of $\Pi(\mathtt{P}, D)^{M^\star_{\mathcal{C}}(D,D')}$ with $M \subsetneqq M^\star_{\mathcal{C}}(D, D')$, then there exists M' that is a minimal model of $\Pi(\mathtt{P}, D)^{M'}$ with $D_{M'} <_D D'$.

PROOF: Since M is a minimal model of $\Pi(\mathtt{P}, D)^{M^\star_{\mathcal{C}}(D,D')}$, $P(\bar{a}) \in M$ iff $P(\bar{a}) \in D$. By definition of $M^\star_{\mathcal{C}}(D, D')$ and $M \subsetneqq M^\star_{\mathcal{C}}(D, D')$, the only two ways that both models can differ is that, for some $P(\bar{a}) \in D$, $P\_(\bar{a}, \mathbf{f}) \in M^\star_{\mathcal{C}}(D, D')$ and neither $P(\bar{a})$ nor $P\_(\bar{a}, \mathbf{f})$ belong to M, or for some $P(\bar{a}) \notin D$, $\{P\_(\bar{a}, \mathbf{t}), P(\bar{a}, \mathbf{t}^\star), P\_(\bar{a}, \mathbf{t}^{\star\star})\} \subseteq M^\star_{\mathcal{C}}(D, D')$ and none of $P(\bar{a})$, $P\_(\bar{a}, \mathbf{t})$, $P(\bar{a}, \mathbf{t}^\star)$, $P\_(\bar{a}, \mathbf{t}^{\star\star})$ belong to M. Now, we can use the interpretation rules over M to construct M' that is a minimal model of $\Pi(\mathtt{P}, D)^{M'}$, as follows:

1. If $P(\bar{a}) \in M$ and $P\_(\bar{a}, \mathbf{f}) \notin M$, then $P(\bar{a}), P\_(\bar{a}, \mathbf{t}^\star)$ and $P\_(\bar{a}, \mathbf{t}^{\star\star}) \in M'$.
2. If $P(\bar{a}) \in M$ and $P\_(\bar{a}, \mathbf{f}) \in M$, then $P(\bar{a}), P\_(\bar{a}, \mathbf{t}^\star)$ and $P\_(\bar{a}, \mathbf{f}) \in M'$.
3. If $P(\bar{a}) \notin M$ and $P\_(\bar{a}, \mathbf{t}) \notin M$, then nothing is added to M'.
4. If $P(\bar{a}) \notin M$ and $P\_(\bar{a}, \mathbf{t}) \in M$, then $P\_(\bar{a}, \mathbf{t})$, $P\_(\bar{a}, \mathbf{t}^\star)$ and $P\_(\bar{a}, \mathbf{t}^{\star\star}) \in M'$.

It is clear that M' satisfies the coherence constraints, and is a minimal model of $\Pi(\mathtt{P}, D)^{M'}$.

It just rests to prove that $D_{M'} <_D D'$. First, we prove that $D_{M'} \leq_D D'$. Let us suppose $P(\bar{a}) \in \Delta(D, D_{M'})$. Then, either $P(\bar{a}) \in D$ and $P(\bar{a}) \notin D_{M'}$ or $P(\bar{a}) \notin D$ and $P(\bar{a}) \in D_{M'}$. In the first case, $P(\bar{a})$, $P\_(\bar{a}, \mathbf{t}^\star)$ and $P\_(\bar{a}, \mathbf{f})$ belong to M'. These atoms are also in M and, given the only two ways in which M and $M^\star_{\mathcal{C}}(D, D')$ can differ, they are also in $M^\star_{\mathcal{C}}(D, D')$. Hence, $P(\bar{a}) \in \Delta(D, D')$. In the second case, $P\_(\bar{a}, \mathbf{t})$ and $P\_(\bar{a}, \mathbf{t}^\star)$ belong to M'. These atoms are also in M and, given the only two ways in which M and $M^\star_{\mathcal{C}}(D, D')$ can differ, they are also in $M^\star_{\mathcal{C}}(D, D')$. Hence, $P(\bar{a}) \in \Delta(D, D')$.

We now prove that $D_{M'} <_D D'$. We know that, for some fact $P(\bar{a})$, $P\_(\bar{a}, \mathbf{t}) \in M^\star_{\mathcal{C}}(D, D')$ and $P\_(\bar{a}, \mathbf{t}) \notin M$, or $P\_(\bar{a}, \mathbf{f}) \in M^\star_{\mathcal{C}}(D, D')$ and $P\_(\bar{a}, \mathbf{f}) \notin M$. If $P\_(\bar{a}, \mathbf{f})$ is in $M^\star_{\mathcal{C}}(D, D')$ and not in M, then, $P(\bar{a}) \in \Delta(D, D')$, but $P(\bar{a}) \notin \Delta(D, D_{M'})$. Alternatively, if $P\_(\bar{a}, \mathbf{t})$ and $P\_(\bar{a}, \mathbf{t}^\star)$ belong to $M^\star_{\mathcal{C}}(D, D')$ but not to M, then $P(\bar{a}) \in \Delta(D, D')$, but $P(\bar{a}) \notin \Delta(D, D_{M'})$. Therefore, $D_{M'} <_D D'$. □

Proposition 1

Given a neighborhood instance D and a set $\mathcal{C}$ of UDECs and RDECs, if D' is a neighborhood solution for D with respect to $\mathcal{C}$, then there is a stable model M of the program $\Pi(\mathtt{P}, D)^M$ with $D_M = D'$. Furthermore, the model M corresponds to $M^\star_{\mathcal{C}}(D, D')$.

PROOF: By Lemma 2, $M^\star_{\mathcal{C}}(D, D')$ is a model of $\Pi(\mathtt{P}, D)^{M^\star_{\mathcal{C}}(D,D')}$. We now show that it is minimal. Assume, by contradiction, that there exists a model M of $\Pi(\mathtt{P}, D)^{M^\star_{\mathcal{C}}(D,D')}$ with $M \subsetneqq M^\star_{\mathcal{C}}(D, D')$. We can assume, without loss of generality, that M is a minimal model. Since $M \subsetneqq M^\star_{\mathcal{C}}(D, D')$, the model M contains the atom $P(\bar{a})$ iff $P(\bar{a}) \in D$.

By Lemma 4, there exists model M' such that $D_{M'} <_D D'$ and M' is a minimal model of $\Pi(\mathtt{P}, D)^{M'}$. By Lemma 3, $D_{M'} \models_N \mathcal{C}$. This contradicts that D' is a neighborhood solution. □

Proposition 2

If M is a stable model of $\Pi(\mathtt{P}, D)$, then D_M is a neighborhood solution for D with respect to $\mathcal{C}$.

PROOF**:** From Lemma 3, it holds $D_M \models_N \mathcal{C}$. We have to prove that it is $\leq_D$-*minimal*. Let us suppose there is a neighborhood solution D' (that satisfies $\mathcal{C}$) with $D' <_D D_M$. From Proposition 1, $M^\star_{\mathcal{C}}(D, D')$ is a stable model of $\Pi(\mathtt{P}, D)$ and $D_{M^\star_{\mathcal{C}}(D,D')} = D'$ (we denote it simple with $M^\star$ in the rest of the proof).

If $D' <_D D_M$, there is an atom $P(\bar{a}) \in \Delta(D, D_M)$, with $P(\bar{a}) \notin \Delta(D, D')$, or there is an atom $P(\bar{a}, \bar{b}) \in \Delta(D, D_M)$, with $\bar{a}', \bar{b} \neq null$, and an atom $P(\bar{a}, \overline{null}) \in \Delta(D, D')$. We analyze both cases:

1. $P(\bar{a}) \in \Delta(D, D_M)$ and $P(\bar{a}) \notin \Delta(D, D')$:
   Since $P(\bar{a}) \in \Delta(D, D_M)$, $P_\_(\bar{a}, \mathbf{t})$ or $P_\_(\bar{a}, \mathbf{f})$ belong to M. By Lemma 1, there are two cases:
   (a) $P(\bar{a})$, $P_\_(\bar{a}, \mathbf{t}^\star)$ and $P_\_(\bar{a}, \mathbf{f})$ belong to M, and no other $P_\_(\bar{a}, v)$, for v an annotation, belongs to M. $P(\bar{a})$, $P_\_(\bar{a}, \mathbf{t}^\star)$ and $P_\_(\bar{a}, \mathbf{t}^{\star\star})$ belong to $M^\star$, and, for any other annotation v, $P_\_(\bar{a}, v) \notin M^\star$.
   (b) $P_\_(\bar{a}, \mathbf{t})$, $P_\_(\bar{a}, \mathbf{t}^\star)$ and $P_\_(\bar{a}, \mathbf{t}^{\star\star})$ belong to M, and no other $P_\_(\bar{a}, v)$, for v an annotation, belongs to M. No $P_\_(\bar{a}, v)$, for v an annotation, belongs to $M^\star$.
   If an atom belongs to a model M_1, e.g. $P_\_(\bar{a}, \mathbf{f})$, and there is another model M_2 in which it is not present, then there must be in M_2 an atom annotated with $\mathbf{t}$ or $\mathbf{f}$ in order to satisfy the rule that was satisfied in M_1 by $P_\_(\bar{a}, \mathbf{f})$. This implies that $M^\star$ has an atom annotated with $\mathbf{t}$ or $\mathbf{f}$ that does not belong to M. This implies that there is an atom that belongs to $\Delta(D, D')$ and that does not belong to $\Delta(D, D_M)$. We have reached a contradiction, because $\Delta(D, D')$ is a proper subset of $\Delta(D, D_M)$.
2. $P(\bar{a}, \bar{b}) \in \Delta(D, D_M)$ and $P(\bar{a}, \overline{null}) \in \Delta(D, D')$:
   If $P(\bar{a}, \bar{b}) \notin M$, then $P_\_(\bar{a}, \bar{b}, \mathbf{t}) \in M$, $P(\bar{a}, \overline{null}) \notin M$, $P_\_(\bar{a}, \overline{null}, \mathbf{t}) \notin M$.
   Since $P(\bar{a}, \overline{null}) \in \Delta(D, D')$ and $P(\bar{a}, \overline{null}) \notin M$, $P_\_(\bar{a}, \overline{null}, \mathbf{t}) \in M^\star$.
   Since $P_\_(\bar{a}, \overline{null}, \mathbf{t}) \in M^\star$, there must be a rule representing a RDEC in $\Pi(D, \mathcal{C})$ such that $P_\_(\bar{a}, \overline{null}, \mathbf{t})$ is the only true atom in the head. For that rule to be also satisfied by M, there must be another atom in the head of that rule that is true in M but not in $M^\star$. This means that there is a $P(\bar{b}) \in \Delta(D, D_M)$ and $P(\bar{b}) \notin \Delta(D, D')$, which brings us back to case 1. above. Again we obtain a contradiction.

Therefore, it is not possible to have $D' <_D D_M$; and D_M is a neighborhood solution for D. □

References

BREWKA, G. 2004. Complex Preferences for Answer Set Optimization. In *Proc. International conference on Principles of Knowledge Representation and Reasoning*. AAAI Press, 213–223.

BUCCAFURRI, F., LEONE, N., AND RULLO, P. 2000. Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering 12,* 5, 845–860.

Chomicki, J. and Marcinkowski, J. 2005. Minimal-Change Integrity Maintenance using Tuple Deletions. *Information and Computation 197,* 1-2, 90–121.

De Giacomo, G., Lembo, D., Lenzerini, and Rosati, R. 2007. On Reconciling Data Exchange, Data Integration, and Peer Data Management. In *Proc. ACM Symposium on Principles of Database Systems*. ACM Press, 133–142.

Faber, W. and Woltran, S. 2011. Manifold Answer-Set Programs and Their Applications. In *Gelfond Festschrift*, M. Balduccini and T. C. Son, Eds. Springer LNAI 6565, 44–63.

Giannotti, F., Pedreschi, D., Sacca, D., and Zaniolo, C. 1991. Non-Determinism in Deductive Databases. In *Proc. International Conference on Deductive and Object-Oriented Databases*. Springer, LNCS 566, 129–146.

Herzig, A., Lorini, E., Hubner, J., Ben-Naim, J., Castelfranchi, C., Demolombe, R., Longin, D., and Vercouter, L. 2008. Prolegomena for a Logic of Trust and Reputation. In *Proc. Third International Workshop on Normative Multiagent Systems*. 143–157.

Hien Nguyen, G., Chatalic, P., and Rousset, M.-C. 2008. A Probabilistic Trust Model for Semantic Peer-to-Peer Systems. In *Proc. European Conference on Artificial Intelligence*. IOS Press, 881–882.

Schaefer, M. and Umans, Ch. 2008. Completeness in the Polynomial-Time Hierarchy: A Compendium. *SIGACT News*.

Xiong, L. and Liu, L. 2004. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions of Knowledge and Data Engineering 16,* 7, 843–857.

Yang, B. and Garcia-Molina, H. 2003. Designing a Super-Peer Network. In *Proc. International Conference on Data Engineering*. IEEE Computer Society, 49.