Distributed Consistency-Based Diagnosis

Vincent Armant, Philippe Dague, and Laurent Simon

LRI, Univ. Paris-Sud 11, CNRS and INRIA Saclay
Parc Club Université, 4 rue Jacques Monod 91893 Orsay Cedex, France
{vincent.armant,philippe.dague,laurent.simon}@lri.fr

Abstract. A lot of methods exist to prevent errors and incorrect behaviors in a distributed framework, where all peers work together for the same purpose, under the same protocol. For instance, one may limit them by replication of data and processes among the network. However, with the emergence of web services, the willing for privacy, and the constant growth of data size, such a solution may not be applicable. For some problems, failure of a peer has to be detected and located by the whole system. In this paper, we propose an approach to diagnose abnormal behaviors of the whole system by extending the well known consistency-based diagnosis framework to a fully distributed inference system, where each peer only knows the existence of its neighbors. Contrasting with previous works on model-based diagnosis, our approach computes all minimal diagnoses in an incremental way, without needs to get any conflict first.

1 Introduction

Model-Based Diagnosis has been introduced in the late eighties by [14, 11], and has since been widely used in many successful works. With this formalism, a logical theory describes the normal (and, optionally, abnormal) behavior of a physical system, and consistency checking against observations is used to derive hypotheses over components reliability (called *diagnoses*), that explain failures. Even if stronger logic may be used, it is often the case where propositional logic is chosen to model the system. In this context, diagnosing the system with respect to observations can be expressed as a classical – and heavily studied – knowledge based compilation problem: restricted prime implicants [6].

Recent years have seen an increasing number of AI works pushing forward the power of *distributed systems*, for instance by adding semantic layers [1]. In such networks, all systems (or "peers") are running the same algorithm, and are working for the same purpose. The framework may however describe two kinds of settings. One which allows any peer to communicate with any other peer (generally by means of distributed hash tables, [17]) and the other where peers only know their neighbors, which is closer to social networks, circuits, and web services composition. In the latter formalism, reasoning is based on the declaration of logical equivalence of variables between peers (the *shared variables*), which locally defines *subsystems* acquaintances.

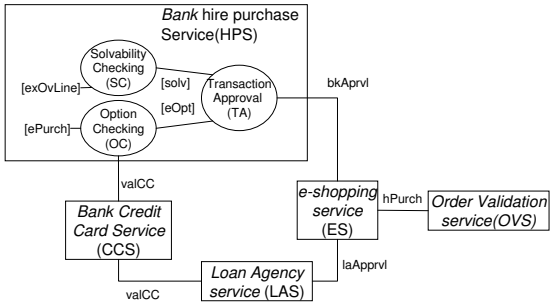

Fig. 1. 3 steps web-payment certification

In this paper, we investigate the problem of diagnosing distributed systems defined by peers acquaintances. Each peer only knows its neighborhood, and has a logical model of its normal and abnormal behavior with respect to its own local variables and its variables shared with its acquaintances, only some of them are observable. The challenging problem is to build a set of global diagnoses for the whole system. Our solution directly computes diagnoses (including all minimal ones for set inclusion) without conflicts analysis, a very hard task which is generally the first step – and the first bottleneck – of all previous model-based diagnoses engines, even when efficient algorithms are used [16].

In our approach, we focus on "static" settings of distributed systems (i.e. we do not deal with connection of new peers or disconnection of existing peers), in order to easily ensure that diagnoses and observations are consistent. If the static behavior is not possible in a fully peer-to-peer setting, it is more realistic in a distributed setting, for instance web services composition, embedded circuits, and social networks. In many cases, additional layers, like memory of past events and counters, can even simulate the "static" hypothesis.

In the next section, we introduce our notations, recall the principles of model based diagnosis and extend it to formulas in Disjunctive Normal Form. In section 3, we introduce our foundations of distributed reasoning for diagnosis. In section 4, we present the distributed algorithm and then we report related work and conclude.

Example 1 (Three steps web-payment certification). We illustrate the paper by a toy example of a web-payment certification, see figure 1. The order validation service (OVS) asks to an eshopping service (ES) for a hire purchase approval (hPurch). In order to maximize its sales opportunity, (ES) waits for the customer bank approval (bkAprvl) or a loan agency approval (laAprvl). The bank hire purchase service (HPS) and the loan agency service (LAS) both check the customer credit card validity (valCC) by a call to the credit card service (CCS). In the following, we restrict the system to (HPS) and will refer to its "global description" as the conjunction of the Transaction Approval (TA), the Solvability Checking (SC) and the Option Checking (OC).

2 From CNF Diagnosis to DNF Diagnosis

We assume familiarity with the standard literature on propositional reasoning and resolution. A literal is a variable v or its negation $\neg v$. Given a set L of literals, we denote by $\overline{L}$ the set of its opposite literals. A Conjunctive Normal Form formula (CNF) is a conjunction of clauses (disjunctions of literals). A Disjunctive Normal Form formula (DNF) is a disjunction of products (conjunctions of literals). For simplicity, we identify a formula with a set of sets of literals. We denote by $T^{\wedge}$ (resp $T^{\vee}$) the set of sets of literals corresponding to a CNF (resp DNF).

A *model* is a set of literals that, when assigned to true, allows the evaluation of a given formula to true. We say that a formula f is satisfiable (or consistent), denoted by $f \not\models \bot$, if there exists a model for f. Let f_1, and f_2 be two formulas, if all the models of f_1 are also models of f_2, which is noted $f_1 \models f_2$, then f_1 is called an *implicant* of f_2 and f_2 is called an *implicate* of f_1. The minimal (with respect to the order relation induced by inclusion of sets of literals) implicate clauses (resp. implicant products) of a formula are called the prime implicates (resp. prime implicants) of this formula. The set of prime implicates is expressed in CNF whether the set of prime implicants is in DNF. Given a formula f and a subset V of its variables, the restriction of f on V is denoted by $f|_V$ and corresponds to recursively apply on f the Shannon decomposition operator on all variables x of f that do not appear in V. This operation, known as *forgetting* in Knowledge Compilation, is well known to be a NP-Hard problem. However, when f is expressed as a DNF, the restriction operator is simply a vocabulary restriction of all products of f. The restriction of $T^{\vee}$ on a set of literals L can be defined as $T^{\vee}|_{\{L\}} = \{I|_{\{L\}} | \exists I \in T^{\vee} \text{ s.t. } I|_{\{L\}} = I \cap L\}$, which is no more a hard task.

2.1 Centralized Model-Based diagnosis

Like many other works, we adapt the model-based diagnosis framework from [11, 8] to the propositional case. Initially, an observed system is a triple (SD, COMPS, OBS) where SD is a first order logical formula describing the system behavior, OBS is a formula describing the observations (that boils down frequently to values assignment to observable variables) and COMPS is the set of monitored components, that appear as arguments of the predefined predicate $Ab()$ in SD ($Ab(C_i)$ denoting that component C_i is abnormal). In propositional logic, we may merge the whole into a single theory T, with the naming convention: all variables okC_i (called mode variables) encode the correct behavioral modes of the components C_i, i.e. $\neg Ab(C_i)$. We note F the set of negative mode literals $\{...,\neg okC_i,...\}$ representing faulty components. For a (boolean) observable coded by a variable v, the elementary observation $v = a$ is coded by v if a equals 1 and $\neg v$ if a equals 0.

Example 2 (Modeling the system). A correct behavior of TA (okTA) will approve a hire purchase (bkAprvl) if the customer is solvent (solv) and fulfills the condition

(eOpt) of OC. The rule for TA is rewritten as $f(TA) : okTA \Rightarrow (solv \wedge eOpt \Leftrightarrow bkAprvl)$*. A normal functioning of SC (okSC) will consider a customer solvent (solv) if he does not exceed his overdraft limit* ($\neg exOvLine$)*. We obtain* $f(SC) : okSC \Rightarrow (\neg exOvLine \Leftrightarrow solv)$*. A correct behavior of OC (okOC) will satisfy (eOPt) if the customer asked for hire purchases by internet (ePurch) and his credit card is valid (valCC). There are only two possible failures for OC: when ePurch keeps its default value (i.e. no internet purchase) whereas the customer asked for the internet option, and when the customer card is believed invalid whereas it is valid. The Option Checking system can thus be encoded by* $f(OC) : okOC \Rightarrow (ePurch \wedge valCC \Leftrightarrow eOpt) \wedge (\neg okOC \Rightarrow \neg valCC \vee \neg ePurch)$*. The behavior of HPS is the conjunction* $f(HPS) : f(OC) \wedge f(SC) \wedge f(TA)$*.*

A diagnosis is a behavioral mode assignment to each component of the system, consistent with its theory.

Definition 1 (Minimal Diagnoses). *Let* T *be the theory that describes an observed system,* F *the consistent set of all negative mode literals of the system.*

$$\triangle \subseteq F \text{ is a diagnosis for } T \text{ iff } T \cup \triangle \cup \overline{\{F \setminus \triangle\}} \not\models \bot$$

We write $Diag(T)$ *the set of diagnoses of* T *and* $min_{\subseteq}(Diag(T))$ *the set of its minimal diagnoses.*

Intuitively, this definition states that, given any minimal diagnosis $\triangle$, one may suppose that all components C' that do not appear in $\triangle$ are correct. We may also notice that, because we can restrict the set of possible failures ($(\neg okOC \Rightarrow \neg valCC \vee ePurch)$ in the previous example), a minimal diagnosis may not be extended by supposing some component C' incorrect (a negative mode literal candidate for extending a diagnosis can be inconsistent with T and the other mode literals).

The theorem 3 of [8] states that the minimal diagnoses are the prime implicants of the conjunction of minimal conflicts, where the minimal conflicts (called minimal conflict sets in [14] and minimal positive conflicts in [8]) are the prime implicates of the formula $SD \wedge OBS$, restricted to the mode literals in F. Intuitively, a minimal conflict refers to a set of components containing at least a faulty one. Minimal diagnoses are thus the minimal (for literals set inclusion) conjunctions of faulty components that can explain all the conflicts, according to observations.

Example 3 (Conflicts and Diagnoses on Scenario 1). Let us suppose the following scenario: the bank approved a hire purchase for an operation whereas the customer exceeds his overdraft limit. He had a valid credit card but asked to stop internet purchasing. In this case the bank service does not fulfill its expected behavior.

We thus look for the minimal subsets of $\{TA, OC, SC\}$ *that may be faulty. This will be expressed by minimal conjunctions of literals from* $\{\neg okTA, \neg okOC, \neg okSC\}$*, which are consistent with the formula* $f(HPS)$ *and the observations* $\{exOvLine, valCC, \neg ePurch, bkAprvl\}$*. The minimal conflicts are* $(\neg okSC \vee \neg okTA)$ *and* $(\neg okOC \vee \neg okTA)$*. The minimal diagnoses that satisfy these conflicts are:* $(\neg okTA)$ *and* $(\neg okSC \wedge \neg okOC)$*.*

Most of previous works on diagnosis compute first the set of conflicts, restricted to mode literals. Then, only when this first stage is over, diagnoses can be computed. These methods are hopeless for building an incremental diagnostic engine as all minimal conflicts have to be known before the first diagnosis can be returned. They are nevertheless motivated by the fact that models of real-world systems are supposed to be close to CNF. If needed, new variables are usually added to practically contain the potential blow-up when translating to CNF. Up to now very few interest has been shown in DNF representations of a system. However, such a DNF representation is very advantageous for diagnosis: if we ensure that the set F of mode variables is consistent, which means that no variable appears both positively and negatively in F, then, if $T^\vee$ is the description of an observed system, each product of the restriction $T^\vee|_{\{F\}}$ is a diagnosis (not necessarily minimal).

Lemma 1. *Let $T^\vee$ be a DNF description of an observed system, and F a consistent set of negative mode literals, then*

$$\forall I \in T^\vee, I|_{\{F\}} \in Diag(T)$$

Sketch of Proof For each $I \in T^\vee$, I is an implicant of T which is trivially consistent with T. Let us consider $\{F \setminus I|_{\{F\}}\}$, which does not contain any literals from $I|_{\{F\}}$. Thus, $T \cup I|_{\{F\}} \cup \{F \setminus I|_{\{F\}}\}$ is consistent and by definition $I|_{\{F\}}$ is a diagnosis.

Consequently, if we compute at least one implicant of T, we obtain at least one diagnosis without waiting for conflicts. In practice, our requirement about a DNF representation of T can be weakened without loss: implicants can be incrementally computed by an efficient SAT solver or even deduced from a compact, compiled, representation of T that allows efficient production of models [6]. The result is that, on small systems, or on large – but distributed – systems, the direct translation from CNF to DNF can be done. The following theorem states that minimal diagnoses are contained in any DNF description encoding the observed system.

Theorem 1. *Let T be the description of an observed system, and F a consistent set of negative mode literals:*

$$min_\subseteq(Diag(T)) = min_\subseteq(T^\vee|_{\{F\}})$$

Sketch of Proof A) Let $\triangle$ be a minimal diagnosis, by the definition 1, $\triangle \cup \overline{\{F \setminus \triangle\}}$ is consistent with the observed system. Thus, for any DNF representation of the system, there exists an implicant I consistent with $\triangle \cup \overline{\{F \setminus \triangle\}}$. Since I is consistent with $\triangle \cup \overline{\{F \setminus \triangle\}}$ we have $I|_{\{F \setminus \triangle\}} = \emptyset$ and thus $I|_{\{F\}} = I|_{\{\triangle\}}$. Because we know that $I|_{\{F\}}$ is a diagnosis and $\triangle$ *is a minimal one* we have $I|_{\{\triangle\}} = \triangle$. **B**) Let $I|_{\{F\}} \in min_\subseteq(T^\vee|_{\{F\}})$, $I|_{\{F\}}$ is a diagnosis, suppose that it is not a minimal one. Then there exists a diagnosis $\triangle$, s.t. $\triangle \subset I|_{\{F\}}$. Consequently, there exists l in $I|_{\{F\}}$ s.t. l is not in $\triangle$. In this case $\triangle \cup \overline{\{ F \setminus \triangle\}} \cup I$ is contradictory. But since $\triangle \cup \overline{\{ F \setminus \triangle\}}$ is consistent with $T^\vee$, then there exists $I' \neq I$ s.t. $\triangle \cup \overline{\{ F \setminus \triangle\}} \cup I'$ is consistent and $I'|_{\{F\}} \subseteq \triangle$. We deduce that $I'|_{\{F\}} \subseteq \triangle \subset I|_{\{F\}}$. It is contradictory with the fact that $I|_{\{F\}} \in min_\subseteq(T^\vee|_{\{F\}})$.

Example 4 (Finding Diagnoses in DNF theory). Let $F_{HPS} = \{\neg okTA, \neg okOC, \neg okSC\}$ *be the set of mode literals and* $T^{\vee}_{HPS}$ *the DNF formula of the description of HPS with observations.*

$$T^{\vee}_{HPS} = \begin{array}{l}(\neg okTA \wedge \neg okSC \wedge \neg eOpt \wedge okOC) \vee \\ (\neg okTA \wedge \neg okSC \wedge \neg okOC) \vee \\ (\neg okTA \wedge \neg solv \wedge \neg okOC) \vee \\ (\neg okTA \wedge \neg okSC \wedge \neg eOpt) \vee \\ (\neg okTA \wedge \neg solv \wedge \neg eOpt) \vee \\ (\neg okSC \wedge \neg okOC \wedge eOpt \wedge solv)\end{array}$$

For simplicity, we omitted, in each product, the conjunction of observed literals $exOvLine \wedge valCC \wedge \neg ePurch \wedge bkAprvl$. *Finally, after restriction on* F_{HPS} *and subsumption elimination, we obtain the two diagnoses* $\{\neg okTA, (\neg okSC \wedge \neg okOC)\}$.

By the lemma 1 we know that each implicant contains a diagnosis. The theorem 1 states that any DNF description of the observed system contains the set of minimal diagnoses. Now, suppose that we monitor and diagnose a distributed system by the means of a distributed diagnostic architecture made up of local diagnostic engines which gradually compute local implicants from the monitored subsystems. A consistent composition of local implicants from each diagnostic engine is actually an implicant for the global system. We note that, as soon as each diagnostic engine returns its first implicant, the composition task can start. In the next section we precise the notion of distributed system which differs from the usual notion of system in diagnosis by taking into account the shared and local acquaintance of each subsystem. We take advantage of this characterization for forgetting symbols and optimizing the composition task.

3 Diagnosing Peer-To-Peer Settings

We formalize our distributed model based diagnosis framework by means of Peer-to-Peer Inference Systems (P2PIS) proposed by [9], and extended in [1] for distributed reasoning. In a P2PIS, a so-called "inference peer" has only a partial knowledge of the global network (generally restricted to its acquaintance) and may have local information, only known by itself. In our work, an inference peer will for instance model the expected behavior of a real peer, a web service, or a subcircuit, of a distributed system. Let us denote by T the description of the global observed system. T is the (virtual) conjunction of all local theories T_p of peers p. Of course, in our framework, T will never be explicitly gathered and privacy of local knowledge will be ensured. T is built on the global variables vocabulary V (excluding mode variables), which can be partitioned into shared variables Sh and local variables Loc

- $Sh = \{v | \exists p \neq p' \text{s.t. } v \text{ appears both in } T_p \text{ and in } T_{p'}\}$
- $Loc = V \setminus Sh$

In addition to this partition, we have to add mode variables in order to be able to diagnose the system. We denote by F the set of all mode variables of the system. Obviously, in order to build a global diagnosis, exchange of mode

variables between peers has to be possible. Thus, the network will allow formulas built on variables from $Sh \cup F$ to be sent from peer to peer. We denote by V_p, Sh_p, Loc_p, F_p the vocabulary, the shared variables, the local variables and the mode variables symbols of any inference peer p.

3.1 A network of DNF models

In the previous section, we assumed that we were able to work directly on the DNF of T. Because T here is a conjunction of formulas, we may push this hypothesis to all T_p. If the first hypothesis, i.e. in the centralized case, may not be considered as a realistic one, at the opposite small peers will admit relatively small DNF encoding, and thus the second hypothesis, i.e. in the distributed case, is of practical interest. If all T_p are in DNF, then writing T in DNF can be done by the distribution property of $\wedge$ over $\vee$. More formally, we use the following operator for this purpose:

Definition 2 (Distribution ($\otimes$)).
$T_1^\vee \otimes T_2^\vee = \{I_1 \wedge I_2 | I_1 \in T_1^\vee, I_2 \in T_2^\vee, I_1 \wedge I_2 \not\models \bot\}$

One may notice that inconsistent products are deleted, and, if the result is minimized, then this operator is exactly the *clause-distribution* operator of [16], but applied to DNF and products.

Because of privacy, and for efficiency purpose, let us introduce the following lemma stating that instead of distributing all theories before restricting the result to mode variables, one may first restrict all theories to shared and mode variables without loss.

Lemma 2. *Let $T^\vee$ be a description of an observed P2P system, F a consistent set of negative mode literals:*

$$(\otimes T_p^\vee)|_{\{Sh,F\}} = \otimes(T_p^\vee|_{\{Sh_p,F_p\}})$$

Sketch of Proof Let I (resp. I') an implicant of $T_p^\vee$, (resp. $T_{p'}^\vee$). Local symbols from I do not appear in I', thus inconsistencies between I and I' can only come from shared symbols.

With this lemma and the first theorem we can show that minimal diagnoses can be computed with shared and mode literals only.

Theorem 2. *Let T be a description of an observed P2P system, F a consistent set of negative mode literals:*

$$min_\subseteq(Diag(T)) = min_\subseteq((\otimes(T_p^\vee|_{\{Sh_p,F_p\}}))|_{\{F\}})$$

Sketch of Proof Let $T^\vee$ be a DNF global description of the observed system s.t. $T^\vee \equiv T$. By theorem 1 we know that $min_\subseteq(Diag(T)) = min_\subseteq(T^\vee|_{\{F\}})$. We have $T^\vee|_{\{F\}} = T^\vee|_{\{Sh,F\}}|_{\{F\}}$ since the restriction of $T^\vee$ on shared and faulty symbols does not delete any faulty symbol. Moreover, because $T^\vee \equiv \otimes T_p^\vee$, we deduce by the lemma 2 that $min_\subseteq(Diag(T)) = min_\subseteq((\otimes T_p^\vee|_{\{(Sh_p,F_p)\}})|_{\{F\}})$.

3.2 Distributions with Trees

We now focus on the distribution of consistent diagnoses between diagnostic engines. Here we consider that any peer may be able to initiate a diagnosis and may ask its neighborhood to help him for this task. When receiving a request for a diagnosis by an initiator, a peer will also query its acquaintances, according to its observation values, and will begin to answer to its initiator as soon as possible. Thus, the initial request will flood into the network top-down and answers will converge to the initial peer with a bottom-up traversal of the network. Implicitly, for a given request for a diagnosis, any peer will maintain who was its local initiator, and thus an implicit tree will be built in the network for each request.

We use this tree to efficiently compute the distribution of peers theories. Let us denote by A_p the subtree rooted in p and by $child(A_p, A_{p'})$ the relation between $A_{p'}$ and A_p s.t. $A_{p'}$ is a subtree of A_p. We note by Sh_{A_p} the variables shared by A_p and any other peer in the distributed system. We note T^{A_p} the theory defined as the conjunction of the theories of all peers occurring in the subtree rooted in p.

$$T^{A_p} = \begin{cases} T_p^{\vee}|_{\{F_p, Sh_p\}}, \text{ if } \nexists p' \text{s.t.} child(A_p, A_{p'}) \text{ is set.} \\ (T_p^{\vee}|_{\{F_p, Sh_p\}} \otimes \bigotimes_{\{A_{p'} | child(A_p, A_{p'})\}} T^{A_{p'}})|_{\{Sh_{A_p}, F_{A_p}\}}, \text{otherwise} \end{cases}$$

The next theorem shows that we can compute global diagnoses by gradually forgetting shared acquaintances which correspond to local acquaintances of a subtree.

Theorem 3. *Let T be the global description of an observed system, $child(A_p, A_{p'})$ a relation defining a Tree on T rooted in r. then:*

$$min_{\subseteq}(Diag(T)) = min_{\subseteq}(T^{A_r})$$

Sketch of Proof We use the theorem 2 and inductively prove that $\forall p, T^{A_p} = (\bigotimes_{q \in A_p} T_q^{\vee}|_{\{Sh_q, F_q\}})|_{\{Sh_{A_p}, F_{A_p}\}}$. Concerning the root r, we note that $Sh_{A_r}=\emptyset$, consequently $min_{\subseteq}(T^{A_r})$ only contains the set of minimal diagnoses.

Thus, intuitively, as soon as we know that a given variable cannot imply any inconsistency in other parts of the tree, we remove it. As answers will go back to the root, peers will filter out useless variables, and, hopefully, will reduce the number and the size of possible answers.

4 Algorithm

In this section, we present our message-passing algorithm M2DT, standing for "Minimal Diagnoses by Distributed Trees" (see algorithm 1). We call *neighbor* of p a peer that shares variables with p. As previously, A stands for the distributed cover tree, dynamically built by the algorithm. We write A_p the subtree of A rooted in p. For a tree A and a peer p, p's *parent* and p's *children* will be included, by construction, in p's neighborhood. Let us recall that T^{A_p} is defined

as the theory (more exactly a subpart of the whole theory, sufficient for diagnostic purpose) of the observed subsystem defined by the conjunction of all peers occurring in the whole subtree A_p. We call *r-implicant* of T^{A_p} a restriction of one implicant of T^{A_p} to its mode variables and shared vocabulary.

4.1 A general view on M2DT

At the beginning, a given peer, called the *starter*, broadcasts a request of diagnosis ($reqDiag$) to its neighborhood. When a peer receives its first $reqDiag$, it sets the sender as its parent and broadcasts the request to its remaining neighbors, in order to flood the network. This first stage of the algorithm aims at building a distributed cover tree: as the request goes along the network, the relationship ($parent$, p) is set and defines the distributed cover tree A. As soon as one peer knows that it is a leaf in A, it answers by sending its r-implicants ($respDiag$) to its parent and thus begins the second stage of the algorithm. When an intermediate node receives r-implicants from one of its children, there are two cases. If it already knows the role of all its neighborhood (parent, direct children and peers that can either occur deeper in the current subtree or elsewhere in the cover tree), it extends all new r-implicants by distributing them over its own r-implicants and those already received from all other children. It then filters out useless variables and sends all resulting implicants to its parent. If it does not know the role of all its neighborhood, it stores the received r-implicants for a future use. With this algorithm, global diagnoses converge to the starter peer. When a peer does not wait any more for any message, it sends its termination message to its parent. When a peer has received all termination messages from all its children, it sends its termination message to its parent (third and last stage of the algorithm). When the starter peer receives the termination message, we are sure that it already received the set of minimal diagnoses from all its children.

4.2 Structures and algorithm

A message can be a request of diagnosis $reqDiag$, a response $respDiag$ or a notification of termination $endDiag$. The structure of a message msg is the following one:

- $msg.Type$: takes its values in $\{reqDiag, respDiag, endDiag\}$, matching the three stages of the algorithm.
- $msg.Desc$: defined only when $msg.Type = respDiag$, represents the descendants of the sender of the message that participated in building the considered r-implicant.
- $msg.rImpl$: defined only when $msg.Type = respDiag$, is an r-implicant of the subtree rooted in the sender of the message.

A peer p sets its $parent$ to the first peer in its neighborhood that sent it a $reqDiag$ message. For all other $reqDiag$ messages that p may receive, it adds

the sender to the set $NotChild$. This set stores all peers that are not direct children of p (peers that can occur deeper in the subtree rooted in p or that do not occur in this subtree). All peers p' that send to p at least one $respDiag$ message are stored in $Child$. The array $TChild$, defined in each peer p only for its direct children p', associates to each child peer p' a DNF theory $TChild[p']$. $TChild[p']$ stores all r-implicants received so far from p'. This set will be known to be complete when p will receive an $endDiag$ message from p'. In order to detect additional useless variables, p also stores in $Desc$ all known descendant of p (peers occurring in the subtree rooted in p). All local variables of all peers are already deleted by the algorithm, but one may now consider as "local" a variable that is guaranteed to occur only in the current subtree and not elsewhere. This is the case for shared variables that are shared only by peers that occur in the current subtree.

To detect and notify termination, p maintains a list of peers from which messages are still waited. This list is called $waitEnd$ and initially set to all neighbors of p ($Neighborhood$). A peer leaves the list if it is the parent, if it is not a direct descendant or if it is a direct child that notified termination.

4.3 Primitives of M2DT

checkEnd $(waitEnd, p')$ Checks and propagates the termination. First, it removes p' from the $waitEnd$ list of p. If $waitEnd$ is empty, it sends the termination message and terminates.

extends $(I, T_p^\vee, TChild, Desc)$ Extends the implicant I from p' by distributing it on the local theory $T_p^\vee$ and all sets $TChild[p"]$ that are defined and different from p'. This primitive, which is only called when the local subtree is entirely known, computes
$(T_p^\vee|_{\{F_p, Sh_p\}} \otimes I \bigotimes_{p" \neq p'} TChild[p"])|_{\{Sh_{A_p}, F_{A_p}\}}$
One may notice that Sh_{A_p} is not directly known. It is deduced from $Desc$: we associate in $Desc$ to each shared variable the unique identifiers of all peers that share it. Thus, one may check if all peers that share a given shared variable are "local" to the subtree, only with the help of the set $Desc$.

flush $(T_p^\vee, TChild, Desc)$ Sends the distribution of all implicants stored in the $TChild$ array and the local theory. This primitive is called only when the local subtree is known to be complete for the first time with a $reqDiag$ message, which means that the last unknown neighbor sent to p a message "I am not your direct child". We thus have to flush all previously stored implicants (if any) to p's parent. This primitive computes
$(T_p^\vee|_{\{F_p, Sh_p\}} \bigotimes_{p" \neq p'} TChild[p"])|_{\{Sh_{A_p}, F_{A_p}\}}$

One may notice that this primitive will be called for all leaves of the distributed tree A.

Algorithm 1 Peer p receives a message msg from peer p'

```
1: switch msg.Type
2:
3:  case : reqDiag
4:    /*A distributed tree is built*/
5:      if parent is not set then /* Flooding alg.*/
6:          parent ← p'
7:          send to all p neighborhood \p' : msg [reqDiag]
8:      else /* p' is not a direct child */
9:          NotChild ← NotChild ∪ {p'}
10:     end if
11:     /* Flushes all stored implicants when the subtree is known */
12:     if {parent} ∪ Child ∪ NotChild = Neighborhood
13:         Π ← flush(T_p^∨, TChild, Desc)
14:         for all I ∈ Π
15:           send to parent msg [respDiag,I,Desc ∪ {p}]
16:         end for
17:     end if
18:     /* p' is either the parent or not a direct child*/
19:     checkEnd(waitEnd, p')
20:
21:  case : respDiag
22:    /* Stores the diag, or extends and propagates it */
23:     Child ← Child ∪ {p'}
24:     Desc ← Desc ∪ msg.Desc
25:     TChild[p'] ← TChild[p'] ∪ msg.rImpl
26:    /* Extends msg.rImpl only if the subtree is already known */
27:     if {parent} ∪ Child ∪ NotChild = Neighborhood
28:         Π ← extends(msg.rImpl, T_p^∨, TChild, Desc)
29:         for all I ∈ Π
30:           send to parent msg [respDiag,I,Desc ∪ {p}]
31:         end for
32:         Tresult ← min_⊆(Tresult ∪ Π)
33:     end if
34:
35:  case : endDiag
36:  /* Notifies termination of this child, and propagates if needed*/
37:     checkEnd(waitEnd, p')
38: end switch
```

4.4 Properties

In the following we assume a FIFO channel between two connected peers and no lost message. The acquaintance graph is connected and the global theory is satisfiable. Messages processing is considered as "atomic", which simply means that the messages are treated one by one by each peer.

Let us first emphasize some observations.

Lemma 3. *If a peer, p, sends a reqDiag to one of its neighbors, p', this message is the only one from p to p'.*

Proof A peer broadcasts the *reqDiag* to each of its neighbors (except its parent) just after having set as its parent the sender of the first received *reqDiag*. Because of the condition line 5, p does not have the opportunity to send other *reqDiag*. Concerning *respDiag* and *endDiag* they are sent to the parent only.

If we now focus on the first event carried out by each peer:

Lemma 4. *All peers, except the starter, will receive a first event, which will be a reqDiag message.*

Proof To receive a *respDiag* or an *endDiag*, p has to be a *parent* of some peer. But to become a parent, p must send a *reqDiag* to at least one peer, and thus p will have to receive *reqDiag* first. Consequently, since the acquaintance graph is connected, and because a peer broadcasts *reqDiag* to its neighbors, then each peer will receive a first *reqDiag* (we rely on the well known flooding algorithm in a graph to ensure this).

With these lemmas we have the following property:

Property 1 (Distributed cover Tree). The relation (*parent*,p), built when p receives its first *reqDiag*, defines a distributed cover tree.

Sketch of Proof Let n be the number of peers, $req_1(p)$ be the reception of the first diagnosis request by p. Since the starter peer does not get any parent, by the previous lemma we know that flooding *reqDiag* will build n-1 connections (parent,p). Suppose a cycle is defined by these connections and an order $<$ s.t. $req_1(p) < req_1(p')$ if $req_1(p)$ is former than $req_1(p')$. Let us take p in the cycle, there exists p' in the cycle s.t. p' got p as parent, then $req_1(p) < req_1(p')$. If we follow the "parent" connection in the cycle, we have by transitivity that $req_1(p') < req_1(p)$. Consequently, we cannot have cycle by the "parent" connection.

Now we know that a distributed cover tree will be built but, at this point, a peer p will only know its parent, but not its direct children. This can be deduced by the *respDiag* messages, when all children of p will send it their r-implicants. *respDiag* messages are only sent when the state of all neighbors of p are known (Parent, Child, NotChild).

Lemma 5. *Let p be a peer, p' be one of its neighbors. If p' does not get p as parent, p will receive a reqDiag from p'.*

Sketch of Proof Let p' be a neighbor of p that does not accept p as its parent. When p sent to it a *reqDiag*, p' already had a parent in order to refuse p as its parent. Consequently p had also sent to p' a *reqDiag*.

With this lemma, we can easily show the following property:

Property 2. Let A_p be the subtree rooted in p and built by the algorithm, T^{A_p}, the theory of A_p as previously defined, then p will send to its parent the r-implicants of T^{A_p}.

Sketch of Proof Let us prove this property recursively, with respect to the maximal depth, d_{max}, of the subtree A_p, rooted in p. If d_{max}=0, p is a leaf, none of its neighbors

gets it as parent. Nevertheless, with the previous lemma, we know that each of them will send a $reqDiag$ to it. Consequently, p will know the state of its neighbors and satisfy the condition ($parent \cup Child \cup NotChild = Neighborhood$). Then p will send all r-implicants of $T_p^{\vee}$ computed by the flush primitive. If $d_{max} > 0$, suppose the property true for all children p' of p: we thus guarantee that p will receive from each of them their r-implicants and will store them in its $TChild$ array. Concerning the other neighbors (NotChild), p will receive from them a $reqDiag$. Consequently, p will know the state of all its neighborhood and satisfy the condition ($parent \cup Child \cup NotChild = Neighborhood$). Since the global theory is satisfiable p will be able to build at least one r-implicant of T^{A_p} either by the method extends or by the method flush.

Since $TResult$ is minimized, the starter peer will save in it the minimal diagnoses. The correction and the completeness of the algorithm are a direct consequence of the previous property. The termination of the algorithm is shown by the following one:

Property 3. The last event carried out by a peer is sending the *endDiag* message to its parent.

Sketch of Proof Similarly to the previous property, we can show this property recursively with respect to the maximal depth, d_{max}. If $d_{max} = 0$, p will receive a $reqDiag$ from all its neighbors and the set $waitEnd$ will be empty. Then p will send the message *end* after the set of r-implicants of $T_p^{\vee}$. If $d_{max} > 0$, p will receive a $reqDiag$ from peers in NotChild and an *endDiag* from its children. Then, the set $waitEnd$ will be empty. As soon as possible, p will send r-implicants to its parent and an *endDiag* as its last message.

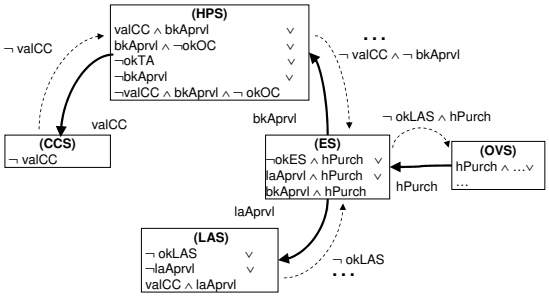

Fig. 2. M2DT algorithm

Example 5 (M2DT Illustration). A customer made a hire purchase by internet whereas his credit card was not valid: the service CCS observed ¬valCC and the service ES observed hPurch. OVS starts the analysis and sends a diagnosis request to ES. ES begins the computation of $T_{ES}^{\vee}|_{\{Sh_{ES}, F_{ES}\}}$ and forwards the request to its neighbors HPS and LAS. HPS and LAS receive the diagnosis request from ES and both forward a diagnosis request to CCS. CCS forwards the request to LAS. When LAS receives the request from CCS it has already received one from ES, so it does not answer. At this

step, LAS has received a message from all its neighborhood, then it starts to send its implicants to ES. When CCS receives the request from LAS, it does not answer and sends its implicant $\neg valCC$ to HPS. Simultaneously, ES gets $\neg laAprvl$ from LAS and HPS gets $\neg valCC$ from CCS. At this step, ES did not received any message from any of its neighbors, unlike HPS, which can send to ES the implicant $\neg bkAprvl \wedge \neg valCC$ built from the 4th implicant of its theory and $\neg valCC$. At this step, ES received a message from its neighbors. It builds $\neg okLAS \wedge \neg bkAprvl \wedge hPurch \wedge \neg valCC$ from its theory and its received implicants. ES removes $\neg bkAprvl$ and $\neg valCC$ which are shared variables only occurring in the subtree rooted in ES and sends $\neg okLAS \wedge hPurch$ to OVS. At this step OVS gets its first diagnosis.

5 Related Works

Our approach has been preceded by many pieces of work. Methods from [2] extend the ATMS principles of [7] in order to incrementally compute the set of conflicts in a distributed framework. However, those methods have still to wait for all conflicts before having a chance to get the first diagnosis. Many other works try to take advantage of the system topology, for instance by using decomposition properties of the model [5, 13]. Similarly [12] searches for diagnoses into a partitioned graph by assigning values for shared variables and maintaining a local consistency. The global diagnosis is thus distributed in all local diagnoses. This method however does not guarantee the minimality and supposes a global system description, which is not our case. In [3], local diagnoses from each agent (peer) are synchronized such that to obtain at the end a compact representation of global diagnoses. But the algorithm searches only for the set of diagnoses with minimal cardinality whereas we look for the set of all minimal diagnoses for subset inclusion. In [15], agents update their sets of local diagnoses in order to be consistent with a global one. However, the algorithm cannot guarantee that any combination of agents local minimal diagnoses is also a global minimal diagnosis. In [4, 10], a supervisor, that knows the global communication architecture between peers, coordinates the diagnosis task by dialoguing with local diagnosers. Thus the computation of global diagnoses from local ones is centralized. Due to the privacy constraint, in our framework a peer just knows its neighbors, no peer knows the network architecture and computation of global diagnoses is distributed, no peer playing a special role.

6 Conclusion

We proposed a distributed algorithm to compute the minimal diagnoses of a distributed Peer-to-Peer setting in an incremental way, with the help of a distributed cover tree of the acquaintance graph of the peers. Our algorithm takes advantage of the DNF representation of the local theories of the peers in order to compute global diagnoses without needs to get conflicts first. However, one has to notice that, in practice, peers do not have to rewrite their local theories in DNF. They may compute answers to requests “on the fly” and thus allow

our algorithm to work on CNF encoding of the peers. Developing this technique will be our next investigation, along with experimental testing and scaling-up study of our algorithm on real examples, such as conversational web services. We can already state the many advantages offered by our approach: we ensure privacy issues, in particular no peer has the global knowledge; we never compute the set of conflicts before computing the diagnoses; we take advantage of the natural structure of the network, which can be generally decomposed such as to obtain a small number of shared variables; we take advantage of the distributed cpu power of the whole network; lastly, we restrict the vocabulary of diagnoses as soon as possible. Future lines of research will include the study of dynamic Peer-to-Peer systems.

References

1. P. Adjiman, P. Chatalic, M.-C Rousset, and Laurent Simon. Distributed reasoning in a peer-to-peer setting. *IJCAI'2005*, 2005.
2. C. Beckstein, R. Fuhge, and G. Kraetzschmar. Supporting assumption-based reasoning in a distributed environment. *Workshop on Distributed Artificial Intelligence*, 1993.
3. J. Biteus, E. Frisk, and M. Nyberg. Distributed diagnosis by using a condensed local representation of the global diagnoses with minimal cardinality. *DX-06*, 2006.
4. L. Console, C. Picardi, and D. Theseider Dupré. A framework for decentralized qualitative model-based diagnosis. *IJCAI'2007*, 2007.
5. A. Darwiche. Model-based diagnosis using structured system descriptions. *Journal of AI Research*, 8:165–222, 1998.
6. A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of AI Research*, 17:229–264, 2002.
7. J. de Kleer. An assumption-based tms. *Artificial Intelligence*, 28:127–162, March 1986.
8. J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and sytems. *Artificial Intelligence*, 56:197–222, August 1992.
9. A. Halevy, Z. Ives, and I. Tatarinov. Schema mediation in peer data management systems. *In: ICDE'O3*, pages 505–516, March 2003.
10. M. Kalech and A. Gal Kaminka. On the design of social diagnosis algorithms for multi-agent teams. *IJCAI'2003*, 2003.
11. J.de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, April 1987.
12. J. Kurien, X. Koutsoukos, and F. Zhao. Distributed diagnosis of networked, embedded systems. *DX-02*, 2002.
13. G. Provan. A model-based diagnosis framework for distributed systems. *DX-02*, 2002.
14. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–96, April 1987.
15. N. Roos, A. ten Teije, and C. Witteveen. A protocol for multi agent diagnosis with spatially distributed knowledge. *AAMAS 2003*, 2003.
16. L. Simon and A. del Val. Efficient consequence finding. *IJCAI'01*, 2001.
17. I. Stoica, R. Morris, D. Karger, M.F. Kaasshoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM 2001*, 2001.