A First Approach to the Automatic Generation of Service Graphs for Building Trust

Walter M. Grandinetti[1]
wmg@cs.uns.edu.ar

Carlos I. Chesñevar[2]
cic@eps.udl.es

[1]Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur – Alem 1253 – B8000CPB Bahía Blanca (Argentina)

[2]Grupo de Inteligencia Artificial – Departament d'Informàtica i Enginyeria Industrial
Universitat de Lleida – C/Jaume II, 69 – 25001 Lleida, Catalunya (Spain)

Abstract

In recent years, web services have turned out to be an emerging feature that is transforming how the web is conceptualized. Rather than considering the Web as a huge collection of static pages, for many purposes the WWW can be better understood as a collection of *entities* that provide and use services. In this setting, graph-based representations for modelling trustworthiness in a provider-consumer framework for agents have proven to be an attractive approach. However, computing and mantaining the underlying graph may be a considerably complex task. This paper presents a first approach towards computing such service graphs automatically on the basis of so-called *concept lattices*. Our proposal is intended to enhance existing service-graph representations for modelling trust.

Key Words: Local and Social Trust, Concept Lattice, Referrals.

1 Introduction

In recent years, web services have turned out to be an emerging feature that is transforming how the web is conceptualized. Rather than considering the Web as a huge collection of static pages, for many purposes the WWW can be better understood as a collection of *entities* that provide and use services [10]. Such entities are free to decide *to whom they serve* and *from whom they consume*. Because of the freedom each entity has over the production of a particular service, the quality of a given service could be different among several possible providers, meaning implicitly that there are no guarantees about the quality of service provided by the participants. Clearly, in such a setting finding a way to detect high quality providers is particularly important.

In the context of multiagent systems there have been several approaches to find trustworthy agents [5, 6, 8]. Recent approaches take into account *local* as well as *social* evidence in order to rate or provide a reputation for the agent. *Local evidence* is gathered from direct interaction with the agent whereas *social evidence* is provided by third parties. Unlike other approaches, graph-based representation [11] for modelling trustworthiness in a provider-consumer framework focuses on two properties of trust which have not been adequately addressed before:

- Trust often builds up over interactions. That is, an agent might trust a "stranger agent" for a low-value transaction, but would only trust a known party for a high-value transaction.

- Trust often flows across service types. That is, an agent might assume that a party who is trustworthy in one kind of dealings will also be trustworthy in similar dealings.

Graph-based service representations outperform other approaches, such as vector representations with or without referrals, for modelling the above situations [10, 11]. However, graph-based service representations depend heavily on having an adequate *dependency graph* otherwise the quality of the set of providers given as a result will be affected. Hence, maintain the relations among services updated is an essential task. Although, it is not inherent of the graph-based service representation, service graph presented in [11] is same for every agent and is static meaning that once the graph is created (services, providers and relations) it remains as it is. The only tuning variable is a weight value describing how much two services are related.

Nonetheless, open systems are intrinsically dynamic meaning that providers could change the set of services they offer along time, and the set of available services would not necessarily remain fixed. Therefore, the limitations imposed over the graph-based service representation approach make it not adequate for dynamic environment. One of the difficulties of open environments is to keep track of every change ocurring within it. Although, it is unfeasible to keep track of every change of the providers and services, this accounting of changes is needless in order to maintain the quality of the service graph. At the moment of looking for a trustworthy provider only the services related with the service wanted need to be properly updated. Hence, two agents could have, at the same time, a different set of providers for a service or a different set of services. Also, it seems natural to think that the relation among services may be different for each agent, though they have the same set of services and providers. For instance, an agent could consider that for a provider to be trusted for a service such as web hosting, this provider should be a trustworthy provider of email service, whereas other agent may consider that relation worthless but a relation with a service such as virus protection really useful.

This paper outlines a first approach to extend a graph-based service representation setting for modelling trustworthiness among agents by generating automatically a *dependecy-service graph* and enabling agent to have their own tailored service graph. The computation of such a graph will be based on gathering public information from the environment. In addition to help in the generation of the initial service graph, we will show how the proposed approach can also be used to keep track of changes on the provided services, reflecting such changes in the service graph.

The rest of the paper is structured as follows. First, in Section 2 we will summarize the fundamentals of graph-based service representation. Next, in Section 3 we will present our proposal for automatic generation of service graphs on the basis of a concept lattice. Section 4 we will present the main conclusions obtained, and discuss some issues for our future research work.

2 Graph-Based Representation

The graph-based service representation [11] aims to model the relation that could be present among services and providers in a distributed environment. The main goal of this representation is to establish a partial order among services in order to consider *only trustworthy providers* to fulfill needs associated with high risk services. Should it be the case that there is no provider for such service, a *promotion* of potential providers is performed on the basis of that provider which has the highest probability to fulfill the expectations.

In graph-based service representation, every node represents a service and arrows are used to denote a relationship between two of them. Each arrow is weighted with a probability representing the expectation that having a given performance in one service could be "reproduced" in the other service. For instance, having two nodes S_1 and S_2 connected as $S_1 \overset{0.5}{\rightarrow} S_2$ denotes

that a provider performing S_1 well is likely to reproduce such performance in S_2 half of the time.

Example 1 Figure 1 shows a sample setting of services [11]. For the sake of simplicity, we omit the weight values attached to the arrows. It can be seen that each node represents a service denoting a different transaction value. The agents trusted for a node are a subset of the agents trusted for the lower node. That is, if an agent trust someone for a \$10 transaction, it will trust him for a \$1 transaction as well. Clearly there is a *total* ordering among services, as for a high value transaction it is sound to consider only those those providers who have shown their reliability or trustworthiness over time.

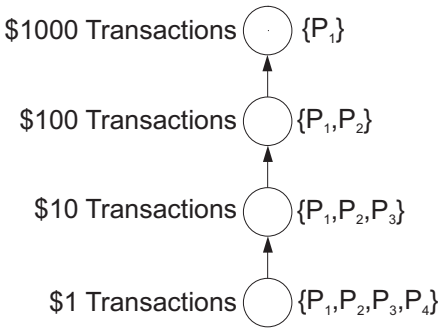


Figure 1: A totally-ordered service graph.

From the previous example, it is interesting to note that if a provider is trusted for a service S_i, the same provider is also trusted for "less risky" services related to this service, i.e., those services S_j such that $S_j \xrightarrow{*} S_i$. It should also be noted that since the relation among services is a form of dependency, we can restrict ourselves to consider only acyclic graphs.

Following the approach presented in [11], our graph-based representation includes also the notion of *promotion* which allows to use historical information about the provider. A provider will be promoted to a higher service if there is a query for such higher service and the provider has performed well in an immediately lower related service. Service providers with low ratings are thus replaced with the newly promoted providers because they have potentially more chances to perform better. In this promotion process both the *relation strength* and the *trustworthiness* of the provider are considered. The trustworthiness for a provider p at a service S_i, denoted as t_{pi}, is calculated by its ratings at S_i and the number of interactions at the service associated with the service. The strength of the relation between S_i and S_j is given by the edge weight w_{ij}. If the product of the edge weight and the trustworthiness is greater than a promotion threshold θ (i.e., $w_{ij} \times t_{pi} \geq \theta$), then the provider is considered for promotion. After promoting and testing the performance of the provider at the higher service the edge weight is updated accordingly. The rating of the provider is also updated after each interaction.

Following [11], in order to compare different representations there are two important notions: effectiveness and efficiency.

Definition 1 (Effectiveness) A representation is *effective* if it allows agents to find the desired service providers. The ratio of queries leading to useful service providers to all the generated queries is used in order to measure effectiveness. □

Definition 2 (Efficiency) A representation is *efficient* if it allows the service providers to be found with as few messages as possible. □

3 Computing Service Graphs: a first approach

Even though the graph-based representation is more expressive than the vector space model [11], its effectiveness depends on the particular setting of the services nodes. In fact, the graph layout establishes the relation between the possible services which could be distributed differently for each agent. Services are arranged within the graph according to the agent's current knowledge and needs.

In order to start up with the interaction among agents, the graph-based representation needs an initial setting of services along with the numbers that denote the relation between them and the particular information of each provider for each service. As the interaction progresses, the initial numbers need to be tailored in order to reflect the observations of the agent. However, the setup of nodes is never changed, thus its effectiveness is affected by the initial knowledge available. The main problem is that an agent developing such a graph could not have the necessary knowledge concerning the relation among all pair of services, or may be unaware of the existence of other services.

A possible solution could be requesting trustworthy agents (such as authoritative institutions or the agent's acquaintances) their own graph of services as an initial representation. However it is possible that such authoritative institutions might not exist, or the agent's acquaintances may just have a graph representing a different set of services. On the other hand, the service graph server needs to know every service and the relation among them, but these relations may not be explicitly available, even though they could be inferred from the interaction among agents. Therefore, if there is an authoritative institution which maintains the service graph, it can not depend on a manual process to keep updated the service graph but there is a natural need for an automatic process which can create and keep updated the service graph.

Moreover, in large-scale, decentralized information systems the set of available services is continually changing, growing and shrinking. In those systems, providers may have been offering a set of services for some time and afterwards they would change to a different set. For these reasons, the graph-based representation should be able to adapt itself in a way that best fits with the environment.

In order to address the problems of setting up an initial service graph and keeping it updated (i.e., reflecting possible changes in the environment), we propose to use the notion of *concept lattices* which can be generated and updated without supervision. As a guideline to generate a real life service graph, it is expected that more service providers offer easier services than harder ones [11]. Hence *the more a service is offered (number of providers), the lower its level in the service graph.*

Next we will show how an initial service graph can be obtained on the basis of knowing for each provider the services he or she is offering and those he or she is consuming. Later we will discuss how the related issue of gathering this information can be solved. First, we will introduce an example which shows how such a graph service could be intuitively generated.

Example 2 Consider an environment consisting of four providers $(P_1, \ldots, P_4)$ and six services $(S_1, \ldots, S_6)$ where each provider is offering any number of services. This information could be placed in a cross table as it is shown in Table 1.

It can be noted that services S_1 and S_6 are more offered than other services, and thus they should be at the lower levels of the graph (as they are probably the easier services to provide). Each of the other services are offered by a different provider, so that we can assume that there is no relation among them or if such a relation exists, it is too weak to be considered. Without any previous knowledge, it can be assumed that there is a relationship between each of the

	P_1	P_2	P_3	P_4
S_1	×			×
S_2		×		
S_3			×	
S_4				×
S_5	×			
S_6		×	×	×

Table 1: Services-Providers Cross Table

services S_2, S_3, S_4 and the service S_6 as well as between each of the services S_4 and S_5 and the service S_1. This inference is based on the assumption that if the services are offered by the same provider, they may be related. A graphical representation of this situation is shown in figure 2.

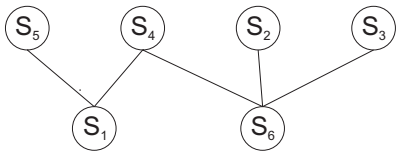


Figure 2: Intuitive Service Graph

It could be noticed that, although S_1 and S_6 are both offered by P_4, they are not related according to figure 2 since the relationship established among nodes captures some kind of "if-then implication" between services. For instance, $S_2 \rightarrow S_6$ and $S_5 \rightarrow S_1$ since every provider of service S_2 $\{P_2\}$ is present in the set of providers of service S_6 $\{P_2, P_3, P_4\}$ and the providers of service S_5 are a subset of the providers of service S_1. But neither $S_6 \rightarrow S_1$ nor $S_1 \rightarrow S_6$ since $\{P_2, P_3, P_4\} \not\subseteq \{P_1, P_4\}$ and $\{P_1, P_4\} \not\subseteq \{P_2, P_3, P_4\}$.

3.1 Formal Context of Services-Providers

In this section we will introduce some basic notions of Formal Concept Analysis (FCA), which will be used for our proposal. A more formal approach can be found in [1, 2, 9]. FCA has been developed as a mathematical model for concepts and it has been successfully applied to data analysis, information retrieval and knowledge discovery. The notion of concept consists of two parts: *a*) An *extension* consisting of all objects belonging to the concept, *b*) An *intension* consisting of all attributes common to all the objects belonging the same concept. FCA can be understood as a *conceptual clustering method*, which clusters both objects and their descriptions.

Definition 3 (Formal Context, Wille 99) A *formal context* is a triple (G, M, I) where G and M are sets and I is a binary relation between G and M. The elements of G are called *objects*, and the elements of M are called *attributes*. For any $g \in G$ and $m \in M$, gIm denotes that there exists a relationship between g and m, also noted as $(g, m) \in I$. □

The central notion of FCA is a duality called a "Galois connection" [4]. This duality relating two kind of items implies that *if one makes the sets of one type larger, they correspond to smaller sets of the other type, and viceversa.*

Definition 4 (Wille 99) Let (G, M, I) be a formal context with $X \subseteq G$ and $Y \subseteq M$, the following mappings can be established:

$$s : \mathcal{P}(G) \mapsto \mathcal{P}(M) \qquad s(X) := \{m \in M \mid (g, m) \in I \text{ for all } g \in X\} \tag{1}$$

$$t : \mathcal{P}(M) \mapsto \mathcal{P}(G) \qquad t(Y) := \{g \in G \mid (g, m) \in I \text{ for all } m \in Y\} \tag{2}$$

□

Definition 5 (Formal Concept, Wille 99) Given a formal context (G, M, I), a *formal concept* in (G, M, I) is a pair (X, Y) with $X \subseteq G$ and $X \subseteq M$, $s(X) = Y$, and $t(Y) = X$. X is called the *extension* and Y is called the *intension* of the concept. □

Definition 6 (Concept Lattice, Wille 99) The *concept lattice* of a formal context (G, M, I) is the set of all formal concepts of (G, M, I), together with the partial order

$$(A_1, B_1) \geq (A_2, B_2) \Leftrightarrow A1 \supseteq A2 \quad (\Leftrightarrow B1 \subseteq B2).$$

The concept lattice is denoted by $\mathcal{B}(G, M, I)$. □

It is interesting to note how the above ideas can be recast into the service-providers problem: in Example 2 two sets can be identified (services and providers) as well as a relationship between them. Moreover, the Galois connection is also verified between these sets, i.e., the larger a set of one type, the smaller the related set of the other type. For instance, let us take the set $\{S_6\}$, which is related with the set of providers which offers this service, i.e., $\{P_2, P_3, P_4\}$. Now, suppose that the set of services grows by adding a new service S_1. Consequently, the set $\{S_1, S_6\}$ is now related with the set $\{P_4\}$ as this is the only provider offering *both* services. On the other hand, let us take the set $\{P_1\}$ involving one provider, related with the set of services which he offers, namely, $\{S_1, S_5\}$. Augmenting this set of providers (for instance to $\{P_1, P_4\}$) would imply a shrinking in the set of services (in this case, $\{S_1\}$).

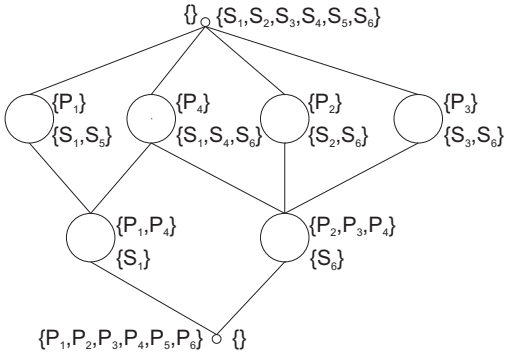


Figure 3: Concept Lattice for cross table 1.

From our previous remarks it follows that it is possible to establish a formal context based on these sets and their relationship and visualize it using the concept lattice, in a so-called *line diagram*. Figure 3 shows the concept lattice associated with the relation shown in table 1. The similarity of the graph layout in figures 2 and 3 is clear. The main difference between the service graph and the concept lattice is that each node in a service graph refers to a particular service whereas each node in a concept lattice refers to a concept (i.e., a relationship between a set of services and a set of providers). However, there is another way of labelling the nodes within a concept lattice known as *minimal labelling* where every service (object) and every provider (attribute) appears *just once on the whole lattice*. This labelling is depicted in figure 4. In this particular case, it can be observed that both graphs match. This situation, however, will not always be the case, as we will see in the following section.

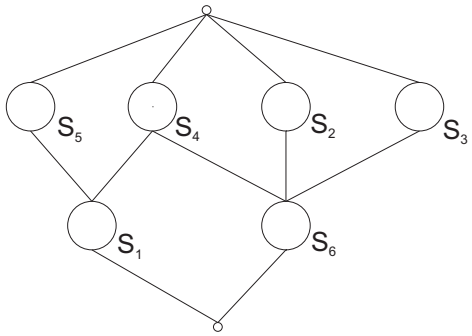


Figure 4: Concept Lattice with Minimal Labelling.

3.2 Changes in the environment

As we have shown in the previous section, a service graph can be generated automatically (without supervision) from a cross table. However, even though such a service graph could be stored either by an authoritative entity or by each single agent, it would need to be updated in order to reflect changes in the environment. As we pointed out at the beginning of this section, the lower the service (node) is, the bigger the numbers of providers offering it. Thus, if either new providers or services are added or some providers change the services they offer, the service graph should reflect those changes in order to keep accurate the information about the providers that offer each service. Let us consider again the previous example:

Example 3 Suppose that provider P_1 "shrinks" his or her business by diminishing the services offered, just providing service S_1, whereas provider P_4 "expands" his business with a new service, namely S_5, and provider P_3 changes his business direction with an exchange of the services offering S_5 instead of S_6. So, service S_6 suffers a diminishing on the number of providers while the offer of service S_5 is increased as it is shown on Table 2.

Although the initial service graph may still be useful, there are certain "inconsistencies" within

	P_1	P_2	P_3	P_4
S_1	×			×
S_2		×		
S_3			×	
S_4				×
S_5			×	×
S_6		×		×

Table 2: Changes on the Services-Providers Cross Table

it, such as:

- The relation between service S_1 and service S_5 is no longer valid because none of the services implies the other, i.e., neither S_1 implies S_5 (since there is at least a provider (P_3) that offers S_5 without offering S_1) nor S_5 implies S_1 (since there is also at least a provider (P_1) that offers S_1 without offering S_5).
- There is a *new* relation between service S_4 and service S_5 because every provider of S_4 also provides S_5.

- There is a change in the way the set of services is related toward service S_3. This service (S_3) is only provided by P_3 and this provider changed the services offered, so that this also changes the relations for service S_3. In other words, there is no longer a relationship between S_3 and S_6 and now there is a new relation between S_3 and S_5 (because every provider of service S_3 also offers services S_5 according to the cross table).

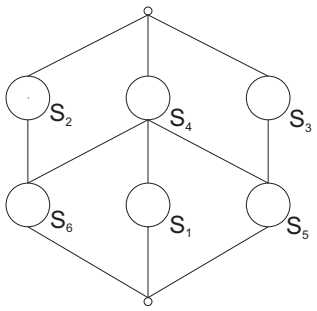


Figure 5: Updated Service Graph based on Concept Lattices.

The new service graph is depicted in figure 5. Interestingly, if the service graph were not properly updated, it could have happened that for future requests:

- Providers of service S_5 should also offer service S_1. Hence, the approach of relying on a non-updatable service graph would fail for obtaining a high quality provider, just because this provider is not offering another service which is so far unrelated.
- Providers for service S_4 were only considered from providers of services S_1 and S_6 preventing us from discovering new high quality providers that offer services S_4 and S_5.
- Providers of service S_3 were required to offer service S_6 which is unrelated to service S_3. Moreover, providers of service S_5 would not be considered for promotion. □

3.3 Adapting the Concept Lattice

As we have previously discussed, concept lattices appear to be an appropiate structure to automatically generate service graphs, even though they may not perfectly match with the definition of a service graph which states that every node corresponds to a service and every arrow linking two nodes corresponds to a relation or dependency between these nodes. There are two issues that needs to be addressed before a concept lattice could be used as a service graph: *a*) consider nodes within the concept lattice that match with *no service*, or *b*) consider nodes within the concept lattice that match with *more than a service*. The former issue happens when the lattice needs more nodes than services (objects) available to model the cross table as it can be appreciated in the following example.

Example 4 Let us take our previous example and make some modifications to the original cross table. The figure 6 shows the cross table along with its concept lattice.

From that figure, we can see that in addition to bottom and top concept lattice nodes there is a special node which does not have a service label. The purpose of this node is to represent a pseudo-service S_3–S_6 which involves providers of *both* services (P_3 and P_4). Since the service graph does not model these pseudo-services, there are two options:

	P_1	P_2	P_3	P_4
S_1			×	
S_2		×		
S_3	×		×	×
S_4				×
S_5	×		×	
S_6		×	×	×

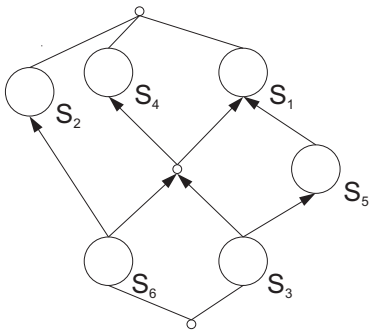


Figure 6: Pseudo-Services Cross Table

1. Extend the service graph to consider such special nodes.
2. Get rid of these special nodes transforming the lattice into a graph keeping all the information retrieved so far.

The first option requires to modify the algorithms proposed in [11] to propagate every provider promoted to service either S_3 or S_6 also to the pseudo-service just created. The second option requires a transformation of the lattice according to the following pseudocode:

For every pseudo-node p
  For every *outgoing* arrow j of p
    For every *incoming* arrow i to p
      Create an arrow from i to j
      Eliminate arrows (i, p) and (p, j)

The resulting final service graph is depicted in figure 7.

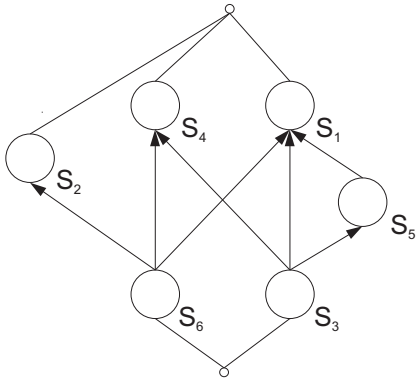
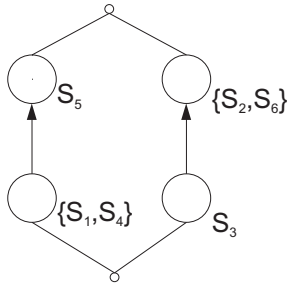


Figure 7: Service Graph without pseudo-services.

The last issue that needs to be considered is what happens when a concept lattice node is labelled with *more than a service*, as is the case shown in the following example.

Example 5 Let us consider again a modification of the original cross table as depicted in figure 8 along with its concept lattice. It can be seen that there are two nodes which are labelled with more than a service, namely, $\{S_1, S_4\}$ and $\{S_2, S_6\}$.

	P_1	P_2	P_3	P_4
S_1	×			×
S_2			×	
S_3		×	×	
S_4	×			×
S_5	×			
S_6			×	



□

Figure 8: Multiple-Services Cross Table

In this case the solution is even simpler than the solution of the previous issue, as it suffices to duplicate the node in as many nodes as the number of services labeled in that node. Note that we also have to duplicate all the inward and outward arrows for every multi-service labeled node. The transformed service graph is shown in figure 9. Similarly, it is also possible to extend the service graph to handle those special nodes, keeping track of the proper service for every provider of a multi-service labeled node.

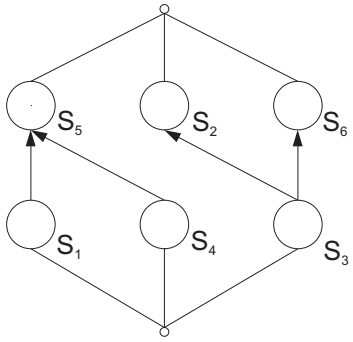


Figure 9: Service Graph without merged services.

3.4 Building up the cross table

Finally, we will address the problem of setting and updating the cross table. It should be clear by now that the service graph based on concept lattices depends on a cross table defining the relationship between services (objects) and providers (attributes). Hence, the more accurate the information filled in the cross table is, the better the service graph quality.

As we have mentioned before, a cross table is used to store the relation between services and providers. Thus, it is important to distinguish this relationship between services and providers from the relation either between services or between providers. This relation between services and providers is established when a provider offers a particular service. Also, a relation could be established when a provider needs or uses a particular service and it can not longer be able to distiguish between services offered and services needed. This latter case will not be analyzed in this paper.

Let us explain how the services offered by a provider could be obtained. This piece of information should be readily available because every the provider is interested to have his

list of services offered as widespread as possible in order to catch the attention of potential consumers. It is possible, though, that some providers do not fall in this category because the service they offer needs to be kept secret. For instance, most of illegal services are secretly offered thus we will fail to gather such information unless provided to us by some other agent.

Unlike services offered by each provider, consumed services are harder to discover because this is sensible information about possible weaknesses of the provider. However, this is not always the case, as we can identify two sources where this information could be collected from, namely:

1. **The provider itself**: In certain domains the providers themselves post their needs to put up for auction.

2. **The providers of the provider**: Usually, service providers are ranked according to the number of successful transactions done by themselves and there is a public available list of their customers along with the feedback provided by the customer, as well as the information about the transaction itself (e.g. eBay). Then, providers transactions can be searched in order to gather all the information about the consumed services for each provider.

To sum up, the cross table could be filled with the services offered and consumed by each provider. However, for sake of simplicity, we will restrict ourselves to the case when the relation between service and provider corresponds to a service that the provider offers. The information about services offered is extracted from the services explicitly offered by each provider. The information about services consumed could be gathered from the provider itself and from other providers whom the former have had a committed transaction.

4 Conclusions and Future Work

In this paper we have presented a first approach towards computing service graphs automatically on the basis of so-called *concept lattices*. Our proposal is intended to enhance existing service-graph representations for modelling trust. As we have discussed along this paper, the basic approach of graph-based service representation has some limitations that could be problematic in some large scale or open environments (such as fixing the offered services or the services that each provider offers). Our proposal aims to provide a possible solution to such limitations incorporating an automatic tool in order to generate the service graph. Our goal is twofold:

1. *Helping to develop an initial graph without supervision.* This feature is especially interesting not only for those agents with scarce knowledge of the domain but also for those experimented agents, because it provides some insights from the environment that may be not known. It should also be taken into account that it is not feasible to manually detect all the dependencies between services if there is a great number or services. Hence an automatic tool that provides an initial graph could also be useful even for the more experimented agents, though, they use a manual approach.

2. *Keeping the service graph automatically updated.* Since in dynamic environments providers come and go, new services are created and old services are no longer available, providers change the services they offer, etc., it is mandatory to keep up the track of those changes accordingly in the service graph as well. Otherwise, the efficiency and effectiveness of the approach would be diminished.

We think that graph-based service representation provides many opportunities for further research. For instance, the promotion process in our analysis is based on a *single* related service. However, sometimes a service could be composed of several smaller services. Then, the trustworthiness of providers considered for promotion should consist of the composed trustworthiness at such smaller services. This problem was first detected by Pinar and Singh [11] and by Sabater and Sierra [7]. Sometimes, it would also be helpful to model strict dependency among several services in order to require that the promotion process only consider providers performing well at every dependant service.

Currently our work is focused on integrating *external knowledge about relations among services* in order to enrich the service graph automatically generated. This information could be requested to the external agent in form of dependency rules and integrated into the cross table before generating the service graph. It should be interesting to integrate into the service graph information about services needed by an agent in addition to the set of offered services. This could provide some new insights about dependencies among providers and it could also help to relieve some problems such as *biased referrals*, that is referrals that provide a misleading advise according to their own benefit. Research in this direction is also being pursued.

Acknowledgments

We wish to thanks the reviewers for their comments. This research was funded by Agencia Nacional de Promoción Científica y Tecnológica (PICT 13096, PICT 15043, PAV 076), by CONICET (Argentina), by projects TIC2003-00950 and TIN2004-07933-C03-03 (MCyT, Spain) and by Ramón y Cajal Program (MCyT, Spain).

References

[1] B. Ganter and Rudolf Wille, *Formal concept analysis: Mathematical foundations.*, Heidelberg, Springer, 1999.

[2] Joachim Hereth, Gerd Stumme, Rudolf Wille, and Uta Wille, *Conceptual knowledge discovery and data analysis*, International Conference on Conceptual Structures, 2000, pp. 421–437.

[3] Robert Meersman and Zahir Tari (eds.), *On the move to meaningful internet systems 2004: Coopis, doa, and odbase, otm confederated international conferences, agia napa, cyprus, october 25-29, 2004, proceedings, part i*, Lecture Notes in Computer Science, vol. 3290, Springer, 2004.

[4] Uta Priss, *Introduction to and overview of formal concept analysis*, to appear in Annual Review of Information Science and Technology, Vol. 40, 2004.

[5] I. Rahwan, S. Ramchurn, N. Jennings, P. McBurney, S. Parsons, and L. Sonenberg, *Argumentation-based negotiation*, Knowl. Eng. Rev. **18** (2003), no. 4, 343–375.

[6] Jordi Sabater, *Evaluating the regret system.*, Applied Artificial Intelligence **18** (2004), no. 9-10, 797–813.

[7] Jordi Sabater and Carles Sierra, *REGRET: reputation in gregarious societies*, Proceedings of the Fifth International Conference on Autonomous Agents (Montreal, Canada) (Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, eds.), ACM Press, 2001, pp. 194–195.

[8] Carles Sierra and Jordi Sabater, *Reputation and social network analysis in multi-agent systems.*, Intl. First AAMAS Conference 2002, 2002, pp. 475–482.

[9] Gerd Stumme, *Formal concept analysis on its way from mathematics to computer science*, Conceptual Structures: Integration and Interfaces (U. Priss, D. Corbett, and G. Angelova, eds.), Springer, 2002, Proc. ICCS 2002, LNAI 2393, Springer, Heidelberg 2002, pp. 2–19.

[10] Pinar Yolum and Munindar Singh, *Engineering self-organizing referral networks for trustworthy service selection*, IEEE Transactions on Systems, Man, and Cybernetics (2004), 195–211.

[11] Pinar Yolum and Munindar P. Singh, *Service graphs for building trust.*, in Meersman and Tari [3], pp. 509–525.