First IJCAI International Workshop on Graph Structures for Knowledge Representation and Reasoning (GKR@IJCAI'09)

Madalina Croitoru, Christophe Gonzales, Jérôme Lang, Boris Motik, Marie-Laure Mugnier

▶ **To cite this version:**

Madalina Croitoru, Christophe Gonzales, Jérôme Lang, Boris Motik, Marie-Laure Mugnier. First IJCAI International Workshop on Graph Structures for Knowledge Representation and Reasoning (GKR@IJCAI'09). pp.59, 2009. lirmm-00410651

HAL Id: lirmm-00410651

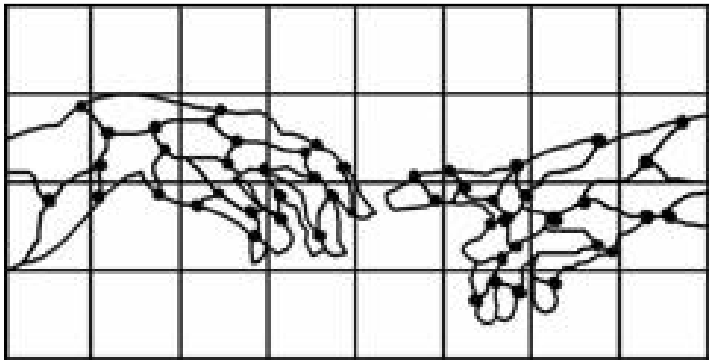
https://hal-lirmm.ccsd.cnrs.fr/lirmm-00410651

Submitted on 21 Aug 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

FIRST IJCAI WORKSHOP ON GRAPH STRUCTURES FOR KNOWLEDGE REPRESENTATION AND REASONING



Madalina Croitoru

Christophe Gonzales

Jerome Lang

Boris Motik

Marie – Laure Mugnier

FIRST IJCAI WORKSHOP ON GRAPH STRUCTURES FOR KNOWLEDGE REPRESENTATION AND REASONING

Madalina Croitoru

Christophe Gonzales

Jerome Lang

Boris Motik

Marie – Laure Mugnier

Program Committee

Jos de Bruijn, Free University of Bozen-Bolzano, Italy, debruijn@inf.unibz.it

Cornelius Croitoru, Al. I. Cuza Univ., Iasi, Romania, croitoru@info.uaic.ro

Paul Doran, ART, CS, University of Liverpool, UK, pdoran@csc.liv.ac.uk

Mathieu Daquin, KMI, Open University, UK, M.Daquin@open.ac.uk

Srinandan Dasmahapatra, ECS, Southampton, UK, sd@ecs.soton.ac.uk

Kees van Deemter, Univ. of Aberdeen, UK, k.vdeemter@abdn.ac.uk

Harry Delugach, Univ. of Huntsville, USA, delugach@cs.uah.edu

Fabien Gandon, INRIA, Sophia Antipolis, France, Fabien.Gandon@sophia.inria.fr

John Howse, Univ of Brighton, UK, John.Howse@bton.ac.uk

Robert Jaschke, University of Kassel, Germany, jaeschke@cs.uni-kassel.de

Mary Keeler, VivoMind Intelligence, Inc., USA, mkeeler@u.washington.edu

Uffe Kjrulff, Aalborg University, Denmark, uk@cs.aau.dk

Michel Leclere, LIRMM, Univ. Montpellier II, France, leclere@lirmm.fr

Guy Mineau, Universite Laval, Canada, guy.mineau@fc.ulaval.ca

Thomas Dyhre Nielsen, Aalborg University, Denmark, tdn@cs.aau.dk

Alun Preece, School of Computer Science, Cardiff University, UK, A.D.Preece@cs.cardiff.ac.uk

Juan Antonio Rodriguez Aguilar, IIIA, Barcelona, Spain, jar@iiia.csic.es

Sebastian Rudolph, AIFB, Karlsruhe, Germany, sru@aifb.uni-karlsruhe.de

Anne Schlicht, WIN, Mannheim, Germany, anne@informatik.uni-mannheim.de

Eric Salvat, I.M.E.R.I.R., Perpignan, France, eric.salvat@imerir.com

Dan Tecuci, University of Texas, USA, tecuci@cs.utexas.edu

Rallou Thomopoulos, INRA, UMR IATE, Montpellier, France, rallou@supagro.inra.fr

Nic Wilson, Cork Constraint Computation Centre, Ireland, n.wilson@4c.ucc.ie

The development of effective techniques for knowledge representation and reasoning (KRR) is a crucial aspect of successful intelligent systems. Different representation paradigms, as well as their use in dedicated reasoning systems, have been extensively studied in the past. Nevertheless, new challenges, problems, and issues have emerged in the context of knowledge representation in Artificial Intelligence (AI), involving the logical manipulation of increasingly large information sets (see for example Semantic Web, BioInformatics and so on). Improvements in storage capacity and performance of computing infrastructure have also affected the nature of KRR systems, shifting their focus towards representational power and execution performance. Therefore, KRR research is faced with a challenge of developing knowledge representation structures optimized for large scale reasoning. This new generation of KRR systems includes graph-based knowledge representation formalisms such as Bayesian Networks (BNs), Semantic Networks (SNs), Conceptual Graphs (CGs), Formal Concept Analysis (FCA), CP-nets, GAI-nets, all of which have been successfully used in a number of applications. The goal of this workshop is to bring together the researchers involved in the development and application of graph-based knowledge representation formalisms and reasoning techniques.

GKR 2009 @ IJCAI Organisers

Using Maximal Join for Information Fusion

Claire LAUDY
Thales Research & Technology
Palaiseau, France
claire.laudy@thalesgroup.com

Jean-Gabriel GANASCIA
Computer Science Laboratory of Paris 6
University of Paris 6, Paris, France
jean-gabriel.ganascia@lip6.fr

Abstract

Information fusion is a very active research domain. A lot of studies exist dealing with information fusion at a low level of semantics. Our claim is that information should be fused at a high level of semantics and using a symbolic representation. Previously, we intuitively presented a framework for high-level symbolic fusion. Our approach relies on the use of the conceptual graphs model. Domain knowledge is a major point of the fusion process. The use of conceptual graphs for knowledge representation fusion eases the process of expressing domain knowledge and fusion heuristics. In this paper, we formalize our approach. In particular, we detail and formalize the introduction of domain knowledge inside the fusion process. We validate our approach within the context of a TV program recommendation system.

1 Introduction

The first step of a decision-making process is to gather the relevant pieces of information and to combine and fuse them in order to have a global representation of the external world. The information sources may be redundant and express different points of view. Furthermore, the information can be acquired through different electronic sensors or may even come from humans and convey a lot of implicit knowledge. Combining all the information items distributed across the different sources in order to build a coherent and accurate global view is a very difficult and time consuming task.

Information fusion is defined as the use of techniques that combine information items comming from different sources in order to merge them. Information fusion is a very active research domain (see www.isif.org). Studies exist that deal with different types of information. A lot of studies deal with the fusion of data expressed at a low level of semantics, but our claim is that information should be fused at a high level of semantics and using a symbolic representation. As the information items coming from the different sources may depict different points of view of a situation or use different levels of details, their fusion may lead to conflict between the information that should be fused. Domain knowledge is then used to solve these conflicts and fuse properly the observations. Using a high level of semantics and a symbolic representation of the domain knowledge and of the fusion heuristics will allow domain experts, with no particular skills in knowledge representation or mathematics, to parameterize the system with their own preferences (i.e. their own heuristics) derived from their knowledge of the domain.

In [Laudy and Ganascia, 2008] we presented a framework for high-level symbolic fusion. Our approach relies on the use of the conceptual graphs model ([Sowa, 1984]). Using conceptual graphs eases the expression of the fusion heuristics and the domain knowledge. We also take the advantage of the operators that are defined among the conceptual graphs structures. The maximal join operation of two conceptual graphs is of major interest. Given two graphs that share compatible subgraphs, the maximal join attempts to build a new graph in which the two initial graphs are fused, according to their compatible subgraphs. We introduce the use of fusion heuristics inside the maximal join in order to take into account the domain knowledge and the user preferences that are necessary to achieve a good quality of fusion. We call them Fusion Strategies. Fusion strategies are rules encoding domain knowledge, used in order to extend the notion of compatibility between the concepts of two graphs.

We conducted a preliminary study within the context of a recommendation system for intelligent numerical television. The recommendation system analyzes TV program descriptions and decides to recommend the programs or not to a specific user. We used our fusion platform in order to obtain precise and reliable TV program descriptions, both regarding the schedule and the content description of the program.

These preliminary studies gave the general idea of our approach and introduced it intuitively. The aim of the present paper is to formalize the approach. In particular, as said before, the use of domain knowledge is a major point of the fusion process, so we detail the introduction of domain knowledge inside the maximal join operation.

This paper is organized as follows. We briefly review our approach in Section 2. In Section 3, we detail the definition of the fusion strategies and the constraints they should respect. Section 4 presents our extension of the maximal join with the use of fusion strategies. We illustrate our approach within the TV program recommandation system in Section 5. In Section 6, we compare our approach to related works. We finally

conclude and present future work.

2 Conceptual Graphs for Symbolic Fusion

In [Laudy and Ganascia, 2008], we presented a framework for high-level information fusion that uses the conceptual graphs formalism. Using a generic and expressiv formalism such as Conceptual Graphs to represent the information items that have to be fused allows us to use our fusion platform within different application domains. The model of the domain may be restricted to a few concepts and relations (as for the decsription of TV programs), or may be much more complex, with a lot of concepts linksd with each other through many different relationships (as for crisis management, see [Laudy and Goujon, 2009].

The Conceptual Graphs model was developed by JF Sowa in [Sowa, 1984]. In this work, we consider a subset of the conceptual graphs named *Simple Graphs* ([Chein and Mugnier, 2008]). A Simple Graph G is defined by a set of concept nodes C_G, a set of relation nodes R_G, a set of edges E_G and a naming function l_G. The labels of the concept nodes are defined by a concepyual type and an individual marker. The relation nodes are labeled with a conceptual relation type. The edges incident to a relation node r are ordered and labeled with (r, i, c), with $1 \leq i \leq n$, n being the arity of r and c the concept linked to r through this edge. A support is defined that contains all the conceptual types. In this work, we consider that the support is a lattice.

The observations that are acquired from the different sensors are stored as conceptual graphs. To fuse them, we use the maximal join operation defined by Sowa. As shown in Figure 1, the maximal join allows to fuse two compatible subgraphs of two conceptual graphs. Graph G3 is the result of the fusion of G1 and G2 using the maximal join operation.

According to Sowa ([Sowa, 1984] pp. 101-103), to be joined maximally, two graphs G1 and G2 must share compatible subgraphs. In other words, they must have a common generalization G0 with compatible projections $P_1 : G0 \rightarrow G1$ and $P_2 : G0 \rightarrow G2$. P_1 and P_2 are compatible, if for each concept c of G0:

- $P_1(c)$ and $P_2(c)$ have a common sub-type different from $\perp$,
- the individual markers of $P_1(c)$ and $P_2(c)$ are conform to their most general common sub-type,
- the individual markers of $P_1(c)$ and $P_2(c)$ are either equals, or one of them is undefined.

The maximal join of G1 and G2 is built by joining them on the maximally extended compatible projection.

The maximal join is a fusion operator. Furthermore, it gives several results, which depict the different ways of combining the information, that is to say, the different fusion hypothesis. However, as stated in [Laudy and Ganascia, 2008], using the maximal join only is not sufficient in order to fuse information coming from real systems. Real data is noisy and knowledge about the domain is often needed in order to fuse two different but compatible values into a single one. Observations such as a person named "J. Smith" and a one named

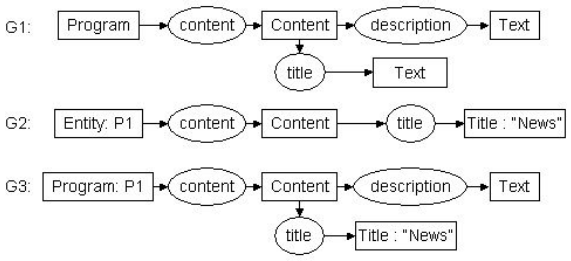


Figure 1: Example of a maximal join operation

"Mr. John Smith" are not equals, but our background knowledge let us believe that the two observations rely to the same person.

Using domain knowledge, the notion of compatibility between concepts is extended from compatibility of conceptual types only to compatibility of individual markers. We use fusion strategies, which are rules encoding domain knowledge and fusion heuristics. The definition of the fusion strategies are divided into two parts :

- The definition of the compatibility conditions between two concepts, and
- the definition of how to process the fused value of two concepts.

3 Fusion Strategies

Formally, the fusion strategies are expressed as rules that emcompass two functions: a compatibility testing function, and a fusion function. These two functions are expressed by domain experts and contain the domain knowledge necessary to achieve a good level of fusion.

Let $\mathcal{S}$ be a lattice of conceptual types and l be a set of individual markers. E is the set of concept nodes defined on $\mathcal{S} \times l$, G_1 and G_2 are two conceptual graphs defined on $\mathcal{S}$ and c_1 and c_2 are concepts belonging to E such that $c_1 \in G_1$ and $c_2 \in G_2$.

A fusion strategy $strategy_{fusion}$ is defined as follows:

$$strategy_{fusion} = \begin{array}{ll} if & f_{comp}(c_1, c_2) \\ then & f_{fusion}(c_1, c_2) \\ else & \{c_1, c_2\} \end{array}$$

where $f_{comp} : E \times E \rightarrow \{true, false\}$ is a function testing the compatibility of two concept nodes,
and $f_{fusion} : E \times E \rightarrow E$ is a fusion function upon the concepts nodes of the graphs.

The fusion strategies applied on two concept nodes result either in a fused concept node if the initial nodes are compatible, or in the initial nodes themselves if they are incompatible.

3.1 Definition of the Compatibility Function

The compatibility function can be defined either regarding the distance that exists between two values or the similarity between them. These two measures (distance and similarity) are defined by domain experts, given the requirements of the application.

If the compatibility is defined regarding the similarity of the two concepts, the similarity measure is compared to a threshold defined by domain experts. The compatibility function f_{comp} is then defined as follows :

$$f_{comp}(c_1, c_2) = sim(c_1, c_2) \geq threshold_{sim}$$

If the compatibility is processed regarding the distance that exists between the concepts, the distance measure is also compared to a threshold defined by domain experts and the compatibility function is given as follows:

$$f_{comp}(c_1, c_2) = dist(c_1, c_2) \leq threshold_{dist}$$

3.2 Definition of the Fusion Function

The fusion function allows, for any couple of concept nodes, to process, if it exists, the concept node resulting from the fusion of the two initial nodes:

$$f_{fusion}(c_1, c_2) = c$$

where $c \in E$ is the concept that results from the fusion of c_1 and c_2.

It is sometimes necessary to know the context of two observations in order to determine whether they are compatible or not as well as to determine the result of their fusion. In these cases, the compatibility and fusion functions defined by the domain experts, take into account the neighboring concepts and relations of the concept nodes to be processed. Two types of context can be considered :

- The role of the processed concepts, which is defined by the neighboring relations that have this concept as target;
- The whole observation, that is to say the whole graphs that have to be fused.

In such cases, we add respectively, the role (i.e. the set of incomming icident relations) or the whole initial graphs in the signature of the fusion function. We used both types of context withing the application on TV programs description fusion that we detail hereafter.

4 Maximal Join given a Fusion Strategy

The fusion strategies are used to extend the maximal join operation that was initially defined by Sowa. Therefore, the building of the set of the fusion hypothesis of two graphs is still directed by the search of compatible projections. The notion of compatibility between two concept nodes is extended, as details hereafter. Furthermore and the construction of the joint (i.e. fused) concepts is also modified, allowing to use heuristics in order to choose the concept values.

In this section, we explain how we propose to use domain knowledge and fusion heuristics inside the maximal join operation. We call this process "*maximal join given a fusion strategy*".

In the remaining of this section, we use the following notations:

- E is the set of all the concept nodes which types are defined on a support $\mathcal{S}$,
- H, G_1 and G_2 are conceptual graphs defined on $\mathcal{S}$,
- f_{comp} is a compatibility function that is defined on $E \times E \rightarrow \{true, false\}$,
- f_{fusion} is a fusion function defined on $E \times E \rightarrow E$ and
- a fusion strategy $strategy_{fusion}$ that encompasses f_{comp} and f_{fusion}

Definition: Compatible concepts given a fusion strategy Two concepts $c_1 = [t_1 : m_1]$ and $c_2 = [t_2 : m_2]$ are compatible given $strategy_{fusion}$ if the following conditions are verified:

- t_1 and t_2 have a most general common sub-type t different from $\perp$,
- m_1 and m_2, are respectively the individual markers of c_1 and c_2 and conform to t,
- m_1 is undefined and $m = m_2$ <u>or</u> m_2 is undefined and $m = m_1$ <u>or</u> $m_1 = m_2 = m$ <u>or</u> $f_{comp}(c_1, c_2) = true$ *and* $f_{fusion}(c_1, c_2) = c = [t : m]$

If c_1 and c_2 are compatible, we can merge them and the resulting concept is $c = [t : m]$.

Definition: Compatible relations Two relations r_1 and r_2 are compatible if the following conditions are verified:

- They share the same conceptual type;
- for each i such that there exist an edge incident to r_1 labeled (r_1, i, c_1), there exist an edge (r_2, i, c_2) such that c_1 and c_2 are compatible;
- for each i such that there exist an edge incident to r_2 labeled (r_2, i, c_2), there exist an edge (r_1, i, c_1) such that c_1 and c_2 are compatible;

Definition: Compatible graphs given a fusion strategy Two graphs G_1 and G_2 are compatible given $strategy_{fusion}$ if there exists an isomorphism p from G_1 to G_2 such that for each node n of G_1, n and $p(n)$ are compatible.

The fusion of compatible graphs G_1 and G_2 according to a fusion function f_{fusion} consists of replacing each concept node c of G_1 by $f_{fusion}(c, p(c))$.

Definition: Maximal join given a fusion strategy A maximal join operation between two graphs G_1 and G_2 according to a fusion function f_{fusion} is obtained by fusing two of their compatible sub-graphs SG_1 and SG_2 according to f_{fusion}. Furthermore, SG_1 and SG_2 are such that, no sub-graphs SG'_1 and SG'_2 respectively of G_1 and G_2 exist with:

- SG'_1 and SG'_2 compatible and
- SG_1 is a sub-graph of SG'_1 and SG_2 id a sub-graph of SG'_2.

When applying the maximal join given a strategy operation between two graphs, one may obtain several results. Indeed, there are as many results as the number of couples of maximal (with respect to node sets inclusion) compatible subgraphs SG_1 and SG_2 between the two initial graphs. These different results depict the different fusion hypothesis and are therefore of importance regarding the global objective of semi-supervised fusion.

5 Experimentations

5.1 Context

We applied the approach within a TV program recommendation system. Based on background information and the description of a new program, the system evaluates whether the new TV program is of interest to a specific user. The description must therefore be very detailed concerning the content of the program itself. It should also be as precise as possible concerning the broadcast times.

The recommendation system initially used the live stream of metadata associated with the video stream on the TNT (Télévision Numérique Terrestre). This stream gives descriptions of TV programs that are very precise concerning the begin and end times of programs. However, no description of the content of the program is given. In order to obtain more detailed and precise descriptions of the TV programs, the descriptions coming from the TNT are fused to ones coming from an on-line TV magazine. The descriptions contain much more details about the contents (summary of the program, category, actors, presenters etc).

5.2 Fusion Strategies

In order to measure the quality of fusion using different fusion strategies, we launched our experimentations using the fusion platform first combined with no strategy and then with three different ones. The first experiment -no fusion strategy- is equivalent to using the initial maximal join operator for information fusion. The three strategies are the following ones:

- **Strategy 1** extends dates compatibility. Two dates are compatible if the difference between the two is less than five minutes. If two dates are compatible but different, the fused date should be the earliest one if it is a "begin date" and the latest one otherwise.
- **Strategy 2** extends dates and titles compatibility. The dates compatibility is the same as for strategy 1. Two titles are compatible if one of them is contained in the other one.
- **Strategy 3** extends dates and titles compatibility. The dates compatibility is the same as for strategy 1. Two titles are compatible if the length of the common substrings exceeds a threshold.

We detail hereafter examples of compatibility and fusion functions used in the three fusion strategies. We will not recall here the whole support that we defined for TV program descriptions and that is described in [Laudy and Ganascia, 2008]. Roughly, this support contains conceptual types such as "Program", "Title", "content", ... which denote the basic attributes that one can find in any TV program description.

Title compatibility

The function described here is the one used in the strategy 3. The value of a "Title" concept node is a string that represent the title of the described program. Our similarity function relies on the total length of the common substrings between the two titles. Two individual concept nodes typed "Title" with individual markers t_1 and t_2 are compatible if and only if:

$$\frac{lengthOfCommonSubStrings(t_1, t_2)}{max(length(t_1), length(t_2))} \geq$$
$$max(length(t_1), length(t_2)) * 0,5$$

Date compatibility

Intuitively, we want to represent the fact that two dates are compatible if they differ from less than 5 minutes .

In order to manipulate the dates as numbers and ease the comparisons, we apply a simple transformation. Each date is given as the number of seconds that passed since a referring date. Two individual concept nodes typed "Date" with individual markers d_1 and d_2 are compatible if and only if:

$$|d_1 - d_2| \leq 300$$

Fusion of compatible dates

For the fusion of concept nodes of type "Title", our strategy consists in taking the longest of the two proposed titles:

$$f_{fus}([Title : t_1], [Title : t_2]) = fus_{title}(t1, t2)$$

with $fus_{title} : S \times S \rightarrow E_T$ (where E_T is the set of all the concept nodes of type "Title") is defined as follows:

$$\begin{cases} fus_{title}(t_1, t_2) = t_1 \text{ if } length(t_1) \geq length(t_2) \\ fus_{title}(t_1, t_2) = t_2 \text{ if } length(t_2) > length(t_1) \end{cases}$$

5.3 Results

We realized the experimentations using sixteen TV channels. The aim is to obtain as much TV program descriptions as possible, concerning the TV programs scheduled on a TV channel, during one day. Our experimentation protocol is the following one. We request every 5 minutes the two sources of information to give us the next program scheduled on one channel. The two provided TV program descriptions are then fused using the fusion platform combined with one of the fusion strategies.

In order to compare the results of the fusion to the programs that were really performed, we collected TV program descriptions from the INAthèque. The INA, Institut National de l'Audiovisuel, collects the descriptions of all the programs that have been broadcast on the French TV. The exact begin and end times are recorded, as well as a brief description of the contents. We processed the percentage of programs that were correctly found, according to the different strategies. By correctly found, we mean that the fused program description is well formed and the values of the individual markers are compatible with the ones of the description stored in the INAthèque.

Figure 2 depicts the model of a well formed description. We added unicity constraints on this model, in order to specify that some of the subgraphs of the model should be present atmost one time in a fused description. For instance, a well formed TV program has only one title (cf. figure 2). The same applies to begin and end times. To test whether a fused

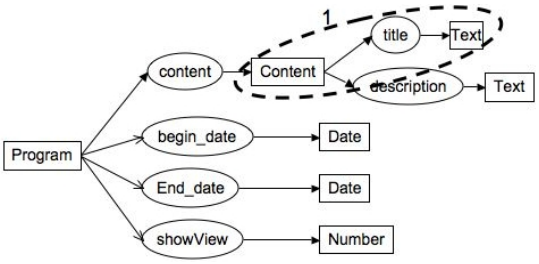


Figure 2: model for a TV program description

description F is well formed, we verify that for each subgraph M_i of the model M on wich a unicity constraint applies, there is a unique isomorphism from M_i to a subgraph F_i of F such that F_i is more specific than M_i.

Figure 3 shows the results that we obtained on a representative selection of TV channels. As expected, we can see that the fusion of observations using the maximal join operation only is not sufficient. Only the descriptions with strictly identical values are fused. There is too much noise in real data for a fusion process that does not take into account some knowledge about the domain. Therefore, we applied the three previously cited fusion strategies. The more the compatibility constraints between two values are relaxed, the better the results are. It is equivalent to inject more and more domain knowledge in the fusion process. The issue then is to find the right amount of knowledge that has to be injected.

6 Related Works

6.1 Information fusion based on graph structures

The necessity of taking into account high-level information has recently been reported by the information fusion community. Recent works such as [Buford *et al.*, 2008], [Sambhoos *et al.*, 2008] and [Laskey, 2008] insist on the importance of such information and propose new approaches for information fusion, taking into account observations provided by humans. Works such as [Matheus *et al.*, 2003] insist on the importance of using ontologies to represent knowledge. Furthermore, graph structures are often used to store information. Therefore, information fusion based on graphs structures is a major stake.

In [Rickard, 2006], information items are stored as graphs and further fused. However, the formalism that is used to store the information items is not easily understandable. Our aim is to let the end users (domain experts) express their preferences and constraints over the observations that should be fused. Therefore the understandability of the knowledge representation formalism is a major concern.

In the work related by [Sambhoos *et al.*, 2008], information item extracted from texts written in natural language are stored as RDF triples. These triples are then organized in more complex graph structures called "observation graphs". Requests can then be processed on the set of observations, in order to determine whether a specific situation occurred or not.

6.2 Similarity of concepts

The issue of processing the similarity of two concepts that belong to two conceptual graphs was studied in [Zhong *et al.*, 2002] for instance. The operation of projection is used for semantic information retrieval. A request graph is projected on the graphs of a knowledge base were information is stored as conceptual graphs. In order to improve the performance of their system, the authors use a similarity measure between the concepts of the request graph and the concepts of the information graphs. Besides the ones that are equal, the concepts that are similar enough are linked one to another. The similarity measure relies on the use of the type hierarchy and the difference between two concept types is lessen according to their depth in the type hierarchy.

[Gandon *et al.*, 2008] uses an extension of the similarity measure proposed in [Zhong *et al.*, 2002] in order to relax the constraint of equality or specialization of the conceptual types during the projection of one graph on a second one.

The major drawback of these two approaches, regarding our aim of information fusion, is that they don't take into account the similarity between the concepts values. Our fusion approach has to be able to deal with the fusion of concepts that have different but sufficiently individual markers.

6.3 Entity Reconciliation

Entity Reconciliation is a problem faced in many domains such as data bases merging, ontology alignment, natural language processing, etc. where several source of information are used. It consists of deciding whether two descriptions (or identifiers) refer to the same entity of the real world.

The first studies regarding entity reconciliation relied on similarity processing of the different entity identifiers. The measures developed within these studies are tuned to specific application domains. Further studies such as [Bilenko and Mooney, 2003] propose learnable measures, so that the approach remains generic and can be adapted to different application domains, thanks to a learning phase.

More recently, studies combine the comparison of identifiers with the use of the context of the compared entity descriptions. Among others, the graphs structures are used, when complex entities or situations are structured as graphs. According to their neighbors in the graph they belong to, it is possible to decide whether two identifiers relate to the same entity ([Bhattacharya and Getoor, 2005] and [Sais *et al.*, 2007] for instance).

7 Conclusion and Perspectives

In previous work, we intuitively presented an approach for information fusion, using the conceptual graphs model. The model is used for domain knowledge representation and for fusion. For the fusion process, we rely on the use of the maximal join operation defined by Sowa which is central to our fusion process. However, domain knowledge is very important within the fusion process. It is used in order to confront the information items coming from the different sources and resolve conflicts between the different points of view if necessary.

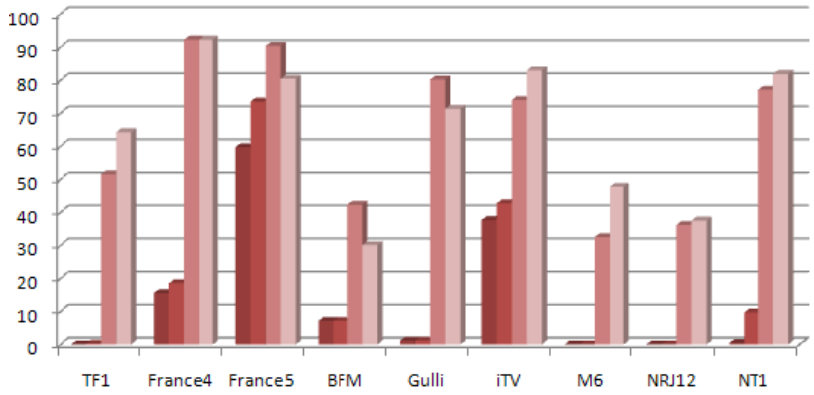


Figure 3: Percentage of programs properly found ((1) no strategy, (2) strategy 1, (3) Strategy 2, (4) Strategy 3)

In this paper, we detail our approach and formalize the use of domain knowledge inside maximal join operation, within the fusion process. We illustrate our work within the context of a TV program recommendation system and emphasize on the importance of using domain knowledge inside the fusion process.

Among others, our future work will concentrate on a step prior to the fusion. Indeed, relying on the studies dealing with graph similarity ([Sorlin *et al.*, 2003], [Gandon *et al.*, 2008]...), we will develop an other facet of our work. Prior to the fusion phase, we will focus on the discrimination of the observations that should be fused or not, regarding their global compatibility. The fusion of two graphs using the maximal join operation may be complex and time consuming, according to the structure of the graphs that should be fused. Our aim then, is to verify first, using simple and local comparisons relying on similarity measures and unicity constraints whether two graphs are compatible and thus mergeable.

References

[Bhattacharya and Getoor, 2005] Indrajit Bhattacharya and Lise Getoor. Entity resolution in graph data. Technical report, University of Maryland, College Park, October 2005.

[Bilenko and Mooney, 2003] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *proceeding of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, 2003.

[Buford *et al.*, 2008] J.F. Buford, L. Lewis, and G. Jakobson. Insider threat detection using situation-aware mas. In *11th International Conference on Information Fusion*, pages 212–219, Cologne, Germany, 2008.

[Chein and Mugnier, 2008] M. Chein and M.L. Mugnier. *Graph-based Knowledge Representation. Computational Foundations of Conceptual Graphs*. Springer, 2008.

[Gandon *et al.*, 2008] F. Gandon, O. Corby, I. Diop, and M. Lo. Distances sémantiques dans des applications de gestion d'information utilisant le web sémantique. In *Semantic similarity workshop, EGC 2008*, Sophia Antipolis, France, 2008.

[Laskey, 2008] K. Laskey. Probabilistic ontologies for knowledge fusion. In *11th International Conference on Information Fusion*, pages 1402–1409, Cologne, Germany, 2008.

[Laudy and Ganascia, 2008] C. Laudy and J-G. Ganascia. Information fusion in a tv program recommendation system. In *11th International Conference on Information Fusion*, pages 1455–1462, Cologne, Germany, July 2008.

[Laudy and Goujon, 2009] C. Laudy and B. Goujon. Soft data analysis within a decision support system. In *to appear in FUSION 2009*, 2009.

[Matheus *et al.*, 2003] C. Matheus, M. Kokar, and K. Baclawski. A core ontology for situation awareness. In *6th International Conference on Information Fusion*, pages 545–552, Cairns, Queensland, Australia, 2003.

[Rickard, 2006] J. T. Rickard. Level 2/3 fusion in conceptual spaces. In *9th International Conference on Information Fusion*, Florence, Italy, 2006.

[Sais *et al.*, 2007] Fatiha Sais, Nathalie Pernelle, and Marie-Christine Rousset. L2r: a logical method for reference reconciliation. In *Twenty-second AAAI Conference on Artificial Intelligence*, pages 329–334, July 2007.

[Sambhoos *et al.*, 2008] K. Sambhoos, J. Llinas, and E. Little. Graphical methods for real-time fusion and estimation with soft message data. In *11th International Conference on Information Fusion*, pages 1621–1628, Cologne, Germany, 2008.

[Sorlin *et al.*, 2003] S. Sorlin, P-A Champin, and C. Solnon. Mesurer la similarité de graphes étiquetés. In *9èmes Journées Nationales sur la résolution pratique de problèmes NP-Complets*, pages 325–339, 2003.

[Sowa, 1984] J. F. Sowa. *Conceptual Structures. Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA, 1984.

[Zhong *et al.*, 2002] J. Zhong, H. Zhu, J. Li, and Y. Yu. Conceptual graph matching for semantic search. In *10th international Conference on Conceptual Structures*, pages 92–106, 2002.

Generalizing Continuous Time Bayesian Networks with Immediate Nodes

Luigi Portinale and **Daniele Codetta-Raiteri**
Department of Computer Science
University of Piemonte Orientale "A. Avogadro"
{portinal, raiteri}@di.unipmn.it

Abstract

An extension to Continuous Time Bayesian Networks (*CTBN*) called Generalized *CTBN* (*GCTBN*) is presented; the formalism allows one to model, in addition to continuous time delayed variables (with exponentially distributed transition rates), also non delayed or "immediate" variables, which act as standard chance nodes in a Bayesian Network. The usefulness of this kind of model is discussed through an example concerning the reliability of a simple component-based system. A semantic model of *GCTBN*s, based on the formalism of Generalized Stochastic Petri Nets (*GSPN*) is outlined, whose purpose is twofold: to provide a well-defined semantics for *GCTBN*s in terms of the underlying stochastic process, and to provide an actual mean to perform inference (both prediction and smoothing) on *GCTBN*s. The example case study is then used, in order to highlight the exploitation of *GSPN* analysis for posterior probability computation on the *GCTBN* model.

1 Introduction

Temporal probabilistic graphical models allow for a factorization of the state space of a process, resulting in better modeling and inference features. Such models are usually based on graph structures, grounded on the theory of Bayesian Networks (*BN*). When time is assumed to be discrete, Dynamic Bayesian Networks (*DBN*) [7; 10] can be adopted. However, a discrete time assumption is not always adequate; for these reasons, Continuous Time Bayesian Networks (*CTBN*) have been proposed in [11; 12] and refined in [14]. Extensions to the basic model have also been proposed both regarding the use of indirect graph models [4] and the use of Erlang-Coxian distributions on the transition time [6].

In this paper, we propose another kind of extension and, in particular, a generalization of the standard *CTBN* framework, by allowing the presence of nodes which have no explicit temporal evolution; the values of such nodes are, in fact, "immediately" determined, depending on the values of other nodes in the network. The resulting framework is called Generalized *CTBN* (*GCTBN*) and is formally presented in Sec. 2. *GCTBN*s allow the modeling of processes having both a continuous-time temporal component and an immediate component capturing the logical/probabilistic interactions among modeled variables. While these modeling features are actually possible in discrete time *DBN*s, our work is, at the best of our knowledge, the first attempt trying to mix in the same *BN*, continuous-time delayed nodes with standard chance nodes.

In case of continuous time, a model having similar features can be found in the framework of Petri Nets, namely Generalized Stochastic Petri Nets (*GSPN*) [1][1] . Briefly, *GSPN*s are stochastic Petri nets, with two different sets of transitions, namely *temporal* with an exponentially distributed delay, and *immediate* transitions (with no delay), having priority over temporal ones. We propose to express a *GCTBN* model in terms of a *GSPN*, by means of a set of translation rules (see [13] for details). This translation is twofold: (1) it provides a well-defined semantics for a *GCTBN* model, in terms of the underlying stochastic process it represents (this is discussed in Sec. 4); (2) it provides an actual mean to perform inference on the *GCTBN* model, by exploiting well-studied analysis techniques for *GSPN*s, as described in Sec. 5.

Actually, in case of a *CTBN* exact inference may often be impractical, so approximations through message-passing algorithms on cluster graphs [12; 14], or through sampling [4; 5] have been proposed. In the present work, we take advantage of the correspondence between *GCTBN* and *GSPN*, in order to propose inference algorithms for *GCTBN* models (both for prediction and smoothing), based on *GSPN* solution algorithms and providing the exact solution of the model.

The possibilities offered by *GCTBN*s, can be exploited in several applications. For example, in system reliability analysis, it is very practical to distinguish between system components (having a temporal evolution) and specific modules or subsystems, whose behavior has to be modeled for the analysis. For instance, in Fault Tree Analysis (*FTA*), basic events represents the system components with their failure rates, while non-basic events are logical gates identifying modules of the system under examination [15]. In Dynamic Fault Trees [3], logical gates identifying sub-modules, can be combined with dynamic gates, modeling time-dependent dependencies (usually assuming continuous time) among com-

[1] Because of space restrictions, we refer the interested reader to [1; 13] for details and formal definitions.

ponents or sub-modules. Also in this case, it is very important to distinguish, at the modeling level, between delayed and immediate entities. Of course, similar considerations apply in other tasks as well, as in medical diagnosis, financial forecasting, biological process modeling, etc. Sec. 3 provides a simple case study in the reliability field, supporting the presentation of the concepts in the following sections.

2 The generalized *CTBN* model

Following the original paper in [11], a *CTBN* is defined as follows:

Definition 2.1 *Let $X = X_1, \ldots, X_n$ be a set of discrete variables, a* CTBN *over X consists of two components. The first one is an initial distribution P_X^0 over X (possibly specified as a standard* BN *over X). The second component is a continuous-time transition model specified as (1) a directed graph G whose nodes are $X_1, \ldots, X_n$ (and with $Pa(X_i)$ denoting the parents of X_i in G); (2) a conditional intensity matrix $Q_{X_i|Pa(X_i)}$ for every $X_i \in X$.*

We can now introduce the notion of a Generalized *CTBN* (*GCTBN*).

Definition 2.2 *Given a set of discrete variables $X = \{X_1, \ldots, X_n\}$ partitioned into the sets D (delayed variables) and I (immediate variables) (i.e. $X = D \cup I$ and $D \cap I = \emptyset$), a Generalized Continuous Time Bayesian Network (*GCTBN*) is a pair $N = \langle P_X^0, G \rangle$ where*

- *P_X^0 is an initial probability distribution over X;*
- *G is a directed graph whose nodes are $X_1, \ldots, X_n$ (and with $Pa(X_i)$ denoting the parents of X_i in G) such that*
  1. *there is no directed cycle in G composed only by nodes in the set I;*
  2. *for each node $X \in I$ a conditional probability table $P[X|Pa(X)]$ is defined (as in standard* BN*);*
  3. *for each node $Y \in D$ a conditional intensity matrix $Q_{Y|Pa(Y)}$ is defined (as in standard* CTBN*).*

Delayed (or temporal) nodes are, as in case of a *CTBN*, nodes representing variables with a continuous time evolution ruled by exponential transition rates, and conditioned by the values of parent variables (that may be either delayed or immediate). Immediate nodes are introduced, in order to capture variables whose evolution is not ruled by transition rates associated with their values, but is conditionally determined at a given time point, by other variables in the model. Such variables are then treated as usual chance nodes in a *BN* and have a standard Conditional Probability Table (*CPT*) associated with them.

A few words are worth to be spent for the structure of the graph modeling the *GCTBN*. While it is in general possible to have cycles in the graph (as in *CTBN*) due to the temporal nature of some nodes, such cycles cannot be composed only by immediate nodes. Indeed, if this was the case, we would introduce static circular dependencies among model variables.

Finally, it is worth noting that the initial distribution P_X^0 can in general be specified only on a subset of X. In particular, let $R \subset I$ be the set of root nodes (i.e. node with no parent in G) which are immediate, then the initial distribution can be computed as $P_X^0 = P_{R \cup D}^0 \prod_{Y_j \in (I-R)} P[Y_j|Pa(Y_j)]$. In fact, while it is necessary to specify an initial distribution over delayed variables, the distribution on the immediate variables can be determined depending on the values of their parents; of course if an immediate variable is modeled as a root node, an initial prior probability is needed[2].

3 An illustrative example

We now consider a case study which can be easily modeled in form of *GCTBN*. This is a typical case in the field of reliability analysis, and consists of a small system composed by the main component A and its "warm" spare component B. This means that initially both components are working, but A is active while B is dormant; in case of failure of A, B is activated in order to replace A in its function. We assume that the activation of B occurs with a 0.99 probability. If B fails before A, B can not replace A.

The system is considered as failed if A is failed and B is dormant or failed. We suppose that only while the system is failed, the components A and B undergo repair. As soon as the repair of one of the components is completed, the component re-starts in working state: if A is repaired the system becomes operative again and the repair of B is suspended; if instead B is repaired, this may determine one of these two situations: 1) B may become active with probability $p = 0.99$ and consequently the system becomes operative again and the repair of A is suspended. 2) B may become dormant with probability $1 - p$, so the system is still failed and the repair of B goes on.

The component time to failure or repair is a random variable ruled by the negative exponential distribution according to the component failure or repair rate respectively. In the case of the main component A, the failure rate is λ_A =1.0E$-$06 h^{-1}. The failure rate of B, λ_B, changes according to its current state: if B is dormant, λ_B is equal to 5.0E$-$07 h^{-1}; if instead B is active, λ_B is equal to 1.0E$-$06 h^{-1}. Because of this, the spare is defined as "warm" [3]. A and B have the same repair rate: μ_A =μ_B =0.01 h^{-1}.

3.1 The *GCTBN* model

The case study described above is represented by the *GCTBN* model in Fig. 1 where the variables A, B, SYS represent the state of the component A, the component B and the whole system respectively. All the variables are binary because each entity can be in the working or in the failed state (for the component B, the working state comprises both the dormancy and the activation). In particular, we represent the working state with the value 1, and the failed state with the value 2.

The variable A influences the variable B because the failure rate of the component B depends on the state of A. Both the variables A and B influence the variable SYS because the

[2] Actually, since prior probabilities on immediate root nodes are a special case of CPT, we could also simply write $P_X^0 = P_D^0 \prod_{Y_j \in I} P[Y_j|Pa(Y_j)]$, to emphasize the fact that, for the specification of the temporal evolution of the model, the only initial distribution is on delayed nodes (the other parameters are actually a fixed specification on the network).

state of the whole system depends on the state of the components A and B. The arcs connecting the variable SYS to A and B respectively, concern the repair of the components A and B: only while the system is failed, they can be repaired.

The variables A and B in the *GCTBN* model in Fig. 1 are delayed variables (Sec. 2) and are drawn as double-circled nodes: both variables implicitly incorporate a Continuous Time Markov Chain (*CTMC*) composed by two states: 1 (working) and 2 (failed). Due to the assumption that both components are initially supposed to work, the initial probability distribution is set equal to 1 for states $A = 1$ and $B = 1$. In the case of A, the current value of the rates λ_A and μ_A depends on the current value of the variable SYS, the only one influencing A. This is shown by the Conditional Intensity Matrix (*CIM*) reported in Tab. 1.a, where we can notice that the rate μ_A is not null only if the value of SYS is 2. The rate λ_A instead, is constant.

In the case of the variable B, the current value of the rates λ_B and μ_B depends on the current value of the variables A and SYS, as shown by the *CIM* appearing in Tab. 1.b, where λ_B is increased only when A is equal to 2 and SYS is equal to 1 (this implies that B is active). As in the case of the variable A, the rate μ_B is not null only if the value of SYS is 2. Notice that the combination $A = 1, SYS = 2$ is impossible, so the corresponding entries are not significant.

The variable SYS is immediate (Sec. 2) and is shown as a circle node in Fig. 1. It is characterized by the *CPT* appearing in Tab. 1.c. In particular, SYS is surely equal to 1 if A is equal to 1, and surely equal to 2 if both A and B are equal to 2. In the case of A equal to 2 and B equal to 1, SYS assumes the value 1 with probability 0.99 (this implies the activation of the spare component B), or the value 2 with probability 0.01 (this implies that B is still dormant).

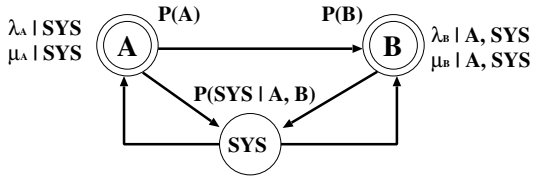


Figure 1: *GCTBN* model of the case study.

4 A Petri Net semantics for *GCTBN*

The combination in a single model of entities explicitly evolving over time with entities whose determination is "immediate", has been already proposed in frameworks other than *CTBN*; as we have already noticed in Sec. 1, *DBN*s provide an example, in case of discrete time. In case of continuous time, *GSPN*s allow to model both kinds of entities by means of temporal and immediate transitions respectively. This means that, in case both an immediate and a temporal transition are enabled, the firing of the former takes precedence over the firing of the latter. Immediate transitions may also have different priority levels among them.

The stochastic process associated with a *GSPN* is a homogeneous continuous time semi-Markov process that can be

a)

	1		2	
			SYS	λ_A
1			1	1.0E-06 h^{-1}
			2	1.0E-06 h^{-1}
	SYS	μ_A		
2	1	0 h^{-1}		
	2	0.01 h^{-1}		

b)

	1			2		
				A	SYS	λ_B
1				1	1	5.0E-07 h^{-1}
				1	2	–
				2	1	1.0E-06 h^{-1}
				2	2	5.0E-07 h^{-1}
	A	SYS	μ_B			
2	1	1	0 h^{-1}			
	1	2	–			
	2	1	0 h^{-1}			
	2	2	0.01 h^{-1}			

c)

A	B	SYS	Prob.
1	1	1	1
1	1	2	0
1	2	1	1
1	2	2	0
2	1	1	0.99
2	1	2	0.01
2	2	1	0
2	2	2	1

Table 1: a) *CIM* for the variable A. b) *CIM* for the variable B. c) *CPT* for the variable SYS in the *GCTBN* model in Fig. 1.

analyzed either by solving the so called *Embedded Markov Chain* or by removing from the set of possible states, the so-called *vanishing states* or *markings* and by analyzing the resulting *CTMC* [1]. Vanishing states are the state (or markings) resulting from the firing of immediate transitions; they can be removed, since the system does not spend time in such states. This removal operation has also the advantage of reducing (often in a significant way) the set of possible states to be analyzed.

Solution techniques for *GSPN*s have received a lot of attention, especially with respect to the possibility of representing in a compact way the underlying *CTMC* and in solving it efficiently [8; 9]. Once a *GCTBN* has been compiled into a *GSPN* [13], such techniques can be employed to compute inference measures on the original *GCTBN* model (see Sec. 5).

There are two main analyses that can be performed with a *GSPN*: *steady state* and *transient analysis*. In the first case, the equilibrium distribution of the states is computed, while in the latter, such a distribution is computed at a given time point. In particular, solving a *GSPN* (for either steady state or transient analysis) can provide the probability distribution of the number of tokens in each place. This information can then be exploited, in order to perform inference on the original *GCTBN* model as it will be shown in Sec. 5.

4.1 The *GSPN* model for the case study

According to the conversion rules described in [13], the *GCTBN* of the case study in Fig. 1 can be converted into the *GSPN* model shown in Fig. 2 where the places A, B and SYS correspond to the variables in the *GCTBN* model. The value of a *GCTBN* variable is mapped into the marking (number of tokens) of the corresponding place in the *GSPN*. Let us consider the place B in the *GSPN*: the marking of the place B can be equal to 1 or 2, the same values that the variable B in the *GCTBN* can assume. B is a delayed variable and its

initialization is modeled in the *GSPN* by the immediate transitions B_init_1 and B_init_2 called *"init" transition*. Such transitions are both initially enabled to fire with the effect of setting the initial marking of the place B to 1 or 2 respectively. The probability of these transitions to fire corresponds to the initial probability distribution of the variable B.

The variation of the marking of the place B is determined by the timed transitions B_1_2 and B_2_1. The transition B_1_2 is enabled to fire when the place B contains one token; the effect of its firing is setting the marking of B to 2. The transition B_2_1 instead, can fire when the marking of the place B is equal to 2, and turns it to 1. The dependency of the transition rate of a variable on the values of the other variables in the *GCTBN* model, becomes in the *GSPN* model, the dependency of the firing rate of a timed transition on the markings of the other places. For instance, in the *GCTBN* model the variable B depends on A and SYS; let us consider λ_B, the transition rate of B from 1 to 2 depending on the values of the variables A and SYS (Tab. 1.b). In the *GSPN* model, λ_B becomes the firing rate of the timed transition B_1_2 whose value depends on the marking of the places A and SYS, and assumes the same values reported in Tab. 1.b. The firing rate of the timed transition B_2_1 instead, will correspond to the rate μ_B reported in Tab. 1.b, still depending on the marking of the places A and SYS.

The initialization of the marking of the place A is modeled by the immediate init transitions A_init_1 and A_init_2, while the variation of its marking is modeled by the timed transitions A_1_2 and A_2_1, but in this case their firing rate will depend only on the marking of the place SYS, because in the *GCTBN* model the variable A depends only on the variable SYS. Such variable is immediate in the *GCTBN* and depends on A and B. Therefore in the *GSPN* each time the marking of the place A or of the place B is modified, the marking of SYS has to be immediately updated: each time the transition A_1_2, A_2_1, B_1_2 or B_2_1 fires, one token appears in the place $emptySYS$; this determines the firing of the immediate transition $reset_SYS_1$ or $reset_SYS_2$ having priority over the other immediate transitions (priority level $\pi = 2$ in Fig. 2), with the effect of removing any token inside the place SYS. At this point, the marking of such place has to be set according to the current marking of the places A and B. This is done by one of the immediate transitions set_SYS_1, set_SYS_2, set_SYS_3, set_SYS_4, set_SYS_5. Each of them corresponds to one entry having not null probability in the *CPT* of the variable SYS in the *GCTBN* model (Tab. 1.c). Each of such transitions has the same probability and the same effect on the marking of the place SYS, as the corresponding entry in the *CPT*.

5 Inference

Standard inference tasks in temporal probabilistic models are *prediction* and *smoothing* [10]. *Prediction* is the task of computing the probability of a set of queried variables, given past evidence, i.e. predicting a future state taking into consideration the observations up to now (a special case occurs when the last evidence time point and the query time are the same and is called *Filtering* or *Monitoring*). *Smoothing* is the task

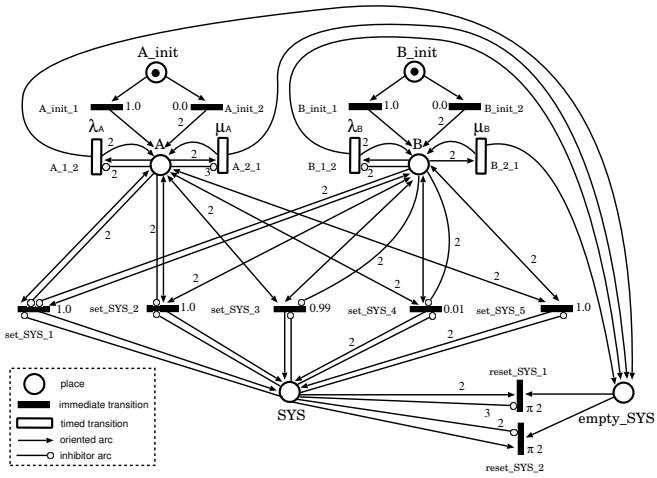


Figure 2: *GSPN* model obtained from the *GCTBN* in Fig. 1.

of estimating what happened $k > 0$ time points in the past, given all the evidence (observations) up to now. Such tasks can be accomplished, depending on the model adopted, by inference procedures usually based on specific adaptation of standard algorithms for Bayesian Networks. For instance, in *DBN* models, both exact algorithms based on junction tree [10] as well as approximate algorithms exploiting the net structure [2] or based on stochastic simulation can be employed. In this paper, we propose the conversion into *GSPN*, and the *GSPN* analysis methods, as means to compute exact inference on the *GCTBN* model, for both prediction and smoothing tasks.

Computing the probability of a given variable assignment $X = x_i$ at time t, will correspond to compute the probability of having i tokens in the place modeling X at time t. In particular, if $P()$ is the probability function associated with the *GCTBN* model and $Pr\{\}$ is the probability function associated with the *GSPN* model, then $P(X_t = x_i) = Pr\{\#X_t = i\}$, where X_t is the value of X at time t and $\#X_t$ is the number of tokens in the place corresponding to X at time t.

5.1 Prediction Inference

The task of prediction consists in computing the posterior probability at time t of a set of queried variables $Q \subseteq D \cup I$, given a stream of observations (evidence) $e_{t_1}, \ldots, e_{t_k}$ from time t_1 to time t_k with $t_1 < \ldots < t_k < t$. Every evidence e_{t_j} consists of a (possibly different) set of instantiated variables.

Prediction can then be implemented by repeatedly solving the transient of the corresponding *GSPN* at the observation and query times. Of course, any observation will condition the evolution of the model, so the suitable conditioning operations must be performed before a new *GSPN* resolution. The pseudo-code for the prediction procedure is shown in Fig. 3. Notice that, in the special case of *filtering*, the last evidence would be available at the query time (i.e. $t = t_k$ in Fig. 3); in such a case, the update of the transition weights (last statement in the `for` cycle) is not necessary, as well as the final transient solution. The procedure would then simply output

Procedure PREDICTION
INPUT: a set of queried variables Q, a query time t, a set of temporally labeled evidences $e_{t_1}, \ldots, e_{t_k}$ with $t_1 < \ldots < t_k < t$
OUTPUT: $P(Q_t | e_{t_1}, \ldots, e_{t_k})$

```
let t_0 = 0;
for i = 1 to k {
   solve the GSPN transient at time (t_i - t_{i-1});
   compute from transient, p_i(j) = Pr{X_j | e_{t_i}} for X_j ∈ D ∪ R;
   update the weights of the immediate init transitions of X_j
     according to p_i(j); }
solve the GSPN transient at time (t - t_k);
compute from transient, r = Pr{Q};
output r;
```

Figure 3: The prediction inference procedure.

$Pr\{Q|e_t\}$ computed from the last transient analysis.

In case there is evidence available at time $t_0 = 0$, if the evidence is on variables $X \in D \cup R$, then it is incorporated into their "init" distribution; if the evidence is on variables $X \in I - R$, then the "init" of the other variables are updated by solving the transient at time $t_0 = 0$.

5.2 Smoothing Inference

The smoothing task consists in computing the probability at time t of a set of queried variables $Q \subseteq D \cup I$, given a stream of observations (evidence) $e_{t_1}, \ldots, e_{t_k}$ from time t_1 to time t_k with $t < t_1 < \ldots < t_k$. The issue is how to condition on variables observed at a time instant that follows the current one. The idea is then to try to reformulate the problem in such a way that it can be reduced to a prediction-like task. The approach is then based on the application of the Bayes rule as follows:

$$P(Q_t|e_{t_1}, \ldots, e_{t_k}) = \alpha P(Q_t) P(e_{t_1}, \ldots, e_{t_k}|Q_t)$$
$$= \alpha P(Q_t) P(e_{t_1}|Q_t) \ldots P(e_{t_k}|e_{t_1}, \ldots, e_{t_{k-1}}, Q_t)$$

In this way, every factor in the above formula is conditioned on the past and can be implemented as in prediction. However, the computation of the normalization factor α, requires that a separate computation must be performed for every possible assignment of the query Q. The interesting point is that such computations are independent, so they can be possibly performed in parallel[3]. Once the computation has been performed for every query assignment, then results can be normalized to get the actual required probability values.

The pseudo-code for the smoothing procedure is shown in Fig. 4. The `normalize` operator, just divide any entry of the vector A by the sum of all the entries, in order to provide the final probability vector of the query.

5.3 Example of inference in the case study

Consider again the case study of Fig. 1. Concerning prediction, let us consider to observe the system working ($SYS = 1$) at time $t = 10^5 h$ and the system failed ($SYS = 2$) at time $t = 2 \cdot 10^5 h$. By considering the procedure outlined in Fig. 3 we can compute the probability of component A being working at time $t = 5 \cdot 10^5 h$, conditioned by the observation

[3] An alternative can be to directly compute the denominator of the Bayes formula (i.e. the probability of the evidence stream); however, this requires a larger number of transient solutions if the length of the observation stream is greater than the the number N of assignments of Q (i.e.if $k > N$), as is usually the case.

Procedure SMOOTHING
INPUT: a set of queried variables Q, a query time t, a set of temporally labeled evidences $e_{t_1}, \ldots, e_{t_k}$ with $t < t_1 < \ldots < t_k$
OUTPUT: $P(Q_t|e_{t_1}, \ldots, e_{t_k})$;

```
let N be the cardinality of possible assignments q_i (1≤i≤N) of Q;
A: array[N];
for i = 1 to N {
  //possibly in parallel
  A[i]=SMOOTH(q_i); }
output normalize(A);

Procedure SMOOTH(q) {
   t_0 = t;
   solve the GSPN transient at time t;
   compute from transient, r = Pr{Q = q};
   ev = q;
   for i = 1 to k {
     compute from transient, p_{i-1}(j) = Pr{X_j | ev} for X_j ∈ D ∪ R;
     update the weights of the immediate init transitions of X_j
       according to p_{i-1}(j);
     solve the GSPN transient at time (t_i - t_{i-1});
     compute from transient, p_i(e) = Pr{e_{t_i}};
     r = r p_i(e);
     ev = e_{t_i}; }
   output r; }
```

Figure 4: The smoothing inference procedure.

stream, as follows: (1) we solve the transient at $t = 10^5 h$ and we compute the probabilities of A and B, conditioned by the observation $SYS = 1$; (2) we use the above computed probabilities as the new init probabilities for the places A and B of the *GSPN*; (3) we solve the transient for another time interval $t = 10^5 h$ and we compute the probabilities of A and B, conditioned by the observation $SYS = 2$; (4) we use the above computed probabilities as the new init probabilities for the places A and B of the *GSPN*; (5) we solve the transient for a time interval $t = 3 \cdot 10^5 h$ and we finally compute the probability of the query A. Tab. 2 shows the values computed

Time (h)	$P(A=1\|e)$	$P(A=2\|e)$	$P(B=1\|e)$	$P(B=2\|e)$
100000	0.909228	0.090772	0.952445	0.047555
200000	0	1	0.071429	0.928571
500000	**0.521855**	**0.478145**	-	-

Table 2: Probabilities for prediction inference in the case study (e is the current accumulated evidence).

during the above process. The last row shows the required results.

Concerning smoothing inference, let us suppose to have observed the system working at time $t = 3 \cdot 10^5 h$ and failed at time $t = 5 \cdot 10^5 h$. We ask for the probability of component A at time $t = 2 \cdot 10^5 h$, conditioned by the above evidence. By considering the procedure outlined in Fig. 4 we can compute the required probabilities as follows: (1) we first consider the case $A = 1$; (2) we solve the transient at $t = 2 \cdot 10^5 h$ and we compute $r1 = P(A = 1)$; (3) we condition A and B on $A = 1$ and we determine the new init probabilities for A and B; (4) we solve the transient for $t = 10^5 h$ (to reach time $3 \cdot 10^5 h$) and we compute $r2 = P(SYS = 1)$; we also condition A and B on $SYS = 1$ and we use such values as new init probabilities for places A and B; (5) we solve the transient for $t = 2 \cdot 10^5 h$ (to reach time $5 \cdot 10^5 h$) and we compute $r3 = P(SYS = 2)$; (6) we compute the un-normalized probability of $A = 1$ as $p1 = r1 \cdot r2 \cdot r3$; By performing the above steps also for the case $A = 2$ we can similarly compute the un-normalized

probability of $A = 2$, namely $p2$. A simple normalization over $p1$ and $p2$ will then produce the required results. Tab. 3 shows the values computed during the above process (partial results $r1, r2, r3$ are shown in bold).

$P(A = 1)$ at $t = 2 \cdot 10^5$ = **0.833086**

Time (h)	$P(A = 1\|e)$	$P(B = 1\|e)$	$P(SYS = 1\|e)$	$P(SYS = 2\|e)$
200000	1	0.891238	-	-
300000	0.913981	0.854028	**0.999988**	-
500000	-	-	-	**0.000022**

p1=0.0000183277

$P(A = 2)$ at $t = 2 \cdot 10^5$ = **0.166914**

Time (h)	$P(A = 1\|e)$	$P(B = 1\|e)$	$P(SYS = 1\|e)$	$P(SYS = 2\|e)$
200000	0	0.999922	-	-
300000	0.056648	0.952429	**0.999950**	-
500000	-	-	-	**0.000049**

p2=0.0000081784

$P(A = 1\|e)$	$\frac{p1}{p1+p2}$=**0.691452**
$P(A = 2\|e)$	$\frac{p2}{p1+p2}$=**0.308548**

Table 3: Probabilities for smoothing inference in the case study (e is the current accumulated evidence).

6 Conclusions and Future Works

In this paper we have presented a generalized *CTBN* formalism, allowing one to mix in the same model continuous time delayed variables with standard "immediate" chance variables. The usefulness of this kind of model has been discussed through an example concerning the reliability of a simple component-based system. The semantics of the proposed *GCTBN* formalism has been provided in terms of Generalized Stochastic Petri Nets (*GSPN*), a well-known formalism isomorph to semi-Markov processes, through which it is also possible to exploit well established analysis techniques, in order to perform standard prediction or smoothing inference. In particular, adopting *GSPN* solution algorithms as the basis for *GCTBN* inference, allows one to take advantage of specialized methodologies for solving the underlying stochastic process, that are currently able to deal with extremely large models; in particular, such techniques (based on data structures like matrices or decision diagrams) allow for one order of magnitude of increase in the size of the models to be solved exactly, with respect to standard methods, meaning that models with an order of 10^{10} tangible states can actually be solved [8; 9].

However analyzing a *GCTBN* by means of the underlying *GSPN* is only one possibility that does not take explicit advantage of the structure of the graph as in *CTBN* algorithms [12; 14]. Our future works will try to investigate the possibility of adopting cluster-based or stochastic simulation approximations, even on *GCTBN* models, and in comparing their performance and quality with respect to *GSPN*-based solution techniques. In particular, since Petri nets are a natural framework for event-based simulation, it would be interesting to investigate how simulation-based approximations can be actually guided by the underlying *GSPN* model. Finally, since symbolic representations (based on matrices or decision diagrams) have been proved very useful for the analysis of *GSPN* models, it would also be of significant interest to study the relationships between such representations and the inference procedures on probabilistic graphical models in general, since this could in principle open the possibility of new classes of algorithms for *BN*-based formalisms.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. J. Wiley, 1995.

[2] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings UAI 1988*, pages 33–42, 1998.

[3] J. Bechta Dugan, S.J. Bavuso, and M.A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Trans. on Reliability*, 41:363–377, 1992.

[4] T. El-Hay, N. Friedman, and R. Kupferman. Gibbs sampling in factorized continuous time Markov processes. In *Proc. 24rd UAI'08*, 2008.

[5] Y. Fan and C. Shelton. Sampling for approximate inference in continuous time Bayesian networks. In *Proc. 10th Int. Symp. on AI and Mathematics*, 2008.

[6] K. Gopalratnam, H. Kautz, and D.S. Weld. Extending continuous time Bayesian networks. In *Proc. AAAI'05*, pages 981–986, Pittsburgh, PA, 2005.

[7] U. Kjaerulff. dHugin: a computational system for dynamic time-sliced Bayesian networks. *International Journal of Forecasting*, 11:89–101, 1995.

[8] A.S. Miner. Decision diagrams for the exact solution of Markov models. *Proceedings in Applied Mathematics and Mechanics (PAMM)*, 7(1), 2007.

[9] A.S. Miner and D. Parker. Symbolic representation and analysis of large probabilistic systems. In *Validation of Stochastic Systems, LNCS 2925*, pages 296–338. Springer, 2004.

[10] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD Thesis, UC Berkley, 2002. http://www.cs.ubc.ca/ murphyk/Thesis/thesis.html.

[11] U. Nodelman, C.R. Shelton, and D. Koller. Continuous Time Bayesian Networks. In *Proc. 18th UAI'02*, pages 378–387, 2002.

[12] U. Nodelman, C.R. Shelton, and D. Koller. Expectation propagation for continuous time Bayesian networks. In *Proc. 21st UAI'05*, pages 431–440, 2005.

[13] L. Portinale and D. Codetta-Raiteri. A GSPN semantics for continuous time Bayesian networks with immediate nodes. Technical Report TR-INF-2009-03-03-UNIPMN, Computer Science Dept., UPO, 2009. http://www.di.unipmn.it/TechnicalReports/TR-INF-2009-03-03-UNIPMN.pdf.

[14] S. Saria, U. Nodelman, and D. Koller. Reasoning at the right time granularity. In *Proc. 23rd UAI'07*, pages 421–430, 2007.

[15] W. G. Schneeweiss. *The Fault Tree Method*. LiLoLe Verlag, 1999.

Join Bayes Nets: A New Type of Bayes net for Relational Data

Oliver Schulte
Computer Science Dept.
Simon Fraser University
oschulte@cs.sfu.ca

Hassan Khosravi
Computer Science Dept.
Simon Fraser University
hkhosrav@cs.sfu.ca

Bahareh Bina
Computer Science Dept.
Simon Fraser University
bba18@cs.sfu.ca

Flavia Moser
Computer Science Dept.
Simon Fraser University
fmoser@cs.sfu.ca

Abstract

Many real-world data are maintained in relational format, with different tables storing information about entities and their links or relationships. The structure (schema) of the database is essentially that of a logical language, with variables ranging over individual entities and predicates for relationships and attributes. Our work combines the graphical structure of Bayes nets with the logical structure of relational databases to achieve knowledge discovery in databases. We introduce Join Bayes nets, a new type of Bayes nets for representing and learning class-level dependencies between attributes from the same table *and* from different tables; such dependencies are important for policy making and strategic planning. Focusing on class-level dependencies brings advantages in terms of the simplicity of the model and the tractability of inference and learning. As usual with Bayes nets, the graphical structure supports efficient inference and reasoning. We show that applying standard Bayes net inference algorithms to the learned models provides fast and accurate probability estimates for queries that involve attributes and relationships from multiple tables.

1 Introduction

Many real-world applications store data in relational format, with different tables for entities and their links. Standard machine learning techniques are applied to data stored in a single table, that is, in nonrelational, propositional or "flat" format [10]. The field of statistical-relational learning (SRL) aims to extend machine learning algorithms to relational data [6]. One of the major machine learning tasks is to use data to build a *generative statistical model* that represents the joint distribution of the random variables that describe the application domain [6]. In the single-table learning setting, the goal is often to represent predictive dependencies between the attributes of a single individual (e.g., between the intelligence and ranking of a student). In the SRL setting, the goal is often to represent in addition dependencies between attributes of different individuals that are related or linked to each other (e.g., between the intelligence of a student and the difficulty of a course given that the student is registered in the course). Many SRL models represent such dependencies on two different levels, a class dependency model and an instance dependency model. For instance, in a graphical SRL model, the nodes in the instance dependency model represent attributes of individuals or relationships [5]. The nodes in the class dependency model correspond to attributes of the tables. A class-level model is instantiated with the specific entities, their attributes and their relationships in a given database to obtain an instance dependency model. For instance, the class-level model may contain a node $age(S)$ to represent the age of a generic member of the student class, and the instance model may contain a node $age(Jack)$ to represent the age of a specific student $Jack$. The node $age(Jack)$ inherits the parameters and associations indicated at the class level for $age(S)$.

In this paper we apply Bayes nets (BNs) to model class-level dependencies between variables that appear in separate tables. What is new about our approach is that we focus on class-level variables only rather than making predictions about individual entities. Our class-level Bayes nets contain nodes that correspond to the descriptive attributes of the database tables, plus Boolean nodes that indicate the presence of a relationship; we refer to these as Join Bayes nets (JBNs). We introduce a new database join operation as a conceptual aid that provides semantics for JBNs. The focus on class-level dependencies brings advantages in terms of the simplicity of the model and the tractability of inference and learning, while it involves some loss of expressive power, because our BN model cannot answer queries about individual entities. Examples of applications that provide motivation for the class-level queries answered by our BN include the following.

(1) *Policy making and strategic planning.* A university administrator may wish to know which program characteristics attract high-ranking students, rather than predict the rank of a specific student in a specific program.

(2) *Query optimization* is one of the applications of SRL where a statistical model predicts a probability for given join conditions that can be used to infer the size of the join result [7]. The join conditions often do not involve specific individuals.

This paper defines JBN models and a probabilistic semantics for them. Our algorithmic contribution is an efficient dynamic programming procedure for parameter learning in JBNs. This algorithm solves the problem of estimating frequencies conditional on the *absence* of a relationship. Due to the construction of our Bayes nets, class-level queries can be answered using standard BN inference algorithms "as is".

Paper Outline We review background from relational databases and Bayes nets. Then we introduce our class-level Bayes nets and define their semantics. We describe algorithms for structure learning and parameter estimation. The algorithms and the inference capabilities of the Bayes nets they learn are evaluated on three data sets, one artificial and

two real-word ones (the MovieLens and the Financial data set).

Related Work Researchers in statistical-relational learning have developed a number of generative models that include attributes and relationships of entities; for an overview see [8; 4; 3]. Markov Logic Networks (MLNs) are a prominent class of SR models that are based on undirected graphs [3]. The most direct comparison of JBNs is with other directed models; we discuss Bayes Logic Networks (BLNs) [9] and Probabilistic Relational Models (PRMs) [5, Sec.5.5.3]. Similar points of comparison apply to other SRL models.

The class-level model of a BLN—-called a Bayes Logic Program (BLP)—is syntactically similar to a JBN: a JBN with n nodes into a n translates into n BLP clauses of the form $x_i|parent_{i,1}, parent_{i,2}, \ldots, parent_{i,k}$, where $i = 1, \ldots, n$ indexes the nodes and node x_i has k parents. In addition, a BLP features *combining rules*. These specify how instance-level predictions from information about different related entities are to be combined into a single prediction. For instance, if the task is to predict a specific student's intelligence based on his grade in 10 courses he has taken, the class-level BLP clauses may be used to predict the intelligence based on a single course, and the combining rule would specify how to collect these predictions into a single prediction for the specific student. A feature of JBNs not necessarily present in BLPs is that variables ranging over entities are associated with entity types (e.g., S ranges over entities in the $Student$ table); the use of such types is key for the probabilistic semantics of JBNs. As for inference, it appears that in principle a BLP could be translated into a Bayes net and standard BN inference algorithm could be used to carry out class-level inference; to our knowledge, this approach to lifted inference with BLNs has not yet been evaluated.

The class-level model of a PRM is also a directed graphical model, and the nodes in the PRM graph are essentially the same as in Join Bayes nets (if the PRM includess uncertainty about the existence of links [5, Sec.5.5.3]). Nodes are associated with entity types as in a JBN. In the case in which entity types may be related to themselves (e.g., the $Parent$ relationship relates people to people), a PRM may contain self-loops. In order to make predictions about individual entities given the other entities they are related to, a PRM requires the specification of an aggregate function for many-many relationships [5, Def.5.2]. For instance, if the task is to predict a specific student's intelligence based on his grade in 10 courses he has taken, a PRM may specify that the prediction is to be based on the student's average grade. The CP-tables for a class-level PRM may be defined in terms of the value of the aggregate functions. In that case, standard BN algorithms cannot be applied to the class-level PRM, and adaptations are required [5].

In addition to inference, the two major differences between JBNs and PRMs resp. BLNs concern semantics and learning. (1) In terms of semantics, SR models are usually viewed as a template for instance-level models: For a given database, the class-level model is instantiated with the specific entities, their attributes and their relationships to obtain an instance-level model, which inherits the parameters specified at the class level. In contrast, we do not view our class-level BNs as templates for instance-level BNs. Thus we avoid problems with potential cycles at the entity level, which is a major concern for directed relational models [5]. (2) In order to make predictions about individual entities given the other entities they are related to, BLNs and PRMs require extra components in addition to the Bayes net-like class-level structure (combining rules resp. aggregate functions). While these extra components considerably increase the expressive power of these models, they also substantially increase the complexity of learning. In particular, fitting the models to data requires evaluating their predictive power with regard to instance-level predictions that are based on the entire relational context of an entity. In contrast, inference and learning for JBNs can be carried out efficiently with algorithms whose design we outline in this paper.

$Student(\underline{student_id}, intelligence, ranking)$
$Course(\underline{course_id}, difficulty, rating)$
$Professor\ (\underline{professor_id}, teaching_ability, popularity)$
$Registered\ (\underline{student_id, Course_id}, grade, satisfaction)$

Table 1: A relational schema for a university domain. Key fields are underlined. An instance for this schema is given in Figure 1.

2 Preliminaries

We employ notation and terminology from [11] for a Bayesian Network. A **Bayes net structure** is a directed acyclic graph (DAG) G, whose nodes comprise a set of random variables denoted by V. A Bayes net (BN) is a pair $\langle G, \theta_G \rangle$ where θ_G is a set of parameter values that specify the probability distributions of children conditional on instantiations of their parents, i.e. all conditional probabilities of the form $P(X = x|\mathbf{pa}_X^G)$. These conditional probabilities are specified in a **conditional probability table** for variable X or CP-table. We write $P(X_1 = x_1, ..., X_n = x_n) = p$, sometimes abbreviated as $P(x_1, ..., x_n) = p$, to denote that the joint probability of random variables $X_1, \ldots, X_n$ taking on values $x_1, \ldots, x_n$ is p. We also use vector notation $P(\mathbf{X} = \mathbf{x}) = p$.

We assume a standard **relational schema** containing a set of tables, each with key fields, descriptive attributes, and possibly foreign key pointers. A **database instance** specifies the tuples contained in the tables of a given database schema. We assume that tables in the relational schema are divided into *entity tables* and *relationship tables.* This is the case whenever a relational schema is derived from an entity-relationship model (ER model) [13, Ch.2.2]. The symbol E refers to entity tables, and the symbol R refers to relationship tables. Table 1 shows a relational schema for a university domain. A field or attribute named $name$ in table T is denoted by $T.name$. Each attribute has a domain of values denoted by $dom(T.name)$. The number of tuples in a table T for a database instance $\mathcal{D}$ is written as $|T|_{\mathcal{D}}$. We view a descriptive attribute of an entity table E as a deterministic function of an entity from E, and a descriptive attribute of a relationship table R as a deterministic function of entities linked by R. The relationship R itself can be viewed as a Boolean function that indicates for each entity tuple of the appropriate type whether it is linked by R. The **natural join** of two tables is the set of tuples from their cross product that agree on the values of fields common to both tables.

3 Join Bayes Nets and the Attribute-Relation Table

A Join Bayes net contains a node for each attribute field in the database, and a Boolean indicator node for each relationship table. The definition assumes that a given basic entity table is referenced at most once in a given relationship table. A generalization for the case in which entity sets may be related to themselves is treated in [12].

Definition 1 *A **Join Bayes Net** (JBN) structure for a*

database schema with entity tables and relationship tables is a DAG G with one node for each descriptive attribute $A.name$ in the database, whose domain is $dom(A.name)$, and one binary node for each relationship table in the database.

We adopt the following functional *notation for the variables in a JBN*. We use a mnemonic Roman letter, e.g. V, to refer to a given entity table (e.g., S for the $Student$ table, C for the $Course$ table). An entity attribute node for the table is denoted by $name(V)$ (e.g., $ranking(S)$). The node for a descriptive attribute $R.name$ of a relationship table is denoted by $name(V_1, \ldots, V_k)$ where $V_1, \ldots, V_k$ refers to the entity tables linked to R by foreign key constraints (e.g., $grade(S, C)$). Similarly, the indicator node for R is denoted by $R(V_1, \ldots, V_k)$ (e.g., $Registered(S, C)$). Figure 1(e) shows a JBN for the university schema with this notation.

We associate with a given database $\mathcal{D}$ a joint distribution $P_{\mathcal{D}}$ over relationships and descriptive attributes, which is defined by a new join table—called the **attribute-relation table**—that is constructed as follows.

1. Form the cross product of all entity tables.
2. Extend the table with descriptive attributes of the relationship tables and one additional Boolean field for each relation. The boolean field for relationship table R takes the value T when the relationship R holds for the corresponding entity tuple and takes on the value F otherwise. When R is true for an entity tuple, the descriptive attributes of R are filled in with the corresponding values.
3. When R is false for an entity tuple, the descriptive attributes of R are assigned the value $\perp$ for "undefined".
4. Remove the primary key columns.

The attribute-relation table is viewed as a regular data table whose row frequencies represent a joint distribution over its columns, which is the **database distribution** P_D. Figure 1(d) shows the attribute-relation table for a small instance of the university schema.

Discussion The database distribution is closely related to joins as expressed in Datalog-style query languages like the DRC [13]. In logic queries, a table join corresponds to a conjunction; for instance, the join of the Registration table with the Student table selecting courses with $rating = 2$ is expressed by the query formula $\langle S, C : Registered(S, C), rating(C) = 2\rangle$. The probability assigned to this conjunction by the database distribution is the size of the join result in the database that corresponds to the conjunction, divided by the maximum size of the join result given the foreign key constraints:

$$P_{\mathcal{D}}(Registered(S, C) = T, rating(C) = 2) = \frac{|\langle S, C : Registered(S, C), rating(C) = 2\rangle|_{\mathcal{D}}}{|Student|_{\mathcal{D}} \times |Course|_{\mathcal{D}}} \quad (1)$$

Equation (1) illustrates that the probabilities assigned by the attribute-relation table have a natural alternative interpretation. It also implies that from an estimation of the database distribution $P_{\mathcal{D}}$, we can readily compute an estimate of join sizes, which is an important application for query selection.

We define the database distribution over the full cross product of the entities rather than just the join of entities with relationship tables. In relationship tables, entities with more links appear more frequently than others. As a result, the probability of an attribute value derived from the join of relationships with entities may not reflect the real statistical information. For instance, in Figure 1, in the join of the $Student$ table with the $Registered$ table the frequency of rows with $rating = 2$ is $1/2$, whereas the frequency of rows in the $Course$ table with $rating = 2$ is $1/3$. This is one of the problems often raised for basing statistical learning on a join table. The attribute relation table overcomes the problem by using the cross product of entities, so all entities from a given table appear in the same number of rows regardless of how many links they have. The subset of rows of the attribute-relation table in which the indicator variable $R = T$ corresponds to the join of the entity tables with the relationship table R. We now consider learning a JBN model for a given database distribution.

4 Parameter Estimation with a Virtual Join Algorithm

This section treats the problem of computing conditional frequencies in the database distribution, which corresponds to computing sample frequencies in the single table case. The main problem is computing probabilities conditional on the *absence* of a relationship. For instance, to compute $P_{\mathcal{D}}(difficulty(C) = 2 | intelligence(S) = 3, Registered(S, C) = T)$, a frequency count on the constraints given by the query is done on the join of the $Registered$, $Student$, and $Course$ tables. However, computing conditional probabilities on queries with false relationships (e.g., $Registered(S, C) = F$) raises difficulties because it involves non-existent links (cf. [5]). This problem arises because a JBN includes relationship indicator variables such as $Registered(S, C)$, and building a JBN therefore requires modelling the case where a relationship does not hold. In principle, frequencies in the database conditional on the absence of links can be computed with frequency counts over the rows in the attribute-relation table where the link is absent. However, because materializing this table is generally not feasible, we instead use a *virtual join* algorithm that computes the frequencies in the entity join table without actually constructing the entity join. The virtual join algorithm is a dynamic programming algorithm for estimating joint probabilities in a database instance whose database operations involve only: (1) Joins of existing relationship tables with entity tables, and (2) Joins of existing relationship tables with other existing relationships tables that share an entity type (foreign key pointer). Relationship tables, such as $Registered$, are typically much smaller than the cross product of their related entities [5], so the join operations (1) and (2) are feasible for SQL queries, and our algorithm is much more efficient than explicitly constructing the attribute relation table.

Virtual Join Algorithm: Outline and Example Our algorithm computes joint probabilities. Conditional probabilities can easily be computed from joint probabilities via the equation $P(x|y) = P(x, y)/\sum x' P(x', y)$ where the summation is taken over all possible values of x. The basic idea can be described as follows. From probability laws, we have the relation

$$P(\mathbf{x}, R = F) = P(\mathbf{x}) - P(\mathbf{x}, R = T). \quad (2)$$

Equation (2) shows how we can reduce a probability involving a nonexistent relationship $R = F$ to two other computations that do not involve the nonexistent relationship: (1) the case in which we do not condition on the value of R, and (2) the case in which we condition on $R = T$. Let us consider first the case in which the joint probability involves only a single relationship variable together with descriptive attributes of entities (cf. [5, Sec.5.8.4.2]). In that case, the probability $P(\mathbf{x}, R = T)$ can be obtained from a frequency

(a)

Student		
s-id	Intelligence	Ranking
Jack	3	1
Kim	2	1
Paul	1	2

(b)

Registration			
s-id	C.id	Grade	Satisfaction
Jack	101	A	1
Jack	102	B	2
Kim	102	A	1
Kim	103	A	1
Paul	101	B	1
Paul	102	C	2

(c)

Course					
c-id	Prof	Rating	Difficulty	T-a-prof	P-prof
101	Oliver	3	1	1	3
102	David	2	2	1	2
103	Oliver	3	2	1	3

(d)

attribute-relation table								
Intelligence	Rank	Rating	Diff	P-prof	t-a-prof	Reg(S,C)	R.Grade	R.Sat
3	1	3	1	3	1	T	A	1
3	1	2	2	2	1	T	B	2
3	1	3	2	3	1	F	⊥	⊥
2	1	3	1	3	1	F	⊥	⊥
2	1	2	2	2	1	T	A	1
2	1	3	2	3	1	T	A	1
1	2	3	1	3	1	T	B	1
1	2	2	2	2	1	T	C	2
1	2	3	2	3	1	F	⊥	⊥

(e)

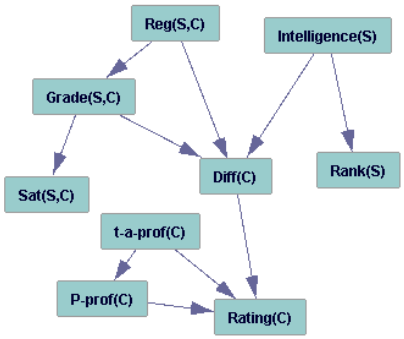


Figure 1: Database Table Instances: (a) *Student*, (b) *Registered* (c) *Course*. To simplify, we added the information about professors to the courses that they teach. (d) The attribute-relation table is formed in two steps: (1) take the cross product of the student and course table (3 x 3 = 9 rows) and extend it with the matching attribute and relationship information. (2) Remove the primary entity keys from the cross product of the entities. (e) A Join Bayes Net for the university schema variables.

count in the relationship table R in the database. The probability $P(\mathbf{x})$ may involve descriptive attributes from more than one entity table. It can be computed using the fact that distinct entity tables are independent, unless they are linked by a relationship variable [12], so the joint probability $P(\mathbf{x})$ is calculated by multiplying frequencies from entity tables.

Inductively, consider a joint probability involving $m > 0$ false relationships $R^1 = F, .., R^m = F$. Then first, change one of the false relationships to be true, e.g., $R^1 = T$, and compute the joint probability for this case recursively, since it involves one less false relationship. Second, change the state of the chosen relationship to be unspecified, e.g., $R^1 = \mathit{unspecified}$, and compute the conditional probability for this case recursively, since it involves one less false relationship. In our dynamic program, frequencies with fewer false relationship variables are computed first, so the two frequencies can be looked up from the previous computations.

Example. Figure 2 shows how to compute a joint probability with exactly one false relationship for the database instance of Figure 1. To illustrate the case with multiple relationships, suppose the university schema features another entity table $TA(\underline{ta_id}, \mathit{expertise})$ to record the expertise of teaching assistants and another relationship table relation $\mathit{Assigned}(\underline{ta_id}, \underline{course_id})$ to record which assistants are assigned to which course. Figure 2 shows how the computation of a joint probability with two false relationships can be reduced to two probabilities, each without the false relationship $\mathit{Assigned}(TA, C) = F$. [12] provides further implementation details, including pseudocode and complexity analysis. In the next section we apply the parameter estimation algorithm to build Join Bayes nets for three relational datasets.

5 Evaluation and Experiments

We present results of applying our learning algorithms to three relational data sets, the MovieLens and Financial real-world databases, and an artifical University database for the schema given in 1. Our evaluation method comprises the following steps.

1. Learn a JBN structure for each database. For comparison, we also apply a standard structure learning algorithm for Markov Logic Networks to each database.
2. Fill in the CP-tables with maximum likelihood estimates.
3. Apply a standard Bayes net inference algorithm to estimate conditional frequencies in the database, and compare the estimates to the result of directly computing conditional frequencies with SQL queries.

5.1 System Resources, Algorithms and Datasets

Our implementation used many of the procedures in version 4.3.9-0 of CMU's Tetrad package [2]. Our Java code is available from the senior author upon request. All experiments were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM.

Learning Algorithms A description of our structure learning method is beyond the scope of this note, but is provided in [12]. Our method is modular in that it upgrades *any* propositional single-table BN learner to a JBN learner. We used the Tetrad implementation of GES search [1] with the BDeu score (uniform structure prior, ESS=8) as the base single-table BN learning program. After learning a JBN structure, parameter estimation is carried out using the algorithm described in the previous section.

Inference Algorithms JBN inference was carried out with Tetrad's Rowsum Exact Updater algorithm. A direct comparison of class-level inference with other SRL formalisms is difficult as the implementations we could find support only instance-level queries. For example, both the Alchemy package for MLNs [3] and the Balios BLN engine [9] support only queries with ground atoms. We could not obtain source code for PRM inference.

Data sets Our datasets are available on-line at ftp://ftp.fas.sfu.ca/pub/cs/oschulte/datasets/.

University Database. In order to check the correctness of our algorithms directly, we manually created a small data set, based on the schema given in 1. The entity tables contain 38 students, 10 courses, and 6 Professors. The *Registered* table has 92 rows and the RA table has 25 rows.

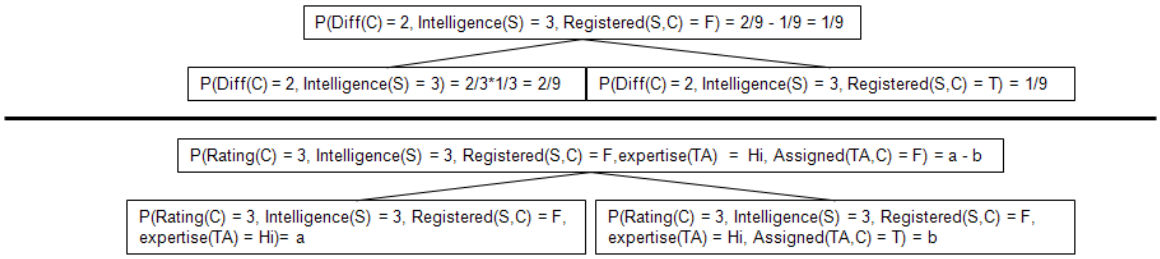


Figure 2: To illustrate the recursive scheme of our parameter estimation algorithm. The top example for the database instance in Figure 1 reduces the computation of a joint probability involving one false relationship to two without any false relationship indicators. The bottom example shows for a generic database instance how the computation of a joint probability involving two false relationships can be reduced to two with just one false relationship each.

MovieLens Database. The second data set is the MovieLens data set from the UC Irvine machine learning repository. It contains two entity tables: *User* with 941 tuples and *Item* with 1,682 tuples, and one relationship table *Rated* with 100,000 ratings. The *User* table has 3 descriptive attributes *age*, *gender*, *occupation*. We discretized the attribute age into three bins with equal frequency. The table *Item* represents information about the movies. It has 17 Boolean attributes that indicate the genres of a given movie; a movie may belong to several genres at the same time. For example, a movie may have the value *T* for both the *war* and the *action* attributes. The full table with 100,000 ratings exceeded the memory limits of Tetrad, so we randomly picked 40% of the ratings of the relationship table as input data.

Financial Database. The third data set is a modified version of the financial data set from the PKDD 1999 cup. We adapted the database design to fit the ER model. We have two entity tables: *Client* with 5369 tuples and *Account* with 4,500 tuples. Two relationship tables, *CreditCard* with 5,369 tuples and *Disposition* with 892 tuples relate a client with an account. The Client table has 10 descriptive attributes: the client's age, gender and 8 attributes on demographic data of the client. The Account table has 3 descriptive attributes: information on loan amount associated with an account, account opening date, and how frequently the account is used.

5.2 Experimental Results

We evaluate structure learning, parameter estimation and inference with Join Bayes nets. Table 2 presents a summary of the run time for parameter learning and structure learning for the data sets. The computation times are well within the range of practical feasibility (40 min for the most difficult experiment).

Learning The graphs learned are shown in Figures 1, 3, and 4. In the MovieLens data set, the algorithm finds a number of cross-entity table links involving the age of a user. Because genres have a high negative correlation, the algorithm produces a dense graph among the genre attributes. We simplified the graph by omitting genre variables that have only indirect links with the rating or User attributes. The richer relational structure of the Financial data set is reflected in a more complex graph with several cross-table links. The birthday of a customer (translated into discrete age levels) has especially many links with other variables.

The university database is small enough to materialize its attribute-relation table and verify the correctness of our parameter estimates directly. For the larger databases, this is not feasible. The next section provides an indirect way to check the learning algorithms by comparing the probabilities estimated by the JBN with the database frequencies computed directly from SQL queries.

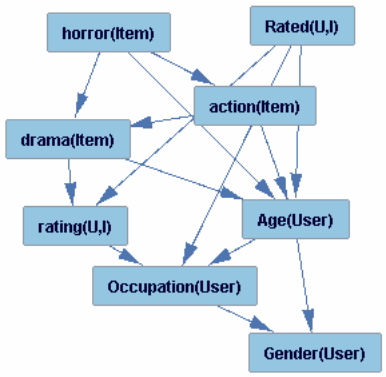


Figure 3: The JBN structures learned by our merge learning algorithmfor the MovieLens Data set.

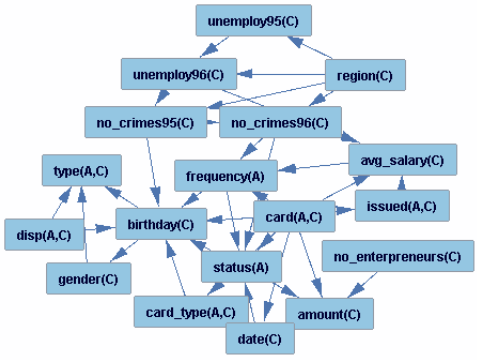
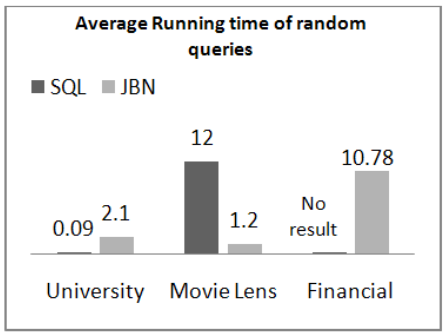


Figure 4: The JBN structures learned by our merge structure learning algorithmfor the Financial Data set.

Inference To avoid bias, we randomly generated 10 queries, each involving 4 nodes, for each data set according to the following procedure. We compared the probabilities predicted by the JBN with the frequencies in the database

Data set	PL	SL in JBN
University	0.495	0.64
Movie Lens	2,018	135
Financial	2,472	574

Table 2: The run times—in seconds—for structure learning (SL) and parameter learning (PL) on our three data sets.



DataSet	University	MovieLens	Financial
Average Probability difference	0.003	0.027	N/A

Figure 5: Comparing the probability estimates and run times from the learned JBN models with SQL queries. Not all SQL queries for the Financial data set terminated with a result. The average is taken over 10 randomly generated queries.

as computed by an SQL query, as well as the run times for computing the probability using the JBN vs. the SQL. We do not expect the probabilities predicted by a JBN to be exactly the same as the data frequencies, for the same reason that in the single table case a BN learner would not just reproduce the sample frequencies: the absence of links in the graph entails probabilistic independence between variables that may be slightly correlated in the data. But since we use maximum likelihood estimates, and our sample sizes are not small, we would expect the predicted probabilities to be fairly close to the sample frequencies if the JBN structure is adequate. This expectation is confirmed by our results: we see in Figure 5 that the predicted probabilities are close to the data frequencies. For the small university data set, SQL queries are faster than JBN inference. But for the larger MovieLens data set, model inference is much faster, and for the largest Financial data set, SQL queries were infeasible when conditioning on the absence of relationships, whereas the JBN returns an answer in around 10 seconds. Where the SQL queries did return a frequency, it was close to the JBN estimate.

We observed that the number of tuples in the database table is a very significant factor for the speed of SQL queries but does not affect JBN inference. This is an important observation about the *data scalability of JBN inference*: While the learning algorithms depend on the size of the database, once the learning is completed, query processing is independent of database size. So for applications like query optimization that involve many calls to the statistical inference procedure, the investment in learning a JBN model is quickly amortized in the fast inference time.

6 Conclusion

We showed how Join Bayes nets can be used to represent class-level dependencies between attributes of entities or relationships. This contrasts with instance-level dependencies between attributes of specific entities. Class-level generic dependencies are of interest in themselves, and they support applications like policy making, strategic planning, and query optimization. We defined a new semantics for class-level Bayes nets based on a new database join operation. The focus on class-level dependencies brings gains in tractability of learning and inference. We described efficient and scalable algorithms for structure and parameter estimation in Join Bayes nets. Inference can be carried out with standard algorithms "as is". An evaluation of our methods on three data sets shows that our algorithms are computationally feasible for realistic table sizes, and that the learned structures represented the statistical information in the databases well. After learning has compiled the database statistics into a Join Bayes net, querying these statistics via the net is faster than directly with SQL queries, and does not depend on the size of the database.

References

[1] David Maxwell Chickering and Christopher Meek. Finding optimal bayesian networks. In *UAI*, pages 94–102, 2002.

[2] The Tetrad project: Causal models and statistical data, 2008. http://www.phil.cmu.edu/projects/tetrad/.

[3] Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning* [8].

[4] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[5] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In *Introduction to Statistical Relational Learning* [8].

[6] Lise Getoor and Ben Taskar. Introduction. In Getoor and Taskar [8], pages 1–8.

[7] Lise Getoor, Benjamin Taskar, and Daphne Koller. Selectivity estimation using probabilistic models. *ACM SIGMOD Record*, 30(2):461–472, 2001.

[8] Lise Getoor and Ben Tasker. *Introduction to statistical relational learning*. MIT Press, 2007.

[9] Kristian Kersting and Luc De Raedt. Bayesian logic programming: Theory and tool. In *Introduction to Statistical Relational Learning* [8].

[10] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[11] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kauffmann, 1988.

[12] Oliver Schulte, Hassan Khosravi, Flavia Moser, and Martin Ester. Join bayes nets: A new type of bayes net for relational data. *CS-Learning Preprint Archive*, http://arxiv.org/abs/0811.4458, 2008.

[13] J. D. Ullman. *Principles of database systems*, volume 2. Computer Science Press, 1982.

Hierarchy Analysis of Knowledge Networks

Cliff Joslyn
Chief Scientist for Knowledge Sciences
Pacific Northwest National Laboratory
cjoslyn@pnl.gov

Abstract

Knowledge systems technologies are dominated by graphical structures such as ontologies, semantic graph databases, and concept lattices. A critical but typically overlooked aspect of all of these structures is their admission to analyses in terms of formal hierarchical relations. Transitivities of network links necessarily result in hierarchical levels, whether explicitly within directed acyclic graphs (DAGs) or implicitly through the identification of cycles. And whether from transitive link types in semantic graphs, or the explicit lattice structures of Formal Concept Analysis, the partial order representations of whatever hierarchy is present within a knowledge structure afford opportunities to exploit these hierarchical constraints to facilitate a variety of tasks, including ontology analysis and alignment, visual layout, and anomaly detection. In this short survey paper we introduce the basic concepts involved and address the impact of a hierarchical (order-theoretical) analysis on directed acyclic graphs in knowledge systems tasks.

1 Introduction

Knowledge systems technologies are dominated by graphical structures, including:

- **Semantic graph databases** [19] take the form of labeled directed graphs implemented in RDF[1]. Their OWL[2] ontological typing systems are also labeled directed graphs, frequently dominated by directed acyclic graph (DAG) and other hierarchical structures. Fig. 1 shows a toy example, where the ontology of classes on the left forms the typing system for the semantic graph of node and link instances on the right.
- **Concept lattices** [10; 11; 17] are hierarchical lattice structures derived from identifying the maximal connections among groups of objects and properties (rows and columns) of an attribute matrix, called a formal context. Fig. 2 shows a simple example from [11], indicating semantic generality of the attributes in terms of the number of their shared objects, and *vice versa.*

While other examples of graph-based knowledge structures abound, what characterizes these structures in particular is their hierarchical nature. There is an increasing emphasis on hierarchical structure in network science [4; 5], but these methods partition the set of nodes of an underlying simple (undirected) graph to produce a hierarchical decomposition. We are interested rather in the *intrinsic* hierarchical (level-based) nature of an underlying *directed* graph.

A good example is our concept lattice in Fig. 2, which is an explicit hierarchy in its entirety, as are semantic taxonomies such as the Gene Ontology [2] (GO[3]). But where OWL ontologies include hierarchical class structures, other portions can be non-hierarchical. And more general knowledge structures like semantic graphs are not explicitly or necessarily hierarchical, but may contain large hierarchical components.

In practice, ontologies are dominated by their "hierarchical cores", specifically their class hierarchies connected by `is-a` subsumptive and `has-part` compositional links. And many of the most common links in RDF graphs are transitive, including `causes`, `implies`, and `precedes`. We will show in Sec. 3 below that any transitive link yields a hierarchical structure in terms of the connectivity of its strongly connected components, and is thus amenable to a hierarchical analysis.

Whether from transitive link types in semantic graphs, or the explicit lattice structures of concept lattices, the partial order representation of whatever hierarchy is present within a knowledge structure affords opportunities to exploit these hierarchical constraints for a variety of tasks, including

Clustering and Classification: Including characterizing a portion of a hierarchy (e.g. groups of ontology nodes) to identify common characteristics [15; 23],

Alignment: Casting ontology matching [8][4] as mappings between hierarchical structures [13; 14].

Induction from Source Data: For example using concept lattices to induce ontologies from textual relations [17].

Visualization: Including exploiting the level structure of hierarchies to achieve a satisfactory layout [16].

In general, such a hierarchical analysis, when available, promises complexity reduction, improved user interaction

[1] http://www.w3.org/RDF
[2] http://www.w3.org/TR/owl-features
[3] http://www.geneontology.org
[4] http://www.ontologymatching.org

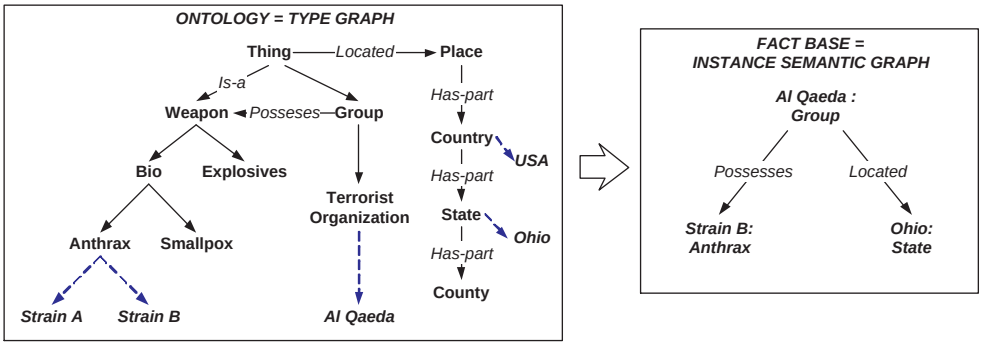


Figure 1: Toy model of a semantic graph database. (Left) Ontological typing system as a labeled, directed graph of classes (sample instances shown below dashed links). (Right) Conforming instance sub-graph.

with the knowledge base, and improved layout and visual analytics. In the remainder of this short survey paper we explicate the basic concepts referred to here and draw connections among these application areas.

2 DAGs and Partial Orders

Mathematically, hierarchies are represented as partially ordered sets (posets), which are reflexive, anti-symmetric, and transitive binary relations $\mathcal{P} = \langle P, \leq \rangle$ on an underlying finite set of nodes P [7]. While we typically think of hierarchies as tree structures, more general kinds of hierarchies have "multiple inheritance", where nodes can have more than one parent. These include lattice structures like the concept lattice in Fig. 2, where pairs of nodes have unique least common subsumers (and unique greatest lower bounds as well); partial orders where pairs of nodes can have an indefinite number of least common subsumers and greatest lower bounds; and finally general DAGs can also include "transitive links" which form shortcuts across paths.

Consider simple DAG in the top of Fig. 3. The two transitive links $1 \to H, 1 \to E$ connect the two paths $1 \to K \to H$ and $1 \to C \to I \to E$ respectively. Given a DAG $\mathcal{D}$, the DAG $\mathcal{P}(\mathcal{D})$ produced by including all possible transitive links consistent with its paths is its **transitive closure**, and determines an ordered set $\mathcal{P}(\mathcal{D}) = \langle P, \leq \rangle$ where $a \leq b \subseteq \mathcal{P}$ if there is a directed path from a to b in $\mathcal{D}$. The graph $\mathcal{V}(\mathcal{D})$ produced from a DAG $\mathcal{D}$ by removing all its transitive links (its transitive reduction [1]) determines a **cover relation** or **Hasse diagram**. Thus each cover relation $\mathcal{V}$ determines a unique poset $\mathcal{P}(\mathcal{V})$, and *vice versa* a poset $\mathcal{P}$ determines a unique cover $\mathcal{V}(\mathcal{P})$; each DAG $\mathcal{D}$ determines a unique poset $\mathcal{P}(\mathcal{D})$ and cover $\mathcal{V}(\mathcal{D})$; and each unique poset-cover pair determines a class of DAGs equivalent by transitive links.

For a DAG $\mathcal{D}$ we can measure its **degree of transitivity** as

$$TR(\mathcal{D}) := \frac{|\mathcal{D} \setminus \mathcal{V}(\mathcal{D})|}{|\mathcal{P}(\mathcal{D}) \setminus \mathcal{V}(\mathcal{D})|},$$

where $\setminus$ is set subtraction, we interpret each structure as the binary relation on P^2 of its incidence matrix, and $|\cdot|$ is cardinality, so that $|\cdot|$ is the number of links in $\cdot$, seen as a graph. $TR(\mathcal{D})$ measures the number $|\mathcal{D} \setminus \mathcal{V}(\mathcal{D})|$ of transitive links in $\mathcal{D}$ relative to the total possible number $|\mathcal{P}(\mathcal{D}) \setminus \mathcal{V}(\mathcal{D})|$ in its transitive closure $\mathcal{P}(\mathcal{D})$. In Fig. 3 we have $TR(\mathcal{D}) = \frac{2}{11}$, indicating a relatively low degree of transitivity.

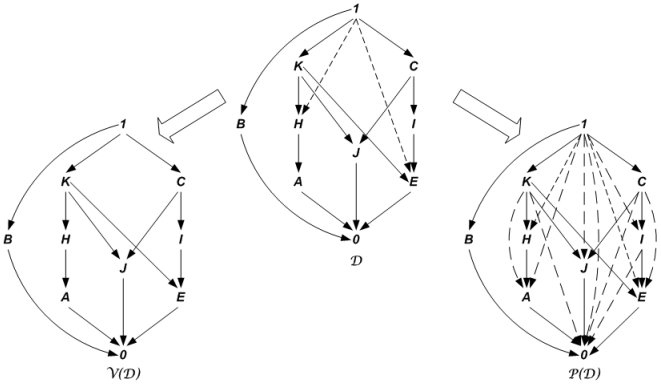


Figure 3: (Top) A DAG $\mathcal{D}$. (Left) Transitive reduction $\mathcal{V}(\mathcal{D})$. (Right) Transitive closure $\mathcal{P}(\mathcal{D})$.

In knowledge systems such as ontologies, our interpretation of the presence or absence of transitive links in DAGs is significant. If the link-type in question is anti-transitive, so that transitive links are disallowed, then clearly the presence of transitive links is in error. If, on the other hand, the link-type in question is atransitive, so that transitive links are allowed, but not required, then the $TR(\mathcal{D})$ measures this extent. But finally, if, as is the case with our subsumption and composition types, the link type represents a fully transitive property, then the presence of transitive links are irrelevant or erroneous. Effectively, such link types live in the trasitively equivalent class of DAGs, that is, in the partial order $\mathcal{P}(\mathcal{D})$, and $TR(\mathcal{D})$ can be used as an aid to the user or engineer to identify issues with the underlying ontology.

3 The Hierarchical Cores of Directed Graphs

So central to the consideration of hierarchy in a knowledge network is the question of the presence and prevalence of DAGs in general directed graphs which can possibly have cycles. Consider the network shown in Fig. 4 using a standard network layout with a primarily radial link distribution around centralized nodes. We can analyze this network as:

		a	b	c	d	e	f	g	h	i
1	Leech	×	×					×		
2	Bream	×	×					×	×	
3	Frog	×	×	×				×	×	
4	Dog	×		×				×	×	×
5	Spike – weed	×	×		×		×			
6	Reed	×	×	×	×		×			
7	Bean	×		×	×	×				
8	Maize	×		×	×		×			

Context of an educational film "Living Beings and Water". The attributes are: a: needs water to live, b: lives in water, c: lives on land, d: needs chlorophyll to produce food, e: two seed leaves, f: one seed leaf, g: can move around, h: has limbs, i: suckles its offspring.

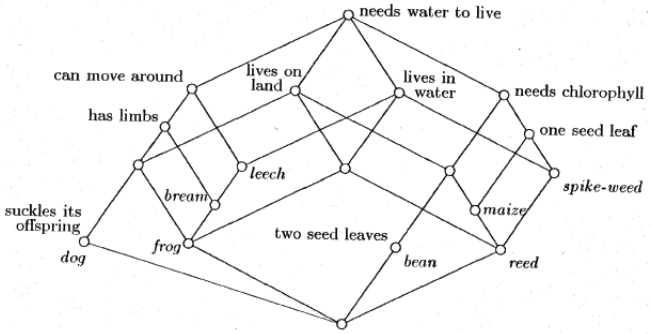


Figure 2: (Left) A formal context of objects and their properties. (Right) Its concept lattice. From [11].

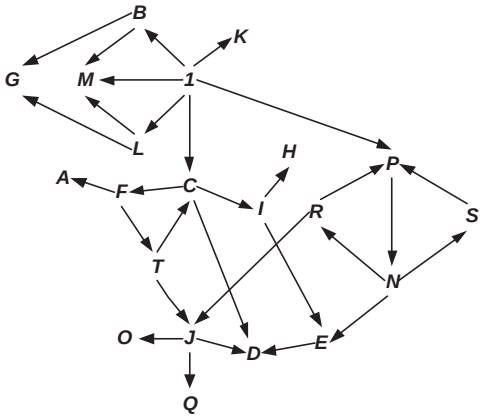


Figure 4: A network.

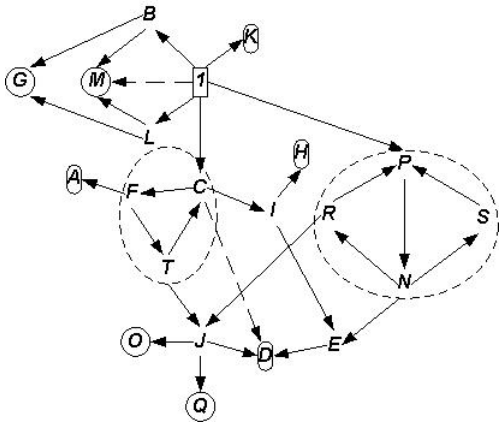


Figure 5: Components of the network's cyclic decomposition: roots are squares, leaves are oblongs, transitive links are dashed arrows, and strongly connected components are shown in dashed circles.

- Identify the leaves as those nodes $\{A, D, G, H, K, M, O, Q\}$ with no children.
- Identify the roots as those nodes $\{1\}$ with no parents.
- Identify the strongly connected components (SCCs [21]) as those sets of nodes which are directed cliques, where a directed path exists between all pairs on nodes. Each SCC is either a single directed cycle or a union of directed cycles, and are necessarily disjoint from each other: if two SCCs intersect, they'd form a single SCC together. In the example, there are two SCCs, $X = \{C, F, T\}$ (a single 3-cycle) and $Y = \{P, R, S, N\}$ (the union of the two 3-cycles $\{N, R, P\}$ and $\{N, S, P\}$).
- Identify the transitive links, these are the two links $1 \rightarrow M$ shortcutting the 3-paths $1 \rightarrow B \rightarrow M$ and $1 \rightarrow L \rightarrow M$, and the link $C \rightarrow D$ shortcutting the 4-path $C \rightarrow I \rightarrow E \rightarrow D$.

Fig. 5 shows our network with these components identified.

We proceed by contracting each SCC to a new node in a higher-order space, combining any multiple links between SCCs into one. The resulting structure is necessarily a DAG $\mathcal{D}$. Finally, we derive the transitive reduction of the cyclic decomposition to eliminate the transitive links, measuring $TR(\mathcal{D})$. The resulting hierarchy is shown in Fig. 6, where the two SCCs are replaced with new nodes X and Y respectively. While this structure is substantially similar, of course these two new nodes will be identified as being new meta-nodes, and available for "double-clicking" to open up the SCCs revealing the original structure below.

While such an approach to a "cyclic decomposition" is a known standard for directed graphs [6], it is not usually applied for hierarchical anlaysis. Note that the layout is adjusted to bring the root to the top and the leaves to the bottom, and to emphasize the stratified nature of the hierarchy. This concept of "level" or rank is central to our approach, and will be dealt with in Sec. 4.1 below.

In addition to deriving the base cyclic decomposition, we are also interested in some numerical quantities such as the number of roots, leaves, and SCCs, along with the spectrum of sizes of the SCCs. Depending on the results of these measurements, it may or may not be appropriate to proceed with a hierarchical analysis of the network.

In particular, for a DAG $\mathcal{D}$ produced from an underlying directed graph $\mathcal{G}$, we can measure the degree of cyclicity as

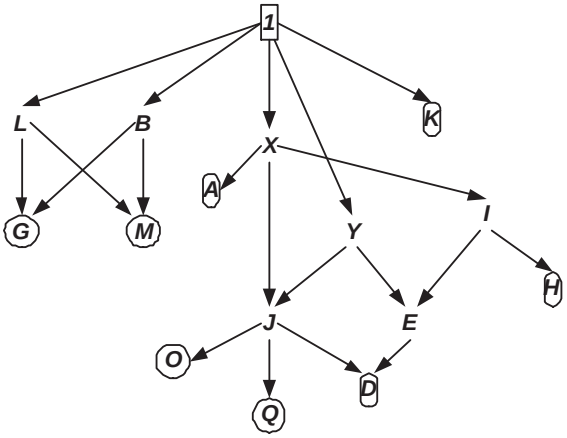


Figure 6: The network's cyclic decomposition displayed as a hierarchy.

$C(\mathcal{G}) := |\mathcal{D}|/|\mathcal{G}|$. As two extremes, $C(\mathcal{G}) = 1/|\mathcal{G}|$, so that the network consists of a single, large SCC; or, if $C(\mathcal{G}) = 1$, then the network was already a DAG at the outset, and no SCCs are observed. In either event, continuing with a hierarchical analysis would not be fruitful. But if there are a number of moderately sized SCCs, then the resulting hierarchical structure will provide greater simplification and a stratified view of the underlying complex structure.

4 Measures on Hierarchical Graphs

Given a hierarchical structure as a DAG represented by its transitive closure poset $\mathcal{P}$, perhaps derived as the cyclic decomposition of a network, or provided natively as in a taxonomic ontology, we now have a number of tools available to measure this hierarchical structure. Here we discusses **interval-valued rank** measuring the vertical level of nodes, and **order metrics** measuring the distances between nodes. See [12; 16; 18] for more details.

4.1 Interval-Valued Rank

Given a hierarchy e.g. in Fig. 6, we are concerned with the proper representation of the vertical level of each node, as represented by its positioning in a layout. We note that all children of the root are the same "distance" from the root, but if these are *also* leaves then they should be positioned further down. In other words, we need to exploit the vertical distance from *both* the top *and* a global bottom, in this case a virtual node $0 \in P$ we can insert and place below all the leaves.

For $a, b \in P$, let $h^*(a, b)$ be the length of the maximum path from a to b. Then the distance of a node $a \in P$ from the root $1 \in P$ is the **top rank** $r^t(a) := h^*(a, 1)$. Dually we define the **bottom rank** $r^b(a) := h^*(0, 1) - h^*(0, b)$, where $h^*(0, 1)$ is the overall **height** of the structure. Then the **interval rank** $\bar{R}(a) := [r^t(a), r^b(a)]$ becomes available as an interval-valued measure of the vertical levels over which a can range, while the **rank width** $W(a) := r^b(a) - r^t(a)$ is a measure of that range [16; 18].

We can exploit this vertical rank in terms of hierarchical layout and visualization, as shown for our example now in Fig. 7. Each node which sits on a complete chain (a path from 1 down to 0) of maximal size is placed horizontally at the center of the page. Nodes are laid out horizontally according to the size of their largest chains maximal chains. The result it to place maximal complete chains along a central axis, and short complete chains towards the outer edges. Nodes are placed vertically according to the mathematical quantity of the midpoint of their interval rank, but can be free to move between top rank $r^t(a)$ and bottom rank $r^b(a)$.

The result is that while nodes on maximal complete chains (all those intersecting the chain $0 \to D \to E \to I \to X \to 1$ in the example) exist at a single level, some (for example K) do not. While Fig. 7 shows a 2D layout, we have also deployed this concept in a 3D layout [16].

4.2 Order Metrics

Given the need to perform operations like clustering or alignment on ontologies represented as ordered sets $\mathcal{P} = \langle P, \leq \rangle$, it is essential to have a general sense of distance $d(a, b)$ between two nodes $a, b \in P$. The knowledge systems literature has focused on **semantic similarities** to perform a similar function, which are available when $\mathcal{P}$ is equipped with a probability distribution, derived, for example, from the frequency with which terms appear in documents (for the Wordnet[5] [9] thesaurus), or genes are annotated to GO nodes.

So assume a poset $\langle P, \leq \rangle$ with a base probability distribution $p\colon P \to [0, 1], \sum_{a \in P} p(a) = 1$, and a "cumulative" function $\beta(a) := \sum_{b \leq a} p(a)$. We then generalize the join (least upper bound) and meet (greatest lower bound) operations in lattices as follows. Let $\uparrow a := \{b \geq a\}$ and $\downarrow a := \{b \leq a\}$ are the up-set (filter) and down-set (ideal) respectively of a node $a \in P$. Then for two nodes $a, b \in P$, let $a \nabla b := \uparrow a \cap \uparrow b$ and $a \Delta b := \downarrow a \cap \downarrow b$ be the set of nodes above or below respectively both of them. Then the generalized join $a \vee b$ is the set of minimal (lowest) nodes of $a \nabla b$, and the generalized meet $a \wedge b$ is the set of maximal (highest) nodes of $a \Delta b$. When $\mathcal{P}$ is a lattice, then $|a \vee b| = |a \wedge b| = 1$, recovering traditional join and meet.

Traditional choices for the semantic similarity $S(a, b)$ between two nodes then include the measures of Resnik, Lin, and Jiang and Conrath [3]:

$$S(a, b) = \max_{c \in a \vee b} [-\log_2(\beta(c))]$$

$$S(a, b) = \frac{2 \max_{c \in a \vee b} [\log_2(\beta(c))]}{\log_2(\beta(a)) + \log_2(\beta(b))}$$

$$S(a, b) = 2 \max_{c \in a \vee b} [\log_2(\beta(c))] - \log_2(\beta(a)) - \log_2(\beta(b))$$

respectively. But most of these are not metrics (not satisfying the triangle inequality), and all of these lack a general mathematical grounding and require a probabilistic weighting.

Our approach uses ordered set metrics [20; 22] which can use, but do not require, a quantitative weighting such as β, and always yield a metric. They are based on valuation functions $v\colon P \to \mathbb{R}^+$ which are, first, either isotone ($a \leq b \to v(a) \leq v(b)$) or antitone ($a \leq b \to v(a) \geq v(b)$); and then semimodular, in that

$$v(a) + v(b) \sim v^{\nabla}(a, b) + v_{\Delta}(a, b),$$

[5] http://wordnet.princeton.edu

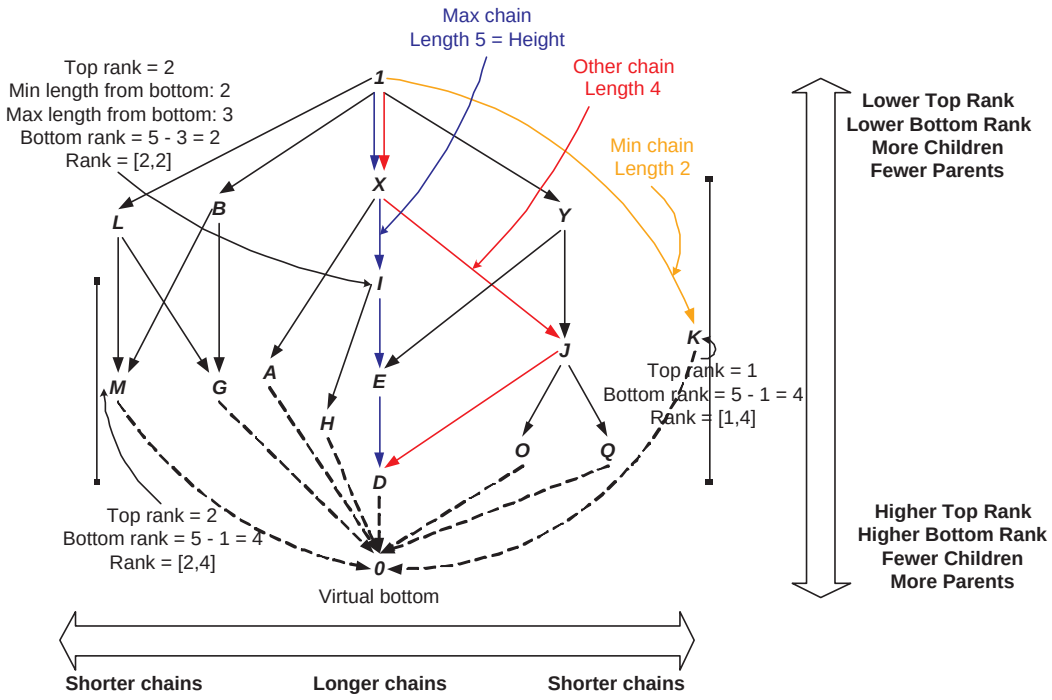


Figure 7: Chain layout of the cyclic decomposition of the network in Fig. 4.

where $\sim \in \{\leq, \geq, =\}$, yielding super-modular, sub-modular, and modular valuations respectively; and

$$v^{\nabla}(a,b) := \min_{c \in a \nabla b} v(c), \quad v_{\Delta}(a,b) := \max_{c \in a \Delta b} v(c).$$

Whether a valuation v is antitone or isotone, and then sub- or super-modular, determines which of four distance functions is generated, e.g. the antitone, supermodular case yields $d(a,b) = v(a) + v(b) - 2v^{\nabla}(a,b)$. When $\mathcal{P}$ is a lattice, then this simplifies to $d(a,b) = v(a) + v(b) - 2v(a \vee b)$.

Typical valuations v include the cardinality of up-sets and down-sets: $v(a) = |\uparrow a|, v(a) = |\downarrow a|$, and the cumulative probabilities used in semantic similarities $v(a) = \beta(a)$. In this way, poset metrics generalize semantic similarities and provide a strong basis for various analytical tasks.

5 Order Metrics in Ontology Alignment

A good example of the utility of this order theoretical technology in knowledge systems tasks is in ontology alignment [13; 14]. An ontology **alignment** is a mapping $f\colon \mathcal{P} \to \mathcal{P}'$ taking anchors $a \in P$ in one semantic hierarchy $\mathcal{P} = \langle P, \leq \rangle$ into anchors $a' \in P'$ in another $\mathcal{P}' = \langle P', \leq' \rangle$. In seeking a measure of the structural properties of the mapping f, our primary criterion is that f should not distort the metric relations of concepts, taking nodes that are close together and making them farther apart, or *vice versa.*

It should be noted that a "smooth" maping f is neither necessary nor sufficient to be a good alignment: one the one hand, a good structural mapping may be available between structures from different domains; and on the other, differences in semantic intent between the two structures may be irreconcilable. Nonetheless, other things being equal, it is preferable to have a more smooth mapping than not.

So, for two ontology nodes $a, b \in \mathcal{P}$, consider the **lower cardinality distance** $d_l(a,b) := |\downarrow a| + |\downarrow b| - 2 \max_{c \in a \wedge b} |\downarrow c|$. We can measure the change in distance between $a, b \in P$ induced by f as the **distance discrepancy**

$$\delta(a,b) := |\bar{d}_l(a,b) - \bar{d}_l(f(a), f(b))|,$$

where $\bar{d}_l(a,b) := \frac{d_l(a,b)}{\text{diam}_d(\mathcal{P})} \in [0,1]$ is the normalized lower distance between a and b in $\mathcal{P}$ given the diameter $\text{diam}_d(\mathcal{P}) := \max_{a,b \in P} d(a,b)$. We can measure the entire amount of distance discrepancy at a node $a \in P$ compared to all the other anchors $b \in P$ by summing

$$\delta_f(a) := \sum_{b \in P} \delta(a,b) = \sum_{b \in P} |\bar{d}_l(a,b) - \bar{d}_l(f(a), f(b))|,$$

yielding the discrepancy $\delta(f) := \sum_{a \in P} \delta_f(a)$ of the alignment.

Consider the example in Fig. 8, with the partial alignment function f as shown, mapping only certain nodes $\{B, E, G\}$ from $\mathcal{P}$ to $\mathcal{P}'$. Then we have e.g. the lower normalized distance between nodes E and G as $\bar{d}_l(E,G) = 1/3$; the distance discrepancy between the two nodes E, G in virtue of f as $\delta(E,G) = |1/3 - 3/5| = .267$; the entire distance discrepancy at the node E as $\delta_f(E) = 2/5$; and finally the distance discrepancy for the entire alignment as $\delta(f) = .47$.

6 Future Work

Our work continues across the range of tasks outlined here, and includes a number of future targets:

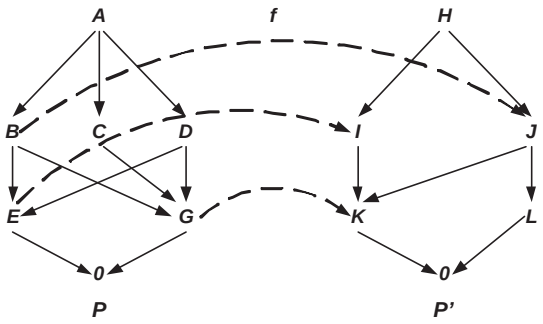


Figure 8: An example alignment.

- A characterization of semantic similarities S in terms of order metrics.
- Induction of new alignment links based on searching for low-discrepancy mappings within the space of order morphisms.
- Characterization of ontology link types in terms of hierarchical structure, factoring transitive and non-transitive link types.
- Identification of measures of centroid and dispersion in ontology clustering tasks.
- Anomaly detection in concept lattice based on correlation of interval rank and extent/intent size.

References

[1] Aho, AV; Garey, MR; and Ullman, JD: (1972) "The Transitive Reduction of a Directed Graph", *SIAM Journal of Computing*, v. **1**:2, pp. 131-137

[2] Ashburner, M; Ball, CA; and Blake, JA et al.: (2000) "Gene Ontology: Tool For the Unification of Biology", *Nature Genetics*, v. **25**:1, pp. 25-29

[3] A Butanitsky and G Hirst: (2006) "Evaluating WordNet-based Measures of Lexical Semantic Relatedness", *Computational Linguistics*, v. **32**:1, pp. 13-47

[4] A Clauset, C Moore, M Newman: (2006) "Structural Inference of Hierarchies in Networks", in: *Proc. 23rd Int. Conf. on Machine Learning*

[5] A Clauset, C Moore, M Newman: (2008) "Hierarchical Structure and the Prediction of Missing Links in Networks", *Nature*, v. **453**, pp. 98-101, doi:10.1038/nature06830

[6] T Cormen, CE Leiserson, RL Rivest: (1990) *Introduction to Algorithms*, MIT Press, Cambridge MA

[7] BA Davey, HA Priestly: (1990) *Introduction to Lattices and Order*, Cambridge UP, Cambridge UK, 2nd Edition

[8] Euzenat, Jérôme and Shvaiko, P: (2007) *Ontology Matching*, Springer-Verlag, Hiedelberg

[9] Fellbaum, Christiane, ed.: (1998) *Wordnet: An Electronic Lexical Database*, MIT Press, Cambridge, MA

[10] Ganter, Bernhard; Stumme, Gerd; and Wille, Rudolf, eds.: (2005) *Formal Concept Analysis: Foundations and Applications*, Springer-Verlag

[11] Ganter, Bernhard and Wille, Rudolf: (1999) *Formal Concept Analysis*, Springer-Verlag

[12] Joslyn, Cliff: (2004) "Poset Ontologies and Concept Lattices as Semantic Hierarchies", in: *Conceptual Structures at Work, Lecture Notes in Artificial Intelligence*, v. **3127**, ed. Wolff, Pfeiffer and Delugach, pp. 287-302, Springer-Verlag, Berlin

[13] CA Joslyn, B Baddeley, J Blake, C Bult, M Dolan, R Riensche, K Rodland, A Sanfilippo, A White: (2009) "Automated Annotation-Based Bio-Ontology Alignment with Structural Validation", *Proc. Int. Conf. on Biomedical Ontology (ICBO 09)*

[14] CA Joslyn, A Donaldson, P Paulson: (2008) "Evaluating the Structural Quality of Semantic Hierarchy Alignments", *Int. Semantic Web Conf. (ISWC 08)*, http://dblp.uni-trier.de/db/conf/semweb/iswc2008p.html#JoslynDP08

[15] Joslyn, Cliff; Mniszewski, Susan; Fulmer, Andy, and Heaton, G: (2004) "The Gene Ontology Categorizer", *Bioinformatics*, v. **20**:s1, pp. 169-177

[16] CA Joslyn, SM Mniszewski, SA Smith, PM Weber: (2006) "SpindleViz: A Three Dimensional, Order Theoretical Visualization Environment for the Gene Ontology", in: *Joint BioLINK and 9th Bio-Ontologies Meeting (JBB 06)*, http://www.bio-ontologies.org.uk/2006/download/Joslyn2EtAlSpindleviz.pdf

[17] CA Joslyn, P Paulson, KM Verspoor: (2008) "Exploiting Term Relations for Semantic Hierarchy Construction", in: *Proc. Int. Conf. Semantic Computing (ICSC 08)*, pp. 42-49, IEEE Computer Society, Los Alamitos

[18] CA Joslyn, A Pogel, S Schmidt: (2008) "Ordered Set Interval Rank for Knowledge Systems Analysis and Visualization", in preparation

[19] McBride, Brian: (2002) "Jena: A Semantic Web Toolkit", *IEEE Internet Computing*, v. **6**:6, pp. 55-59

[20] Monjardet, B: (1981) "Metrics on Partially Ordered Sets - A Survey", *Discrete Mathematics*, v. **35**, pp. 173-184

[21] Nuutila, Esko and Soesalon, Eljas: (1994) "On Finding the Strongly Connected Components in a Directed Graph", *Information Processing Letters*, v. **49**, pp. 9-14

[22] C Orum, CA Joslyn: (2009) *Valuations and Metrics on Partially Ordered Sets*, in: *Discrete Mathematics*, http://arxiv.org/abs/0903.2679v1, submitted

[23] Verspoor, KM; Cohn, JD; Mniszewski, SM; and Joslyn, CA: (2006) "A Categorization Approach to Automated Ontological Function Annotation", *Protein Science*, v. **15**, pp. 1544-1549

A Graphical Rethinking of the Cognitive Inner Loop

Paul S. Rosenbloom
Department of Computer Science & Institute for Creative Technologies
University of Southern California
rosenbloom@usc.edu

Abstract

Explorations of graphical representation and reasoning have yielded intriguing results spanning symbol, probability and signal processing. Here we explore an integrative application of graphs, as a path towards cognitive architectures of increased elegance, functionality, and extensibility. The specific focus is on steps towards a graphical reimplementation and extension of the cognitive inner loop within the Soar architecture. Alchemy, an implementation of Markov logic, is used for initial experiments, yielding insights into what will ultimately be required for full graphical implementations of enhanced cognitive inner loops.

1 Introduction

In [Rosenbloom, 2009] a new strategy was laid out for developing cognitive architectures [Newell, 1990] via a uniform *implementation level* based on factor graphs [Kschischang *et al*., 2001]. A cognitive architecture seeks to provide a coherent integration of capabilities sufficient for human-level artificial intelligence, whether in the context of a detailed model of human cognition or a system more loosely tied to the specifics of human behavior. Such an architecture requires the integration of a wide range of cognitive capabilities for, among other things, representation and memory, problem solving and planning, learning, reflection, interaction (including perception and motor control, use of language, etc), and the social aspects of cognition (such as emotion, collaboration, etc.).

The implementation level for cognitive architectures sits below the architectural memories and mechanisms, and provides the technologies out of which they are built. Traditionally, it is simply a programming language of some sort that may impact the efficiency, portability and robustness of the architecture, but is itself of little theoretical interest. The idea of basing the implementation level on graphical models looks to go beyond this by leveraging the uniform manner in which they support broad varieties of symbol, probability, and signal processing. The intuition is that, if the range of capabilities required for human-level intelligence can be built out of, and integrated within, such a uniform substrate then new architectures that are more elegant, functional and extensible may be possible. The ultimate goal is thus to determine whether a graphical implementation level can enable a new and improved generation of architectures.

As a step in this direction, I have been investigating a graphical reimplementation and enhancement of the Soar architecture [Rosenbloom *et al*., 1993]. Soar is one of the longest standing – over 25 years – and most thoroughly investigated cognitive architectures. It also possesses the unusual status of existing in both relatively uniform (up through version 8 [Laird and Rosenbloom, 1996]) and diverse (version 9 [Laird, 2008]) forms, providing a natural path for reimplementation that starts with a uniform version and then attempts a more uniform reintegration of later diversity. Simultaneously, opportunities can also be sought for expanding beyond Soar's predominant symbol processing paradigm, through the deep integration of probability and signal processing, in support of improved reasoning about, and interaction with, the real world.

This article: (1) examines what is involved in reconstructing a more uniform and functional graph-based *cognitive inner loop* for Soar, i.e., its core *decision cycle*, in which memory is accessed about the current situation and a decision is made about what to do next; (2) reports results from experiments towards this end based on Alchemy [Domingos *et al*., 2006]; and (3) identifies the path forward from here. While not yet achieving a fully implemented and enhanced version of Soar's decision cycle, it does yield critical insights into what will be necessary. First, however, the next two sections cover prior background on cognitive scales, Soar, and the graphical reimplementation of Soar.

2 Cognitive Scales and Soar

Part of the theory behind Soar as a model of human cognition is that *scale counts in cognition* [Newell, 1990]. As cognition is analyzed in depth, the phenomena and their properties change as the focus shifts from small spatiotemporal scales to larger ones. Newell discusses time scales from 10^{-4} seconds (100 μs) up to 10^{7} seconds (months), and divides them into four bands in human cognition: biological (10^{-4}-10^{-2} seconds), cognitive (10^{-1}-10^{1} seconds), rational (10^{2}-10^{4} seconds) and social (10^{5}-10^{7} seconds). In the biological band in particular there is also a spatial aspect to these scales, since signals are limited in how far they can

travel within such small time frames. Organelles (10^{-4} seconds), neurons (10^{-3} seconds) and neural circuits (10^{-2} seconds) yield spatial scales within the biological band, before primitive deliberate acts (10^{-1} seconds) and operations (10^{0} seconds) are reached at the base of the cognitive band.

The architectural mechanisms in the earlier uniform versions of Soar were traditionally mapped onto a subset of these time scales, starting with the *elaboration cycle* at 10 ms (neural circuits), the decision cycle at 100 ms (deliberate acts), and activity in problem spaces at 1 second (operations) above this. The elaboration cycle involves parallel match (via a variant of the Rete algorithm [Forgy, 1982]) and firing of productions based on the contents of a global working memory. Functionally, it achieves one round of parallel associative retrieval of information relevant to the current situation. Production actions specify knowledge for potential retrieval while production conditions specify the circumstances under which that knowledge is relevant. Conditions also bind variables for use in actions.

The decision cycle involves repeated cycles of elaboration until quiescence; i.e., until no more productions can fire. This *elaboration phase* is followed by a decision based on preferences retrieved during elaboration. The elaboration phase yields an interpretation of the current situation, while the decision either selects an *operator* or generates an *impasse* if no operator can be selected. Impasses engender *reflection*, enabling processing to recur at the meta-level on the problem of making the decision. The decision cycle is Soar's cognitive inner loop – it accesses whatever knowledge is immediately available about the current situation and then attempts to decide what to do next.

A sequence of decisions yields activity in a problem space, amounting to some form of search if knowledge is limited and impasses occur. Search in problem spaces (ps-search) is: *slow*, with each decision occurring at the 100 ms level; *serial*, via a sequence of operator selections and applications; and potentially *combinatoric*, yielding trees that grow exponentially in the depth of the search. However, ps-search is open to control by any knowledge accessible during the decisions that occur within it. When the knowledge is sufficient to uniquely determine the outcome of each decision, behavior is more accurately characterized as algorithmic, or knowledge-driven, than as search.

Accessing knowledge during a decision can also be viewed as a search process – termed knowledge search (k-search) – but one that contrasts strongly with ps-search in character. K-search is: *fast*, with a 10 ms cycle time, *parallel*, both in match and firing of productions; and *subexponential*, at least in theory, if not in reality in most implementations. K-search occurs over a closed, extensionally defined, set of structures – the knowledge/productions in the system – rather than dynamically generating an open search space in the manner of ps-search. It is inherently algorithmic, rather than using an open cognitive loop, and is thus not itself penetrable by additional control knowledge.

Chunking [Laird *et al*., 1986] is a learning mechanism in Soar that generates new productions based on the results of problem space activity during impasses. It compiles knowledge that is initially only available through activity at time scales of 1 second or more down to knowledge that is "immediately available" for use at the 10 ms time scale. Chunking, in combination with the flexibility of Soar's problem solving, has been shown to yield a much wider range of learning behaviors than just simple speed ups [Rosenbloom, 2006] – such as concept acquisition and episodic learning – but speeding up behavior remains its most essential functionality. In fact, the difficulty of producing some of these wider learning behaviors, and of integrating them with routine cognitive activity, was a key driver in Soar 9's shift towards diversity. Soar 9 adds, among other things, new varieties of long-term memory and learning.

3 Prior Work on the Elaboration Cycle

The work reported in [Rosenbloom, 2009] focused on reimplementing Soar's elaboration cycle (10 ms); and, in particular, on factor-graph algorithms for production match. Factor graphs in general provide a means of efficiently working with nearly decomposable functions of many variables. They arose in coding theory, where they underlie the surprisingly effective performance of turbo codes. They are similar to Markov networks (aka Markov random fields) in being undirected graphs with nodes that correspond to variables. However, in addition to variable nodes there are also factor nodes that represent functions over subsets of the variables. Factor nodes are analogous to clique potentials/weights in Markov networks, but they are directly incorporated as network nodes in factor graphs. Inference in factor graphs may be done through variations on the standard summary-product algorithm – a message passing approach that generalizes the more familiar (loopy) belief propagation algorithm in Bayesian networks [Pearl, 1983] – or via Monte Carlo methods.

Although Soar's Rete match algorithm could potentially be implemented directly via factor graphs, the focus of the prior work was on match algorithms arising more naturally from factor graphs. The investigation began with a straightforward, although ultimately naïve, approach. Working memory was represented as a three dimensional array of potential working memory elements, with one dimension for each of the three slots of a working memory element – object, attribute and value – and a value of one in every array cell for which the corresponding element was in working memory and zero otherwise. In the factor graph, variable nodes corresponded to production variables while factor nodes corresponded to conditions and actions. Match occurred via the summary-product algorithm, passing messages about the legitimate bindings of condition variables, and eventually converging on bindings for action variables.

Without going into the gory details, this initial approach raised generality, correctness and efficiency issues that ultimately led, through a sequence of optimizations and conceptual adjustments, to a new graphical match algorithm combining: (1) a junction-tree-like approach for graph construction, to enable the tracking of compatible combinations of bindings for different variables; and (2) an N-dimensional generalization of quad/octrees (called *exptrees* for lack of an

existing term) for working memory and messages that enables uniform regions – i.e., regions in which all of the potential working memory or message elements are either present (one) or absent (zero) – to be matched without examining each element individually. The resulting match algorithm yielded correct results with dramatically reduced match times from the naïve approach. It also avoided creating the full production instantiations required by Rete, reducing the worst-case bound on match cost to exponential in the *treewidth* of a production rather than in the number of conditions in the production (as in Rete).

Beyond match, the remainder of the elaboration cycle consists of the firing of productions, empowering instantiated actions to add and delete working memory elements by flipping the corresponding array values from zero to one or vice versa. Once working memory is updated, the next elaboration cycle can begin.

4 Rethinking the Decision Cycle

In the work reported here, the focus has moved up to the decision cycle (100 ms) – Soar's cognitive inner loop – comprising an elaboration phase and a decision. This is the lowest level at which knowledge may affect decisions, at which multiple fragments of knowledge may be combined, and at which k-search may involve more than one cycle of match and firing. It is also the key scale at which extending Soar beyond strictly symbolic processing could lead to radically expanded functionality and at which it makes sense to begin considering incorporation of Soar 9's diversity.

Any reimplementation of Soar's elaboration phase must support its three core functions: (1) elaborating the description of the current situation in working memory based on relevant long-term knowledge; (2) generating operator preferences based on this elaborated working memory; and (3) altering working memory to reflect the application of selected operators. The first two functions are mostly monotonic, while the third is inherently non-monotonic. Overall, operation is similar to that of a truth maintenance system [Doyle, 1979], with operators determining the current assumptions and elaborations automatically asserting and retracting as these assumptions change.

Two additional constraints on the long-term knowledge must also be met by any reimplementation of the elaboration phase. The first constraint is that it must be capable of being processed in bounded time and space. Soar's production-based elaboration phase runs in time that is bounded by the *volume* of the elaboration phase – cost per production × number of productions × number of elaboration cycles. In reality, the second dimension is close to constant, as a suitably optimized Rete algorithm enables match time to remain close to constant with growth in the number of productions [Doorenbos, 1993]. However, the other two dimensions can be problematic. As mentioned earlier, the cost per production may be exponential in the size of the production. Even worse, the length of the elaboration phase can be infinite – new working memory elements can be generated on each elaboration cycle that lead to more productions firing in the next cycle. A reimplementation should at least avoid exacerbating these boundedness issues, and ideally improve on them (such as the prior work's improved match bound).

The second constraint is that the long-term knowledge must be learnable. Soar acquires productions via chunking, and Soar 9 adds other mechanisms to acquire its additional varieties of long-term knowledge; but satisfying this constraint in a graphical reimplementation is left to future work.

Beyond these two constraints, the uniform versions of Soar also lived with the constraint that all long-term knowledge must be cast as productions. Productions have the advantage that they are uniform, active, relatively flexible, and learnable. They also have a long successful history in cognitive modeling. Still, they have proven balky in dealing with both declarative and perceptual knowledge, ultimately leading to the elimination of this long held constraint in Soar 9 and the addition of three new long-term memories – two for declarative knowledge (semantic and episodic) and one for perceptual knowledge (visual imagery) – each with its own distinct variety of knowledge structures.

The approach explored here is not to eliminate the third constraint, but to replace it with one based on the varieties of knowledge structures efficiently implementable via graphical models. The hope is thereby to support a much wider range of functionality – including symbol, probability, and signal processing, as well as Soar 9's new kinds of knowledge structures – in a general yet uniform fashion.

The prior work discussed in Section 3 implemented a complete elaboration cycle. A straightforward elaboration phase is thus obtainable merely by repeating these cycles until quiescence is reached. While such an elaboration phase has been implemented, and initial ideas exist for extending it to continuous values and declarative memory, it has a serious flaw in only being able to propagate information forward across rule firings. Bidirectional information flow is needed for probabilistic information to propagate correctly across rules. It is also necessary for the implementation of *trellis* diagrams – in which a graph is composed of a linked sequence of identical subgraphs – such as the hidden Markov models used in speech recognition and other varieties of sequential signal processing.

The prior implementation supported bidirectional information flow within rules, and reused the same rule graph on each elaboration cycle – as is needed for a trellis – but the only linkage across cycles was implicit in the working memory elements generated during early elaboration cycles and matched on later ones. In addition to a graph for the generalized rules, a graph representing rule instantiations and the linkages among them may be needed to support bidirectional information flow across the rule instantiations generated within an elaboration phase.

In contrast to the elaboration phase, there are many fewer constraints on the decision procedure that follows it. Decisions in Soar were based on vote counting in a very early version, on symbolic preferences – acceptable, reject, better worse, etc. – in most versions, and on a combination of symbolic and (additive) numeric preferences in Soar 9. The key constraint on a reimplementation of the decision procedure is that all of the preferences accessed during the elabo-

ration phase must be combined in an appropriate and tractable manner to yield either the selection of a unique operator or the detection of an impasse.

5 Progress towards a New Decision Cycle

The lack of bidirectional message passing across elaboration cycles in the existing implementation, in conjunction with a desire to better understand the utility of existing graphical languages – in particular those that already combine some forms of symbolic and probabilistic reasoning, such as Alchemy, BLOG [Milch *et al*., 2007], and FACTORIE [McCallum *et al*., 2008] – for implementing cognitive architectures, led to the decision to begin investigating the revision of Soar's decision cycle via such a language. Alchemy, which is based on combining first-order logic and Markov networks to form *Markov logic*, was ultimately selected because it: supports forms of both symbolic and probabilistic processing along with nascent signal processing [Wang and Domingos, 2008], provides an obvious approach to working with both rules and their instantiations, is publically available, runs on multiple types of computers, and has manuals, tutorials, and rapid response to emailed questions.[1]

To date, several small-scale experiments have been run with Alchemy: (1) re-implementing simple production systems that had previously been implemented via factor graphs; (2) adding a form of semantic long-term memory to the production memory; (3) exploring an implementation of the eight puzzle, one of the earliest tasks investigated in Soar [Laird and Newell, 1983] and the basis for early learning experiments with it [Laird *et al*., 1986]; and (4) experimenting with trellis diagrams.[2]

In Alchemy, a *Markov logic network* (MLN) is defined via first-order predicates and formulas, with weights assigned to the formulas. The MLN is then compiled into a *ground Markov network* with binary nodes for each ground predicate, links among nodes that appear in common formulas, and features for each possible ground formula. Inference is performed on this ground Markov network, unless additional optimizations such as laziness (where grounding only occurs for variables that take on non-default values [Poon *et al.*, 2008]) or lifting (where multiple ground atoms are combined into single network nodes when they can be guaranteed to pass the same messages during belief propagation [Singla and Domingos, 2008]) are included.

The initial mapping of Alchemy to Soar's decision cycle focused on the first two functions of the elaboration phase: elaborating the current situation in working memory based on the contents of (a production-based) long-term memory, and generating preferences. Productions were represented as conditional formulas in an MLN file and the state of working memory at the beginning of the decision cycle was represented as evidence in an Alchemy database file. A single elaboration phase was then mapped onto a single invocation of Alchemy's inference procedure with this network and database.

The details of this mapping and the ensuing experiments are relatively uninteresting, so they are omitted here to conserve space. What is worth noting though are the implications of these experiments for a graphical reimplementation of Soar in particular, and a graphical implementation level for cognitive architectures in general. The most critical result is that the core of the mapping works, enabling a uniform elaboration phase that combines Soar's standard rule-based capabilities with probabilistic reasoning, simple trellises and semantic memory. The approach solves the aforementioned bidirectional, across rule, information flow problem by compiling the rules into a ground Markov network, and then performing inference in this ground network. Because nodes in this network correspond to working memory elements, and each such node links to every other element with which it coexists in a ground formula, the ground Markov network provides a single linked network for the entire elaboration phase. If the rules define a trellis, by repetition across elaboration cycles, bidirectional inference also occurs appropriately for it.

Another major result concerns the nature of production match under this mapping. Alchemy does not use inference in graphs to perform the equivalent of match for conditional formulas. Instead, match corresponds to Alchemy's extra-network process of compiling (first-order) Markov logic networks down to ground Markov networks. In essence, the Markov logic network corresponds to the definition of the production system while the ground Markov network corresponds to working memory elements (the ground nodes) and production instantiations (the ground formulas). In contrast to the prior implementation, working memory elements correspond to distinct nodes in this network rather than simply serving as the basis for messages among nodes.

Given that the goal is ultimately to implement a broadly functional cognitive architecture uniformly in graphs, Alchemy's match-as-compilation approach is problematic. A key question for future work therefore becomes whether it is possible to unify match – i.e., the computation of ground instances from first-order formulas – with the other necessary forms of inference into a single graph that is processed in a uniform manner, or whether it will be necessary to develop a dual graph/network approach in which match occurs via a first-order graph that generates, and is linked to, a ground graph in which the remaining inference occurs. Either way, the decision cycle will need to be extended from its current two stages to three: (1) compilation/match to generate a ground/instantiated network; (2) inference in the ground/instantiated network; and (3) decision making.

A final significant outcome is more conceptual, and concerns the general mapping between graphical systems and the hierarchy of cognitive scales, particularly as mediated by the mapping of both onto Soar. If the elaboration phase – which performs k-search (100 ms) – consists of the compi-

[1] Alchemy has also been explored in the Icarus cognitive architecture [Langley and Choi, 2006], with a focus specifically on the implementation of an inference component [Stracuzzi, 2009].

[2] Several of these experiments have been replicated with BLOG, but the results do not fundamentally alter the conclusions reported here based on Alchemy.

lation of, and inference in, a multi-layer ground network, then two important consequences follow:

1. The goal for a probabilistic first-order reasoner should not be a single uniform system capable of directly solving any problem no matter how complex. Instead, it should be bounded to the needs of k-search; e.g., only being capable of finding local minima in the solution space. Problems too complex to be solved in this manner would require a sequence of deliberate acts – i.e., steps in a problem space (ps-search) at the 1-second time scale – to move among local minima in search of a global minimum. Systems like Alchemy can get stuck in local minima [Stracuzzi, 2009], but according to this argument that is all a flat inference system should ever strive for. Reaching global minima in general requires a sequence of deliberate acts.
2. Cycles of message passing map onto the neural circuit (10 ms) scale. Functionally this implies that the 10 ms scale supports (only) local propagation of information, the 100 ms scale supports global propagation but (only) local minima, and global minima generally require time scales of 1 sec and above unless the problem is particularly simple or the system gets lucky.

Beyond these major implications, several smaller yet still interesting results have also been extracted from the mapping and resulting experiments:

3. Production systems utilize specific forms of non-monotonic reasoning, including an implicit closed-world assumption about the contents of working memory, and the ability to arbitrarily add and delete working memory elements. Such capabilities map awkwardly onto first-order reasoners, such as Alchemy.
4. Many production systems, Soar included, provide the ability to generate new unique symbols via production actions. Although such actions are local to individual productions, the process of checking uniqueness is a global activity that is difficult to implement through local message passing in a graph/network.
5. Exptrees served a role in the prior work that is analogous to what laziness and lifting achieve in Alchemy. The latter mechanisms eliminate unnecessary computation, either by avoiding the processing of default values or by grouping together items that can be treated the same. With exptrees, defaults are identified naturally and items are grouped by region if their values are identical. Exptrees appear to be a coarser approach, but it may ultimately be possible to bring these approaches more into alignment.
6. Experiments with simple trellises (linked repetitions) and semantic memory (encoded as ground atoms) have shown the feasibility of incorporating both within the decision cycle, but they involve computing most probable explanations (MPEs) rather than the marginals used for production match in the prior work to generate all instantiations. One possibility for the future is to localize the use of marginals to the generation of ground networks from first-order networks, and use MPE for all computations in the ground network.

Reflections on the first two of these smaller outcomes, in conjunction with the prior conclusion that the 10 ms scale only performs local propagation, has led to the conclusion that neither non-monotonicity nor the generation of new unique symbols should occur in individual productions (i.e., within an elaboration cycle). Non-monotonic reasoning has an implicit global aspect to it, given that the current answer is always dependent on nothing else being true that would overturn it. Operator implementation – the third function of the elaboration phase – and negated conditions in productions are examples of non-monotonic processing that thus should be banned from rules and moved up to the level of decision cycles. Generation of new unique symbols also involves an obvious global aspect.

Beyond the issues of non-monotonicity and symbol generation, limiting global information propagation to decision cycles and above implies that semantic memory, when defined in terms of finding the best match in memory to a cue [Anderson, 1990], should also occur at the level of decision cycles, as it currently does in Soar 9. Even more critically, though, this raises hard questions about the use of a global working memory in production match. One possible resolution to this dilemma would be to allow operator application to have global effects on working memory, as it is already being shifted up to the decision level, but to require elaboration to proceed via local propagation of information. Whether this can work, and more generally how to develop an effective architecture when all of the non-local forms of processing currently embodied by rules are moved up to the decision level, is a key issue for future work.

The actual decision making process has been neglected so far in this discussion. Limited experiments have been performed by leveraging Alchemy's provision of weights on formulas to encode preferences, and MPE inference to select operators based on these preferences. This has proven adequate for simple examples, but more complex ones are presently foundering on the preliminary step of dynamically generating operator instantiations and the accompanying unique symbols that are needed to identify them. Developing a full decision mechanism is thus left for future work.

6 Summary and Future Directions

This article has begun the exploration of graphical models for Soar's cognitive inner loop, with an Alchemy-based implementation of an elaboration phase that combines Soar's symbolic, rule-based, long-term memory with probabilities, simple bidirectional trellises and long-term semantic memory. In the process, four directions for the future have been explicitly called out: (1) satisfying the learnability constraint on long-term knowledge; (2) unifying rule match with inference in graphs while determining the respective roles of marginal versus MPE inference; (3) understanding how to feasibly and functionally move all non-local processing from the elaboration cycle to the decision cycle; and (4) implementing a complete decision procedure.

In addition, the full incorporation of signal processing, for perception and motor control, and of semantic and episodic knowledge is critical, and remains to be done. Beyond the

inner loop, there is more of Soar to be explored, along with other existing architectures and hybridizations among them. Totally new architectures that take full advantage of what graphical models provide also need investigation. The ultimate intent is to definitively answer the question first posed in [Rosenbloom, 2009] as to whether implementing cognitive architectures on top of a uniform graph-based implementation level can yield a new generation of architectures with improved uniformity, functionality, and extensibility.

Acknowledgments

This effort was made possible by sabbatical support from the USC Viterbi School of Engineering plus funding from the Institute for Creative Technologies (ICT). ICT's Cognitive Architecture Working Group has been invaluable for semi-public exploration of these ideas. I would also like to thank the Alchemy group at the University of Washington for their help in installing Alchemy and working through various issues that arose during experimentation with it.

References

[Anderson, 1990] John R. Anderson. *The Adaptive Character of Thought*. Erlbaum, Hillsdale, NJ, 1990.

[Domingos *et al.*, 2006] Pedro Domingos, Stanley Kok, Hoifung Poon, Matt Richardson, and Parag Singla. Unifying logical and statistical AI. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 2-7, July 2006. AAAI Press.

[Doorenbos, 1993] Robert B. Doorenbos. Matching 100,000 Learned Rules. In *Proceedings of the 11th National Conference on Artificial Intelligence. Page 290-296,* 1993.

[Doyle, 1979] John Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12(3): 251-272, 1979.

[Forgy, 1982] Charles L. Forgy. "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem". *Artificial Intelligence*, 19(1): 17-37, 1982.

[Kschischang *et al*., 2001] Frank R. Kschischang, Brendan J. Frey, Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2): 498-519, February 2001.

[Laird, 2008] John E. Laird. Extending the Soar cognitive architecture. In *Artificial General Intelligence 2008: Proceedings of the First AGI Conference*, Memphis, TN, March 2008. IOS Press.

[Laird and Newell, 1983] John E. Laird and Allen Newell. "A Universal Weak Method: Summary of Results." In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 771-773, Karlsruhe, FRG, August 1983. William Kaufmann.

[Laird and Rosenbloom, 1996] John E. Laird and Paul S. Rosenbloom. The evolution of the Soar cognitive architecture. In D. M. Steier. and T. M. Mitchell (Eds.), *Mind Matters: A Tribute to Allen Newell*, pages 1-50. Erlbaum, Mahwah, NJ, 1996.

[Laird *et al*., 1986] John E. Laird, Paul S. Rosenbloom, and Allen Newell. Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning,* 1(1): 11-46, March 1986.

[Langley and Choi, 2006] Pat Langley and Dongkyu Choi. A unified cognitive architecture for physical systems. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston, MA, 2006. AAAI Press.

[McCallum *et al*., 2008] Andrew McCallum, Khashayar Rohanemanesh, Michael Wick, Karl Schultz and Sameer Singh. FACTORIE: Efficient probabilistic programming via imperative declarations of structure, inference and learning. In *Proceedings of the NIPS workshop on Probabilistic Programming*, Vancouver, Canada, 2008.

[Milch *et al*., 2007] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic models with unknown objects. In L. Getoor and B. Taskar, (Eds.) *Introduction to Statistical Relational Learning*, pages 373-398. MIT Press, Cambridge, MA, 2007.

[Newell, 1990] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, 1990.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, CA, 1988.

[Poon *et al.*, 2008] Hoifung Poon, Pedro Domingos, and Marc Sumner. A general method for reducing the complexity of relational inference and its application to MCMC. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1075-1080, July 2008. AAAI Press.

[Rosenbloom, 2006] Paul S. Rosenbloom. A cognitive odyssey: From the power law of practice to a general learning mechanism and beyond. *Tutorials in Quantitative Methods for Psychology*, 2(2): 43-51, 2006.

[Rosenbloom, 2009] Paul S. Rosenbloom. Towards a new cognitive hourglass: Uniform implementation of cognitive architecture via factor graphs. Submitted to the *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[Rosenbloom *et al*., 1993] Paul S. Rosenbloom, John E. Laird, and Allen Newell. *The Soar Papers: Research on Integrated Intelligence*. MIT Press, Cambridge, MA, 1993.

[Singla and Domingos, 2008] Parag Singla and Pedro Domingos. Lifted first-order belief propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1094-1099, July 2008. AAAI Press.

[Stracuzzi, 2009] David Stracuzzi. Personal Communication, 2009.

[Wang and Domingos, 2008] Jue Wang and Pedro Domingos. Hybrid Markov logic networks. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1106-1111, July 2008. AAAI Press.

A generalised network flow approach to combinatorial auctions

Madalina Croitoru [1], **Cornelius Croitoru**[2]
[1]LIRMM (CNRS and Univ. Montpellier), France
[2]Universitatea Al. I. Cuza, Iasi, Romania

Abstract

In this paper we address the problem of (1) representing bids for combinatorial auctions and (2) employing those structures for "reasoning". We propose a graph-based language who's novelty lies (1) in the use of generalized network flows to represent the bids and (2) in the interpretation of winner determination as an adequate aggregation of individual preferences. We motivate the language both from representational and reasoning points of view and show how our language represents the same class of expressivity of bids more concisely compared to existing work.

1 Introduction

In every Artificial Intelligence system addressing a given problem there is a need to (1) represent the state of the world and (2) reason about possible ways to solve the problem. In this paper we address the problem of (1) representing bids for combinatorial auctions and (2) employing those structures for "reasoning" (winner determination). The proposed language we detail is a visual, graph-based language based on network flow modelling techniques that demonstrate better conciseness within the same expressivity classes.

Combinatorial auctions (CAs) can be looked at as a way of approaching allocation problems involving multiple heterogeneous goods. Bidding is the problem of representing one's valuation function over the set of goods on offer. It plays a key role in both central aspects of the allocation problem: preference elicitation and winner-determination (WD). As a consequence bidding languages have not only to address representational issues but also to provide subsequent manipulation techniques for reasoning aspects. Our motivation for introducing a new language is based on the fact that existing languages cannot concisely represent some structured valuations that might occur in practical scenarios. Moreover, these languages were not designed with partial value revelation in mind; this is especially important in domains where the valuation problem is hard. Following from above mentioned representational choices the algorithms for winner determination cannot fully take advantage of the structural optimisation potential of the problem at hand.

This paper proposes a visual language for combinatorial auctions based on generalised flow networks. The nodes of the network will represent either (1) resources, (2) bundles of resources or (3) composite nodes used for calculation of certain partial valuations. The flow defined on the edges will allow concise description of an exponential number of bids. The same structure will be used for the auctioneer to unify all bidders' valuations. The winner determination problem will then be translated into a special MAX FLOW problem on the proposed network structure.

The paper is structured as follows. In section 2 we motivate our language from a representational viewpoint and demonstrate its conciseness. The formal, rigorous semantics of the language are introduced in section 3. Based on the constructs from section 3 we guarantee the soundness of the syntax further introduced in section 4. Section 5 concludes the paper.

2 Motivation

A hypergraph (or a *bundle system*) is a pair $H = (R, \mathcal{B})$ where R is a finite set (the *resources* set, the set of *goods*) called the *vertex set* of H and $\mathcal{B}$ is a family of subsets of R. The members of $\mathcal{B}$ are called *hyperedges* and they are subsets of resources, or *bundles*. A hypergraph H can be explicitly represented in visual manner by a bipartite graph $B(H)$ having one vertex class corresponding to the resources set R and the other class corresponding to the H's bundles, and connecting by an edge a bundle vertex to its corresponding (members) resource vertices. This is shown in Figure 1.

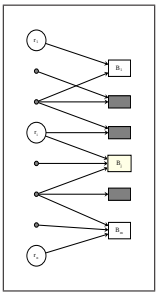


Figure 1: Bipartite graph representation of a bundle system

The bipartite graph has $|R| + |\mathcal{B}|$ vertices and $\sum_{B \in \mathcal{B}} |B|$ edges. If H is given explicitly, this is a concise and intuitive representation for a bundle system. The (directed) edges

of this bipartite graph suggest the containment relation of resources to bundles. However, if H is given by using some constructive (or implicitly) rules, the bipartite representation must be extended in order to be an effective representational tool. For example, if $\mathcal{B}$ is the family of all bundles having $\lceil \frac{|R|}{2} \rceil$ resources, then the corresponding bipartite graph has an exponential number of bundle vertices and edges. We will extend the above containment relation (of resources to bundles) by using paths (a resource belongs to a bundle if and only if there is a certain path from the resource vertex to the bundle vertex) and a mechanism to express which path must be considered in order to instantiate a given bundle. This mechanism is based on a simple extension of network flows, which is described bellow.

In our representational networks we will use the following graphical primitives depicted in Figure 2:

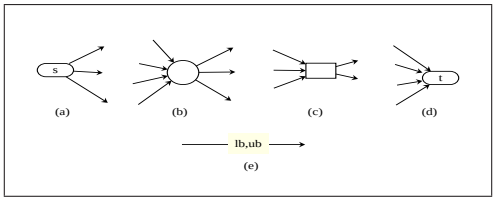


Figure 2: Elements of a bidflow network

The node (a) is the *start* node of the network (sometimes the label *start* is used instead of s). From this node the *flow* is pushed (on the arcs leaving it) in the network. The flow on each arc is a nonnegative integer value. If the flow f_{ij} on an the arc ij is positive it must satisfy the restriction $lb_{ij} \leq f_{ij} \leq ub_{ij}$, where the *lower bound* lb (sometimes denoted by l), and the *upper bound* ub (sometimes called *capacity* and denoted by c) are indicated as labels on the arc, as in construction (e) in Figure 2. The arcs without bound constraints (having $lb = 0$ and $ub = \infty$) are not labelled.

The nodes of type (b) are *transit nodes*, that is nodes which automatically distribute the total incoming flow (the sum of the flows on all arcs entering such a node) on the arcs leaving it. In other words, in these nodes the flow conservation law holds. They can have name-labels inside of the oval, for modelling or referring necessities.

The start node is connected by an arc labelled $0, 1$ to a node of type (b) labelled r, for each resource $r \in R$. From the flow conservation law (which holds in the transit node labelled r) and by the integrality of flows, either the flow on the arc sr is 1, and there is exactly one arc with flow value 1 leaving the node r, or the flow on the arc sr is 0, meaning that the resource r will belong to no bundle. In the former case, a path with positive flows on its arcs will be constructed, which eventually will reach a type (c) node.

The nodes of type (c) are *bundle nodes*, which pass-on the incoming flow exactly on one arc leaving them. The flow on this arc is set to 1. Furthermore, a bundle node b is "on" only if all the flows on the arcs entering it are positive. The intuition is that such a node collects all the resources $r \in R$ which belongs to a path starting from s, having positive flows on its arcs and ending in b. If the bundle represented by the node b is a not a member of $\mathcal{B}$ (the bundle system to be represented by the network) then the arcs leaving b are used to simulate (disjoint) unions in order to construct such a member via paths with positive arc flows. Of course, the structure of the network will prevent the existence of cycles. If the bundle represented by the node b is a member of $\mathcal{B}$, then there is an arc bt, leaving b and entering the *terminus node* of the network (type (d) in Figure 2, labelled *end*).

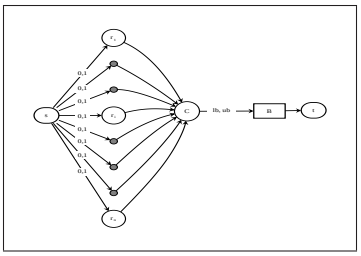


Figure 3: An exponential sized bundle system

Let us consider again the bundle system $H = (R, \mathcal{B})$, where $\mathcal{B}$ is the family of all bundles having $\lceil \frac{n}{2} \rceil$ resources ($n = |R|$). The network representing H using the above principles is given in Figure 3. If the ub and lb values on the arc cb are set to $\lceil \frac{n}{2} \rceil$, then for each $\lceil \frac{n}{2} \rceil$-subset S of R we can consider the flow f^S by putting: $f^S_{sr} = 1, f^S_{rc} = 1 \; \forall s \in S$; $f^S_{sr} = 0, f^S_{rc} = 0 \; \forall r \in R - S$; $f^S_{cb} = \lceil \frac{n}{2} \rceil$; and $f^S_{bt} = 1$. Clearly, the bundle represented by the node b is S. Conversely, it is not difficult to see that each non null flow f in this network generates a $\lceil \frac{n}{2} \rceil$-bundle B^f of R, by considering $B^f = \{r \in R | f(sr) = 1\}$. Note that the network has only $2|R| + 4$ nodes and $2|R| + 2$ arcs. It follows that the internal data structures have total polynomial size and also the number of variables (arc flows values) used is small.

A nice property of this type of representation is that if we are interested in an *induced subhypergraph*, that is to consider the members of $\mathcal{B}$ contained in some subset $S \subseteq R$, then it suffices to block the flow on the arcs sr for $r \in R - S$, by considering $ub_{sr} = 0$. This is clearly important for the use of v-basis as described in section 3. The restriction given by the Corollary in that section, could also be avoided in a succinct way (equivalently, considering Nisan's OR* language [8]) by adding a transit node, a new bundle node and an arc with ub set to 1 as described in Figure 4.

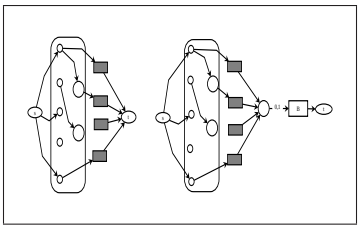


Figure 4: Implementing OR* trick.

In order to represent valuation functions using bids (i.e.v-basis) it is necessary to describe a mechanism to specify the the bid value of a bundle. This is obtained by associating *val-*

ues to the flows via special labels of network nodes. These labels indicate simple local functions which must be incrementally applied to values of the tail nodes from which the flow enter in the current node. The resulting network is called NETBID and is formally described in the section 4.

3 Semantics

This section presents that rationale that led to the contribution of the paper. The results obtained in this section build upon [4] and will lay the foundations for the semantics of the language in a rigorous, complete manner. A *Combinatorial Auction* (CA) can be interpreted as an abstraction of a marked-based centralized distributed system for the determination of *adequate* allocations of heterogenous indivisible resources. In such an *Adequate Resource Allocation* (ARA) system, there is central node a, the *auctioneer*, and a set of n nodes, $I = \{1, \ldots, n\}$, the *bidders*, which concurrently demand bundles of resources from a common set of available resources, $R = \{r_1, \ldots, r_m\}$, held by the auctioneer.

The auctioneer broadcasts R to all n bidders, asking them to submit in a specified common language, the *bidding language*, their *R-valuations* over bundles of resources. Bidder's i R-valuation, v_i, is a non-negative real function on $\mathcal{P}(R)$, expressing for each bundle $S \subseteq R$ the *individual* interest (value), $v_i(S)$, of bidder i in obtaining S. Naturally, it is assumed that $v_i(\emptyset) = 0$, and $v_i(S) \leq v_i(T)$ whenever $S \subseteq T$. No bidder i knows the valuation of any other $n-1$ bidders, but all the participants in the system agreed on an *adequate outcome*: ***(1)*** Based on bidders' R-valuations, the auctioneer will determine a resources *allocation* $\mathbf{O} = (O_1, \ldots, O_n)$, specifying for each bidder i her obtained bundle O_i. $\mathbf{O}$ is a (weak) n-partition of R, that is $O_i \cap O_j = \emptyset$, for any different bidders i and j, and $\cup_{i=1,n} O_i = R$. The global (social) value of the outcome is $v_a(\mathbf{O}) = \sum_{j=1,n} v_j(O_j)$; ***(2)*** $\mathbf{O}$ is an *adequate allocation*: if some bidder i does not receive its most wanted bundle (there is $S \subseteq R$ such that $v_i(S) > v_i(O_i)$) this is explained by the fact that the social (global) value of the outcome $v_a(\mathbf{O})$, would not increase if she will receive S: $v_a(\mathbf{O}) = \sum_{j=1,n} v_j(O_j) \leq \sum_{j=1,n} v_j(O'_j) = v_a(\mathbf{O}')$, for any allocation $\mathbf{O}' = (O'_1, \ldots, O'_n)$ having $O'_i = S$.

It is not difficult to see that there exists always such adequate allocation: let $O^* = (O^*_1, \ldots, O^*_n)$ be such that $v_a(O^*) = \max\{v_a(O) | O \text{ is a } n\text{-partition of } R\}$; if there is i and S such that $v_i(S) > v_i(O^*)$, then, by the choice of O^*, $v_a(O^*) \geq v_a(O')$ for every n-partition O' with $O'_i = S$. Conversely, if O is an adequate allocation and O^* is a maximum value allocation such that $v_a(O^*) > v_a(O)$, then (by the non-negativity property of the valuations) there is a bidder i such that $v_i(O_i) < v_i(O^*_i)$. Since O is adequate, taking $S = O^*_i$, it follows that $v_a(O) \geq v_a(O^*)$, (since O^* is a n-partition of R with $O^*_i = S$), a contradiction. We have obtained that *an allocation is adequate if and only if it is a maximum value allocation.*

The task of the auctioneer finding a maximum value allocation for a given set of bidder valuations $\{v_1, \ldots, v_n\}$, is called in the CA's field the *Winner Determination Problem* (WDP). This is a NP-hard problem, being equivalent to weighted set-packing. It tends to be solvable in many practical cases, but care is often required in formulating the problem to capture structure that is present in the domain ([9]).

WDP can be parameterized by the set R of resources, considering a fixed set I of bidders and bidders' R-valuations $\{v_i | i \in I\}$. Therefore we can write $WDP(R)$ and its corresponding maximum value $v_a(R)$. With these notations, $WDP(S)$ and $v_a(S)$ are well defined for each subset $S \subseteq R$ (by considering the restriction of v_i to $\mathcal{P}(S)$).

In this way, we have obtained a global R-valuation v_a assigning to each bundle $S \subseteq R$ the maximum value of an S-allocation to the bidders from I. By the above observation, this maximum value is the value of an adequate S-allocation. Therefore WDP can be interpreted as *the problem of constructing a social aggregation of the R-valuations of the bidders.*

If we denote by $\mathcal{V}(R)$ the set of all R-valuations, it is natural to consider in our ARA system the set of superadditive R-valuations due to the synergies among the resources: $\mathcal{SV}(R) = \{v \in \mathcal{V}(R) | \, v(B_1 \cup B_2) \geq v(B_1) + v(B_2) \text{ for all } B_1, B_2 \subseteq R, B_1 \cap B_2 = \emptyset\}$.

It is not difficult to see that if all $v_i \in I$ are superadditive then v_a is superadditive. Indeed, if $B_1, B_2 \subseteq R, B_1 \cap B_2 = \emptyset$, then $v_a(B_1) + v_a(B_2) = \sum_{i \in I} v_i(O^1_i) + \sum_{i \in I} v_i(O^2_i)$, where $\mathbf{O}^1$ is a maximum B_1-allocation and $\mathbf{O}^2$ is a maximum B_2-allocation; since $\mathbf{O}$ with $O_i = O^1_i \cup O^2_i$ $(i \in I)$ is a $B_1 \cup B_2$-allocation and $v_i(O^1_i) + v_i(O^2_i) \leq v_i(O^1_i \cup O^2_i)$, it follows that $v_a(B_1) + v_a(B_2) \leq v_a(B_1 \cup B_2)$.

The following lemma gives an interesting characterization of superadditive bidding.

Let us denote $\mathcal{V}^{OR}(R) = \{v \in \mathcal{V}(R) | \, v(B) = \max_{A \subseteq B}[v(A) + v(B - A)] \text{ for all } B \subseteq R\}$. Then,

Lemma 1 $\mathcal{SV}(R) = \mathcal{V}^{OR}(R)$.

Proof. If $v \in \mathcal{SV}(R)$ then for each $B \subseteq R$ and $A \subseteq B$ we have $v(B) \geq v(A \cup (B - A)) \geq v(A) + v(B - A)$, therefore $v(B) \geq \max_{A \subseteq B}[v(A) + v(B - A)]$. Since, for $A = \emptyset$, we have $v(B) = v(\emptyset) + v(B)$, it follows that $v(B) = \max_{A \subseteq B}[v(A) + v(B - A)]$, that is $v \in \mathcal{V}^{OR}(R)$. Conversely, let $v \in \mathcal{V}^{OR}(R)$. If $B_1, B_2 \subseteq R, B_1 \cap B_2 = \emptyset$, then $v(B_1 \cup B_2) = \max_{A \subseteq B_1 \cup B_2}[v(A) + v(B_1 \cup B_2 - A)] \geq v(B_1) + v(B_1 \cup B_2 - B_1) = v(B_1) + v(B_2)$, that is $v \in \mathcal{SV}(R)$.□

Combining this remark on the superadditivity of v_a and Lemma 1 we obtain:

Theorem 1 *If in an ARA system all bidders' R-valuations are superadditive, then the aggregate R-valuation v_a satisfies $v_a(A) = \max_{B \subseteq A}[v_a(B) + v_a(A - B)]$ for all $A \subseteq R$.*

Let $v \in \mathcal{V}(R)$. A *v-basis* is any $\mathcal{B} \subseteq \mathcal{P}(R)$ such that for each $A \subseteq R$ we have $v(A) = \max_{B \in \mathcal{B}, B \subseteq A}[v(B) + v(A - B)]$. In other words, if $\mathcal{B}$ is a v-basis, then the value of $v(A)$ is uniquely determined by the values of v on the elements of the basis contained in A, for each $A \subseteq R$. The elements of a v-basis, $B \in \mathcal{B}$, are called bundles and the pairs $(B, v(B))_{B \in \mathcal{B}}$ are called bids.

Clearly, $v \in \mathcal{V}^{OR}(R)$ if and only if $\mathcal{P}(R)$ is a v-basis. On the other hand, if $\mathcal{B} \subseteq \mathcal{P}(R)$ is a v-basis and $B \subseteq R$, then $\mathcal{B} \cup \{B\}$ is a v-basis too. Therefore, $v \in \mathcal{V}(R)$ has a v-basis iff $\mathcal{P}(R)$ is a v-basis. Using Lemma 1, we obtain the well known result (Nisan, [8]):

Corollary *A R-valuation $v \in \mathcal{V}(R)$ has a v-basis iff $v \in \mathcal{SV}(R)$.*

Let now consider an ARA system in which all bidders' R-valuations are superadditive. Each bidder $i \in I$ sends to the auctioneer its v_i-basis $\mathcal{B}_i$. The aggregate R-valuation v_a can be represented by a v_a-basis $\mathcal{B}_a$, which is obtained by merging the individual basis $\mathcal{B}_i$ in a very simply way: $\mathcal{B}_a = \cup_{i\in I}\mathcal{B}_i$ and if $B \in \mathcal{B}_a$ then $v_a(B) = \max\{v_i(B)|i \in I \text{ and } B \in \mathcal{B}_i\}$.

Indeed, by theorem 1, we have $v_a(A) = \max_{B\subseteq A}[v_a(B) + v_a(A - B)]$, for all $A \subseteq R$. If $\mathbf{O} = (O_1, \ldots, O_n)$ is a maximum A-allocation, then $v_a(A) = \sum_{i\in I} v_i(O_i)$. If $v_i(O_i) > 0$, then it is not difficult to see that $v_i(O_i) = v_a(O_i) \geq v_j(O_i)$ for all $j \in I$ and $v_a(A) = v_a(O_i)+\sum_{j\in I-\{i\}} v_j(O_j) = v_a(O_i)+v_a(A-O_i)$.

Furthermore, since $\mathcal{B}_i$ is a v_i-basis there is $B_i \in \mathcal{B}_i$ such that $v_i(O_i) = v_i(B_i) + v_i(O_i - B_i)$, $v_i(O_i) = v_a(B_i)$ and, moreover, $v_a(A) = v_a(B_i) + v_a(A - B_i)$.

We obtained the following interesting representational theorem:

Theorem 2 *If in an ARA system the bidder superadditive R-valuations v_i are represented using v_i-basis $\mathcal{B}_i$ for each $i \in I$, then the aggregate R-valuation v_a is represented by the v_a-basis $\mathcal{B}_a = \cup_{i\in I}\mathcal{B}_i$, by taking $v_a(B) = \max\{v_i(B)|i \in I \text{ and } B \in \mathcal{B}_i\}$, for all $B \in \mathcal{B}_a$.*

We note here that Lemma 1 can be extended to obtain a similar characterization of a subclass of additive R-valuations.

Let us consider $\mathcal{SUPV}(R)$, the set of all *supermodular R-valuations*, that is $\mathcal{SUPV}(R) = \{v \in \mathcal{V}(R)|\, v(B_1 \cup B_2) \geq v(B_1) + v(B_2) - v(B_1 \cap B_2)$ for all $B_1, B_2 \subseteq R\}$.

Clearly, $\mathcal{SUPV}(R) \subset \mathcal{SV}(R)$.

Also, we restrict the set of OR-valuations, by considering *strongly OR-valuations* (sOR-valuations): $\mathcal{V}^{sOR}(R) = \{v \in \mathcal{V}(r)|\, v(B) = \max_{A_1,A_2\subseteq B}[v(A_1) + v(A_2) - v(A_1 \cap A_2)]$ for all $B \subseteq R\}$.

Lemma 2 $\mathcal{SUPV}(R) = \mathcal{V}^{sOR}(R)$.

Proof. If $v \in \mathcal{SUPV}(R)$, then for each $B \subseteq R$ and $A_1, A_2 \subseteq B$ we have $v(B) \geq v(A_1 \cup A_2) \geq v(A_1) + v(A_2) - v(A_1 \cap A_2)$, therefore $v(B) \geq \max_{A_1,A_2\subseteq B}[v(A_1) + v(A_2) - v(A_1 \cap A_2]$. Since, for $A_1 = B$ and $A_2 = \emptyset$, we have $v(B) = v(B) + v(\emptyset) - v(B \cap \emptyset)$, it follows that $v(B) = \max_{A_1,A_2\subseteq B}[v(A_1) + v(A_2) - v(A_1 \cap A_2]$, that is $v \in \mathcal{V}^{sOR}(R)$. Conversely, let $v \in \mathcal{V}^{sOR}(R)$. If $B_1, B_2 \subseteq R$, then $v(B_1 \cup B_2) = \max_{A_1,A_2\subseteq B_1\cup B_2}[v(A_1) + v(A_2) - v(A_1 \cap A_2)] \geq v(B_1) + v(B_2) - v(B_1 \cap B_2)$, that is, $v \in \mathcal{SUPV}(R)$.□

As above, we have

Theorem 3 *If in an ARA system all bidders' R-valuations are supermodular, then the aggregate R-valuation v_a satisfies $v_a(B) = \max_{A_1,A_2\subseteq B}[v(A_1) + v(A_2) - v(A_1 \cap A_2)]$ for all $B \subseteq R$.*

4 Syntax

The proposed language is based on the following two novel ideas: (1) The use of generalized network flows to represent the bids; and (2) The interpretation of the WD as an adequate aggregation of individual preferences.

In the new language, each bidder submits to the arbitrator a generalized flow network representing its bids. We call such a network flow NETBID and it will represent the valuation of the bidder. More precisely, if the set of resources is $R = \{r_1, r_2, \ldots, r_m\}$, then in the NETBID of each agent there is a special node $START$ connected to m nodes r_j by directed edges having capacity 1. An integer flow in NETBID will represent an assignment of resources to the agent by considering the set of resources r_j with flow value 1 on the directed edge $(START, r_j)$. The node r_j is an usual node, that is, it satisfies the conservation law: the total (sum) of incoming flows equals the total flow of outcoming flows. In the network there are also *bundle* nodes which do not satisfy the conservation law, which are used to combine (via their inputs flows) different goods in subset of goods. The combination is conducted by the (integer) directed edges flows together with appropriate lower and capacity bounds. For example, the additive valuation, $v(S) = |S|$ for each subset S of R can be represented by the NETBID in Figure 5.

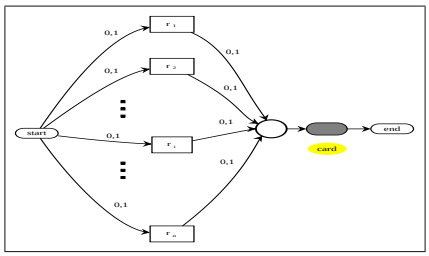


Figure 5: Additive valuation network

There is an important improvement over other existing graph-oriented bidding languages namely the possibility that a bundle node to represent an entire hypergraph having as vertices the resource set R. Furthermore, the nodes values are given by using labels on bundles nodes, which are positive real numbers or even procedural functions having as arguments the values of the incoming flows. This has as consequence a higher expressiveness of the bidding language.

Once the NETBID has been constructed, any maximum value flow (in the sense described above) will represent the valuation function of the agent. In particular cases it is not difficult for a rational bidder to construct a NETBID representing his preferences. For example, the NETBID in Figure 6 expresses that the bidder is interested in a bundle consisting in two or three resources of type E , together with the resource M which adds 10 to the values sum of the particular resources of type E:

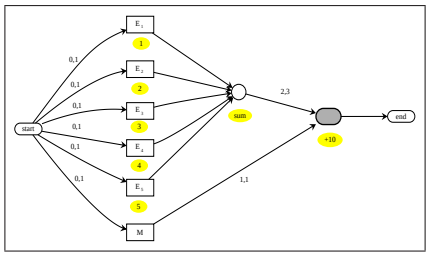


Figure 6: Tbbl valuation network

An important extension of our flows is that if the flow is null on some particular arc then it is not necessary that the lower bounds and capacity constraint to be verified.

Formally a NETBID can be defined as follows.

Definition 1 *A R-**NETBID** is a tuple* $\mathcal{N} = (D, START, END, c, l, \lambda)$*:*

1. $D = (V, E)$ *is an acyclic digraph with two distinguished nodes* $START, END \in V$*; the other nodes,* $V - \{START, END\}$*, are partitioned* $R \cup B \cup I$*: R is the set of resources nodes, B is the set of bundles nodes and I is the set of interior nodes. There is a directed edge* $(START, r) \in E$ *for each* $r \in R$*, and at least a directed edge* $(b, v) \in E$ *for each* $b \in B$*. There are no other directed edges entering in a resource node. The remaining directed edges connect resources nodes to bundle or interior nodes, interior nodes to bundle or interior nodes, bundle nodes to interior nodes or* END *node.*
2. c, l *are nonnegative integer partial functions defined on the set of directed edges of D; if* $(i, j) \in E$ *and c is defined on* (i, j) *then* $c((i, j)) \in \mathbf{Z}_+$*, denoted* c_{ij}*, is the capacity of directed edge* (i, j)*;* $l((i, j)) \in \mathbf{Z}_+$*, if defined, is the lower bound on the directed edge* (i, j) *and is denoted* l_{ij}*; if* (i, j) *has assigned a capacity and a lower bound then* $l_{ij} \leq c_{ij}$*. All directed edges* $(START, r)$ *have capacity 1 and the lower bound 0. No directed edge* (b, END) *has capacity and lower bound.*
3. λ *is a labelling function on* $V - \{START, END\}$ *which assign to a vertex v a pair of rules* $(\lambda_1(v), \lambda_2(v))$ *(which will be described in the next definitions).*

Definition 2 *Let* $\mathcal{N} = (D, START, END, c, l, \lambda)$ *be a R-NETBID. A **bidflow** in* $\mathcal{N}$ *is a function* $f : E(D) \to \mathbf{Z}_+$ *satisfying the following properties (f_{ij} denotes $f((i, j))$):*

1. *For each directed edge* $(i, j) \in E$*: if* $f_{ij} > 0$ *and* c_{ij} *is defined, then* $f_{ij} \leq c_{ij}$*; if* $f_{ij} > 0$ *and* l_{ij} *is defined, then* $f_{ij} \geq l_{ij}$*.*
2. *If* $v \in V - \{START, END\}$ *has* $\lambda_1(v) = conservation$ *then* $\sum_{(i,v)\in E(D)} f_{iv} = \sum_{(v,i)\in E(D)} f_{vi}$*.*
3. *For each* $v \in B$*,* $f_{vu} \in \{0, 1\}$*; there is exactly one vertex u such that* $f_{vu} = 1$ *and this happens if and only if for each* $w \in R \cup I$*, such that* $(w, v) \in E(D)$*, we have* $f_{wv} > 0$*.*

The set of all bidflows in $\mathcal{N}$ *is denoted by* $\mathcal{F}^{\mathcal{N}}$*.*

In order to simplify our presentation we have considered here that for each $v \in V - \{START, END\}$, $\lambda_1(v) \in \{conservation, bundle\}$ giving rise to the flow rules described above. In all the figures considered here, the function $\lambda_1(v)$ is illustrated by the color of the node v: a gray node is a bundle node and a white node is a conservation node. It is possible to use the $\lambda_1(v)$ to have *transformation* internal nodes as [3].

Definition 3 *Let f be a bidflow in the R-NETBID* $\mathcal{N} = (D, START, END, c, l, \lambda)$*. The **value of** f, $val(f)$, is defined as* $val(f) = \sum_{b\in B} val(b) f_{bEND}$*, where $val(v)$ is*

$$val(v) = \begin{cases} 0 & \text{if } v = START \\ \lambda_2(D_f^{-1}(v)) & \text{if } v \neq START, END. \end{cases}$$

$D_f^{-1}(v)$ denotes the set of all vertices $w \in V(D)$ *such that* $(w, v) \in E(D)$ *and* $f_{wv} > 0$*.* $\lambda_2(D_f^{-1}(v))$ *is the rule (specified by the second label associated to vertex v) of computing $val(v)$ from the values of its predecessors which send flows into v.*

Definition 4 *Let* $\mathcal{N} = (D, START, END, c, l, \lambda)$ *be a R-NETBID. The **R-valuation** designated by $\mathcal{N}$ is the function* $v_{\mathcal{N}} : \mathcal{P}(R) \to \mathbf{R}_+$*, where for each* $S \subseteq R$*,* $v_{\mathcal{N}}(S) = \max\{val(f) | f \in \mathcal{F}^{\mathcal{N}}, f_{START\, r} = 0 \, \forall r \in R - S\}$*.*

By the above two definitions, the value associated by $\mathcal{N}$ to a set S of resources is the maximum sum of the values of the (disjoint) bundles which are contained in the set (assignment) S. This is in concordance with the definition of a v-basis given in section 3 for a superadditive valuation v. However, the NETBID structure defined above is more flexible in order to express any valuation. If the bidder desires to express that at most k bundles from some set of bundle nodes must be considered, then these nodes are connected to a new interior node and this last node linked to a new superbundle node by a directed edge having as lower bound 1 and capacity k. Clearly, any valuation represented in a XOR language can be obtained in such way and any R-valuation can be represented.

The NETBIDS submitted by the bidders are merged by the arbitrator in a common NETBID sharing only the nodes corresponding to $START$ and R, and also a common END node in which are projected the corresponding END nodes of the individual NETBIDS. This common NETBID is a symbolic representation of the aggregate valuation of the society. We consider the following definition.

Definition 5 *Let* $\mathcal{N}_i = (D_i, START_i, END_i, c_i, l_i, \lambda_i)$ *be the R-NETBID of the agent* $i \in I$*. The aggregation R-NETBID of* $\{\mathcal{N}_i | i \in I\}$ *is the R-NETBID* $\mathcal{N}_a = (D_a, START_a, END_a, c_a, l_a, \lambda_a)$*, where* $D_a = (V_a, E_a)$ *has* $V_a = \{START_a, END_a\} \cup R \cup B_a \cup I_a$*, B_a (respectively I_a) being the disjoint union of all individual bundle node sets B_i (respectively, internal nodes I_i);* $E_a = \{START_a\} \times R \cup B_a \times \{END_a\} \cup \cup_{i\in I}(E_i - (\{START_i\} \times R \cup B_i \times \{END_i\}))$*.*
All directed edges $(START_a, r)$ *have the capacity* $c_a = 1$ *and the lower bound* $l_a = 0$*. No directed edge* (b, END_a) *has capacity and lower bound. All the remainder capacities and lower bounds are obtained from the corresponding values in the individual NETBIDS. Similarly are constructed the label rules* $\lambda_a(v)$*. If a resource node* r_a *has different* λ_2 *values in some local networks, then* r_a *is connected to new copies of it by directed edges with* $c_a = 1$ *and* $l_a = 0$*, and this new nodes are connected by directed edges corresponding to the local NETBID.*

This definition is illustrated in Figure 7.
From this construction, the following theorem can be proved

Theorem 4 *If in a FRA system each bidder's i R-valuation, v_i, is represented by R-NETBID* $\mathcal{N}_i$ *($i \in I$), then the aggregate R-valuation v_a is designated by the aggregate R-NETBID* $\mathcal{N}_a$*, that is,* $v_a = v_{\mathcal{N}_a}$*.*

Proof. Let $S \subseteq R$. If If $\mathbf{O} = (O_1, \ldots, O_n)$ is a maximum S-allocation, that is, $v_a(S) = \sum_{i\in I} v_i(O_i)$, then for each $i \in \{1, \ldots, n\} = I$, $v_i(O_i) = v_{\mathcal{N}_i}(O_i)$ can be obtained as $val(f_i)$,

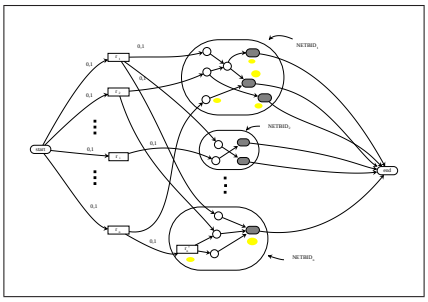


Figure 7: Aggregate NETBID

for a maximum value bidflow f_i in $\mathcal{N}_i$. The flows $(f_i)_{i \in I}$ induce a flow f_a in the NETBID $\mathcal{N}_a$, with $val(f_a) = \sum_{i \in I} v_i(O_i)$, that is $val(f_a) = v_a(S)$. Conversely, each maximum value bidflow f_a in $\mathcal{N}_a$ can be decomposed into disjoint bidflows f_i in the NETBID $\mathcal{N}_i$, which must be maximum value bidflows. It follows that $val(f_a) = \sum_{i \in I} val(f_i) = \sum_{i \in I} v_{\mathcal{N}_i}(S) = v_a(S)$.□

The maximum value of a bidflow in the NETBID $\mathcal{N}_a$ is the social welfare value: the computation of this value implicitly solves the WD problem (by a simple bookkeeping of agents owning the winning bundles in the aggregate NETBID).

5 Discussion

Several bidding languages for CAs have previously been proposed, arguably the most compelling of which allow bidders to explicitly represent the logical structure of their valuation over goods via standard logical operators. These are referred as "logical bidding languages" (e.g. [8]). For instance, an OR bid specifies a set of $< bundle, price >$ pairs, where the bidder is willing to buy any number of the specified bundles for their respectively specified prices. This is equivalent to specifying a set of single-bundle bids. An XOR bid specifies a set of $< bundle, price >$ pairs, where the bidder is willing to pay for only one of the bundles for its corresponding price. Nisan's OR* language [8] provides constraints within an OR bid via "phantom variables" (see also [6]). One explanation of restricting operators to just OR and XOR in the logical framework adopted by these languages, is given by the characteristics of the accompanying WD-solving methodology the language designers proposed. Boutilier and Holger [1] made the next logical step with the LGB language, which allows for arbitrarily nested levels combining goods and bundles by the standard propositional logic operators: OR, XOR, and AND. Day [5] introduces bid tables and bid matrices as a bidding language more connected to the economic literature on restricted preferences and assignment games. Cavallo and colleagues [2], introduce TBBL, a tree-based bidding language that has several novel properties. In TBBL, valuations are expressed in a tree structure, where internal nodes in the tree correspond to operators for combining subsets of goods, and individual goods are represented at the leaves. TBBL allows agents to express preferences for both buying and selling goods in the same tree. Thus, it is applicable to a combinatorial exchange (CE), a generalization of a CA that is important in many multiagent systems. TBBL also provides an explicit semantics for partial value information: a bidder can specify an upper and lower bound on their true valuation, to be refined during bidding. TBBL is a logical tree-based bidding language for CEs. It is fully expressive, yet designed to be as concise and structured as possible. Finally, Cerquides and colleagues [3], explicitly addresses the case of bidding languages for CEs by extending the classical $< bundle, price >$ view of a bid to a $< transformation, price >$ pair. This work is extended by Giovanucci and colleagues [7], which provide an interesting Petri Nets formalism to reason about these CAs extensions.

In this paper we proposed a new visual framework for bidding languages. We have motivated our approach by analyzing adequate resource allocation systems (semantics) and introduced our work in a theoretical manner. We also presented a number of intuitive examples with the purpose of highlighting the advantages of our work. We believe that when bidders are able to express a wide variety of preferences to a sealed-bid or proxy agent, NETBID flows allow to iteratively generate an economically satisfactory market outcome. Moreover, the format for the representation of bidder preferences serves to reinforce the global perspective on the implementation of combinatorial auctions using a new computational technique (CSP based) for determining auction outcomes. We are currently pursuing this line of work for practical evaluation.

References

[1] C. Boutilier and H. Holger. Bidding languages for combinatorial auctions. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*., pages 1211–1217, 2001.

[2] R. Cavallo, D. Parkes, A. Juda, A. Kirsch, A. Kulesza, S. Lahaie, B. Lubin, L. Michael, and J. Shneidman. Tbbl: A tree-based bidding language for iterative combinatorial exchanges. In *International Joint Conferences on A.I.: Workshop on Advances in Preference Handling*, 2005.

[3] J. Cerquides, U. Endriss, A. Giovannucci, and J. Rodriguez-Aguilar. Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In *International Joint Conferences on A.I.*, pages 1221–1226, 2007.

[4] M. Croitoru, P. Lewis, and C. Croitoru. Bidflow: a new graph-based bidding language for combinatorial auctions. In *Proceedings of the 18th European Conference on Artificial Intelligence*, 2008. to appear.

[5] R. Day. *Expressing Preferences with Price-Vector Agents in Combinatorial Auctions*. PhD thesis, University of Maryland, College Park., 2004.

[6] Y. Fujisima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 548–553, 1999.

[7] A. Giovannucci, J. Rodriguez-Aguilar, J. Cerquides, and U. Endriss. Winner determination for mixed multi-unit combinatorial auctions via petri nets. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 2007.

[8] N. Nisan. Bidding and allocations in combinatorial auctions. In *ACM Conference on Electronic Commerce (EC-2000)*, 2000.

[9] M. Rothkopf, A. Pekec, and R. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44:1131–1147, 1998.

A New Approach to Influence Diagrams Evaluation

Radu Marinescu
Cork Constraint Computation Centre
University College Cork, Ireland
r.marinescu@4c.ucc.ie

Abstract

Influence diagrams are a widely used framework for decision making under uncertainty. The paper presents a new algorithm for maximizing the expected utility over a set of policies by traversing an AND/OR search space associated with an influence diagram. AND/OR search spaces accommodate advanced algorithmic schemes for graphical models which can exploit the structure of the problem. The algorithm also exploits the deterministic information encoded by the influence diagram and avoids redundant computations for infeasible decision choices. We demonstrate empirically the effectiveness of the AND/OR search approach on various benchmarks for influence diagrams.

1 Introduction

An influence diagram is a graphical model for decision making under uncertainty. It is composed by a directed acyclic graph where utility nodes are associated to profits and costs of actions, chance nodes represent uncertainties and dependencies in the domain and decision nodes represents actions to be taken. Given an influence diagram, a policy (or strategy) defines which decision to take at each node, given the information available at that moment. Each policy has a corresponding expected utility and the most common task is to find an optimal policy with maximum expected utility.

Over the past decades, several exact methods have been proposed to solve influence diagrams using local computations [Tatman and Shachter, 1990; Shenoy, 1992; Jensen *et al.*, 1994; Dechter, 2000]. These methods adapted classical *variable elimination* techniques, which compute a type of marginalization over a combination of local functions, in order to handle the multiple types of information (probabilities and utilities), marginalizations (sum and max) and combinations ($\times$ for probabilities, $+$ for utilities) involved in influence diagrams. Variable elimination based techniques are known to exploit the conditional independencies encoded by the influence diagram, however, they require time and space exponential in the *constrained induced-width* of the diagram.

An alternative approach for evaluating influence diagrams is based on *conditioning* (or *search*). These methods unfold the influence diagram into a *decision graph* (or *tree*) in such a way that an optimal solution graph corresponds to an optimal policy of the influence diagram. In this case, the problem of computing an optimal policy is reduced to *searching* for an optimal solution of the decision graph [Howard and Matheson, 1984; Pearl, 1988; Qi and Poole, 1995]. In contrast with variable elimination, search algorithms are not sensitive to the problem structure and do not accommodate informative performance guarantees.

This situation has changed in the past few years with the introduction of AND/OR search spaces for graphical models as a paradigm for search algorithms that exploit the problem structure [Dechter and Mateescu, 2007]. In this paper, we extend the AND/OR search space to influence diagrams and develop a depth-first search algorithm that explores a context-minimal AND/OR graph for computing the optimal policy that maximizes the expected utility. Traversing the AND/OR graph allows search algorithms to achieve the same worst case time and space performance guarantees as variable elimination. It also allows a better exploitation of the deterministic information encoded by the influence diagram, thus avoiding redundant computations for impossible decision choices, as well as a better trade-off between time and space. Our experiments show that the new AND/OR search approach improves significantly over state-of-the-art algorithms, in some cases by two orders of magnitude of improved performance.

Following background on influence diagrams (Section 2), Section 3 presents the AND/OR search space for influence diagrams. In Section 4 we describe the AND/OR search algorithm for computing the optimal policy. Section 5 is dedicated to our empirical evaluation, while Section 6 concludes.

2 Background

2.1 Influence Diagrams

An *influence diagram* (ID) [Howard and Matheson, 1984] is defined by $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{R} \rangle$, where $\mathbf{X} = \{X_1, ..., X_n\}$ is a set of *chance* variables on multi-valued domains and $\mathbf{D} = \{D_1, ..., D_m\}$ (indices represent the order in which decisions are made) is a set of *decision* variables. The discrete domain of a decision variable denotes its possible set of actions. Every chance variable $X_i \in \mathbf{X}$ is associated with a conditional probability table (CPT), $P_i = P(X_i|pa(X_i))$, $pa(X_i) \subseteq \mathbf{X} \cup \mathbf{D} - \{X_i\}$. Each decision variable $D_i \in \mathbf{D}$ has a parent set $pa(D_i) \subseteq \mathbf{X} \cup \mathbf{D} - \{D_i\}$ denoting the set

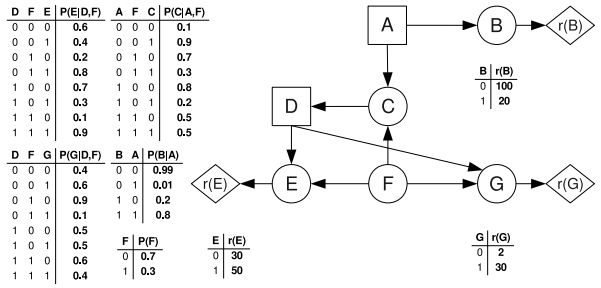


Figure 1: A simple influence diagram.

of variables that will be observed before decision D_i is made. The *reward* (or *utility*) functions $\mathbf{R} = \{r_1, ..., r_j\}$ are defined over subsets of variables $\mathbf{Q} = \{Q_1, ..., Q_j\}$, $Q_i \subseteq \mathbf{X} \cup \mathbf{D}$, called *scopes*. The directed acyclic graph of an ID contains nodes for the chance variables (depicted as circles) and decision variables (depicted as squares) as well as for the reward components (depicted as diamonds). For each chance or decision node there is an arc directed from each of its parent variables to it, and there is an arc directed from each variable in the scope of a reward component towards its reward node.

Given a temporal order of the decisions, an influence diagram induces a partial order $\prec$ on its variables. The set of chance variables observed before the first decision is denoted $\mathbf{I}_0$, the set of chance variables observed between decisions D_k and D_{k+1} is denoted $\mathbf{I}_k$, and the set of chance variables unobserved before the last decision is denoted $\mathbf{I}_m$. The partial order $\prec$ is: $\mathbf{I}_0 \prec D_1 \prec \mathbf{I}_1 \prec ... \prec D_m \prec \mathbf{I}_m$.

A *decision rule* for a decision variable $D_i \in \mathbf{D}$ is a mapping: $\delta_i : \Omega_{pa(D_i)} \rightarrow \Omega_{D_i}$, where Ω_S is the cross product of the individual domains of the variables in $S \subseteq \mathbf{X} \cup \mathbf{D}$. A *policy* is a list of decision rules $\Delta = (\delta_1, ..., \delta_m)$ consisting of one rule for each decision variable. To *evaluate* an influence diagram is to find an *optimal policy* maximizing the expected utility. As shown in [Jensen *et al.*, 1994], this is equivalent to computing optimal decision rules for the quantity:

$$\sum_{\mathbf{I}_0} \max_{D_1} ... \sum_{\mathbf{I}_{m-1}} \max_{D_m} \sum_{\mathbf{I}_m} ((\prod_{P_i \in \mathbf{P}} P_i) \times (\sum_{r_i \in \mathbf{R}} r_i)) \quad (1)$$

With every ID instance we can associate a *primal graph* which is obtained from the ID graph as follows. All the parents of chance variables are connected, all the parents of reward components are connected, and all the arrows are dropped. Reward nodes and their incident arcs are deleted.

In addition, influence diagrams must be non forgetting in the sense that a decision node and its parents be parents to all subsequent decision nodes. The rational behind the no-forgetting constraint is that information available now should be available later if the decision-maker does not forget. In this paper we do not enforce this restriction.

Example 1 *Figure 1 shows an influence diagram with two decisions and five chance variables. The utility function is the sum of three local utilities defined on single variables (B, E and G, respectively). The partial order $\prec$ is $\{A, C, D, B, E, F, G\}$. Evaluating the influence diagram is to find the two optimal decision rules δ_A^* and δ_D^* for: $max_A \sum_C max_D \sum_{B,E,F,G} P(B|A) \cdot P(C|A,F) \cdot P(E|D,F) \cdot P(F) \cdot P(G|D,F) \cdot (r(B) + r(E) + r(G))$.*

2.2 Variable Elimination for Influence Diagrams

Variable elimination algorithms are characteristic of inference methods for evaluating influence diagrams. This approach reformulates Equation 1 using so-called *potentials* [Jensen *et al.*, 1994], in order to use one combination and one marginalization operator. A potential on a set of variables S is a pair $\Psi_S = (\lambda_S, \theta_S)$ of real-valued functions on Ω_S, where λ_S is non-negative. The initial conditional probability tables $P_i \in \mathbf{P}$ and utility functions $r_i \in \mathbf{R}$ are transformed into potentials $(P_i, 0)$ and $(1, r_i)$, respectively. A *combination* operator $\otimes$ and a *marginalization* (or *elimination*) operator $\downarrow$ are defined on these potentials, as follows: (a) the **combination** of $\Psi_{S_1} = (\lambda_{S_1}, \theta_{S_1})$ and $\Psi_{S_2} = (\lambda_{S_2}, \theta_{S_2})$ is the potential on $S_1 \cup S_2$ given by $\Psi_{S_1} \otimes \Psi_{S_2} = (\lambda_{S_1} \cdot \lambda_{S_2}, \theta_{S_1} + \theta_{S_2})$; (b) the **marginalization** of $\Psi_S = (\lambda_S, \theta_S)$ onto $S_1 \in \mathbf{X}$ is $\Psi_S^{\downarrow S_1} = (\sum_{S-S_1} \lambda_S, \frac{\sum_{S-S_1} \lambda_S \cdot \theta_S}{\sum_{S-S_1} \lambda_S})$ (assuming that $0/0 = 0$), whereas $\Psi_S^{\downarrow S_1} = (\lambda_S, max_{S_1} \theta_S)$ for $S_1 \in \mathbf{D}$.

Evaluating an influence diagram is then equivalent to computing $Q = ((...((\Psi_{\mathbf{X}\cup\mathbf{D}}^{\downarrow \mathbf{I}_m})^{\downarrow D_m})^{\downarrow \mathbf{I}_{m-1}} ...)^{\downarrow D_1})^{\downarrow \mathbf{I}_0}$, where $\Psi_{\mathbf{X}\cup\mathbf{D}} = (\otimes_{P_i \in \mathbf{P}}(P_i, 0)) \otimes (\otimes_{r_i \in \mathbf{R}}(1, r_i))$ is the combination of the initial potentials, which can be done using usual variable elimination algorithms [Jensen *et al.*, 1994; Dechter, 2000]. Since the alternation of *sum* and *max* marginalizations does not commute in general, it prevents from eliminating variables in any order. Therefore, the computation of Q must be performed along *valid elimination orderings* that respect $\prec$, namely the reverse of the elimination order is some extension of $\prec$ to a total order [Jensen *et al.*, 1994]. The performance of variable elimination algorithms can be bounded as a function of the induced-width of the *induced graph* [Dechter, 2000] that reflects the algorithm's execution. Given an ID with primal graph G, variable elimination is time and space $O(N \cdot k^{w^*(o)})$, where $w^*(o)$ is the induced-width obtained along a valid elimination ordering o of G ($i.e.$, *constrained induced-width*), k bounds the domain size and N is the number of variables [Dechter, 2000].

3 AND/OR Search Spaces for IDs

In this section we specialize the AND/OR search space for general graphical models to influence diagrams. AND/OR search spaces accommodate advanced algorithmic schemes for graphical models which can exploit the structure of the model [Dechter and Mateescu, 2007]. Given an ID with primal graph G, its AND/OR search space is based on a *pseudo tree* arrangement of G.

DEFINITION **1 (pseudo tree)** *Let $G = (V, E)$ be the primal graph of an influence diagram $\mathcal{M}$. A directed rooted tree $\mathcal{T} = (V, E')$ is a* pseudo tree *if: (i) any arc of G which is not included in E' is a back-arc, namely it connects a node to an ancestor in $\mathcal{T}$; (ii) the ordering obtained from a depth-first traversal of $\mathcal{T}$ is an extension of $\prec$ to a total order.*

3.1 AND/OR Search Tree

Given an influence diagram $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{R} \rangle$, its primal graph G and a pseudo tree $\mathcal{T}$ of G, the associated AND/OR search tree, denoted $S_{\mathcal{T}}(\mathcal{M})$, has alternating levels of OR and

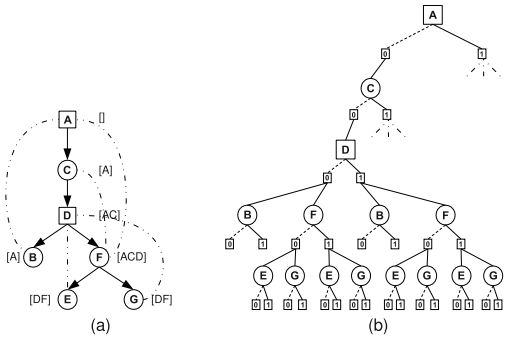


Figure 2: AND/OR search tree.

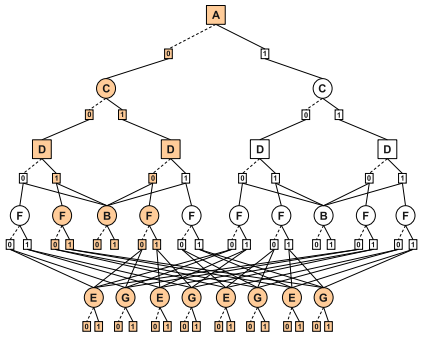


Figure 3: Context-minimal AND/OR search graph.

AND nodes. The OR nodes are labeled X_i (resp. D_i) and correspond to the variables. The AND nodes are labeled $\langle X_i, x_i \rangle$ (resp. $\langle D_i, d_i \rangle$) and correspond to the values in the domains of the variables. The structure of the AND/OR search tree is based on the underlying pseudo tree $\mathcal{T}$. The root of $S_{\mathcal{T}}(\mathcal{M})$ is an OR node labeled with the root of $\mathcal{T}$. The children of an OR node X_i (resp. D_i) are AND nodes labeled with assignments $\langle X_i, x_i \rangle$ (resp. $\langle D_i, d_i \rangle$). The children of an AND node $\langle X_i, x_i \rangle$ (resp. $\langle D_i, d_i \rangle$) are OR nodes labeled with the children of variable X_i (resp. D_i) in the pseudo tree $\mathcal{T}$. A node $n \in S_{\mathcal{T}}(\mathcal{M})$ is called a *chance node* if it is labeled by X_i or $\langle X_i, x_i \rangle$. If n is labeled D_i or $\langle D_i, d_i \rangle$, then it is called a *decision node*.

Example 2 *Consider again the influence diagram from Figure 1. Figure 2(a) shows a pseudo tree of its primal graph, together with the back-arcs (dotted lines). Figure 2(b) shows a portion the AND/OR search tree based on the pseudo tree.*

3.2 AND/OR Search Graph

Often different nodes in the AND/OR search tree root identical subtrees, and correspond to identical subproblems. Any two such nodes can be merged, reducing the size of the search space and converting it into a graph. Some of these mergeable nodes can be identified based on *contexts* [Dechter and Mateescu, 2007]. Given a pseudo tree $\mathcal{T}$, the context of an OR node labeled Y_i, where $Y_i \in \mathbf{X} \cup \mathbf{D}$, is defined as the set of ancestors of Y_i (in $\mathcal{T}$), ordered descendingly, that are connected (in the induced graph) to Y_i or to descendants of Y_i (in $\mathcal{T}$). It is easy to verify that $context(Y_i)$ separates in the primal graph (and also in the induced graph) the ancestors (in $\mathcal{T}$) of Y_i, from Y_i and its descendants (in $\mathcal{T}$). The *context-minimal AND/OR graph*, $C_{\mathcal{T}}(\mathcal{M})$, is obtained from the AND/OR tree by merging all context mergeable OR nodes. Based on earlier work [Dechter and Mateescu, 2007], it can be shown that the size of $C_{\mathcal{T}}(\mathcal{M})$ relative to a pseudo tree $\mathcal{T}$ is $O(N \cdot k^{w^*_{\mathcal{T}}(G)})$, where $w^*_{\mathcal{T}}(G)$ is the induced-width of G over a depth-first traversal of $\mathcal{T}$ ($i.e.$, constrained induced-width).

Example 3 *Figure 3 shows the context-minimal AND/OR graph relative to the pseudo tree from Figure 2(a). The OR contexts of the variables are indicated in square brackets next to each node in the pseudo tree.*

3.3 Arc Labeling

The arcs from Y_i to $\langle Y_i, y_i \rangle$, where $Y_i \in \mathbf{X} \cup \mathbf{D}$, are labeled with the appropriate combined values of the functions in $\mathbf{P} \cup \mathbf{R}$ that contain Y_i and have their scopes fully assigned. It is convenient to group the functions of the influence diagram into *buckets* relative to its pseudo tree, as follows. The bucket $B_{\mathcal{T}}(X_i)$ of a chance variable $X_i \in \mathbf{X}$ is the set of probability and reward functions (if X_i has no decision variables as descendants in $\mathcal{T}$) whose scopes contain X_i and are included in the path from root to X_i in $\mathcal{T}$. The bucket $B_{\mathcal{T}}(D_i)$ of a decision variable $D_i \in \mathbf{D}$ is the set of probability functions whose scopes contain D_i and are included in the path from root to D_i in $\mathcal{T}$. If D_i is the deepest decision variable in $\mathcal{T}$, then $B_{\mathcal{T}}(D_i)$ contains also the set of all remaining reward functions. In each bucket, $B^{\lambda}_{\mathcal{T}}(Y_i)$ and $B^{\theta}_{\mathcal{T}}(Y_i)$ denote probabilistic and reward components, respectively.

DEFINITION **2 (weights)** *Given an AND/OR search graph $C_{\mathcal{T}}(\mathcal{M})$ of an influence diagram $\mathcal{M}$, the* weight *of the arc (n, m), where Y_i labels n and $\langle Y_i, y_i \rangle$ labels m, is a pair (w_λ, w_θ) where w_λ (or $w_\lambda(n, m)$) is the* product *of all the probability functions in $B^{\lambda}_{\mathcal{T}}(Y_i)$ and w_θ (or $w_\theta(n, m)$) is the* sum *of all the utility functions in $B^{\theta}_{\mathcal{T}}(Y_i)$. Formally, $w_\lambda = \prod_{\lambda \in B^{\lambda}_{\mathcal{T}}(Y_i)} \lambda(asgn(\pi_m))$ and $w_\lambda = 1$ if $B^{\lambda}_{\mathcal{T}}(Y_i) = \emptyset$, while $w_\theta = \sum_{\theta \in B^{\theta}_{\mathcal{T}}(Y_i)} \theta(asgn(\pi_m))$ and $w_\theta = 0$ if $B^{\theta}_{\mathcal{T}}(Y_i) = \emptyset$, where $asgn(\pi_m)$ denotes the assignment along the path π_m from the root of $C_{\mathcal{T}}(\mathcal{M})$ to the AND node m.*

Example 4 *Consider again the influence diagram from Figure 1 with partial order (A, C, D, BE, F, G). In this case for example, the bucket of E contains the probability function $P(E|D, F)$ and the reward function $r(E)$ and, therefore, the weight (w_λ, w_θ) on the arcs from the OR node E to any of its AND value assignments include only the instantiated functions $P(E|D, F)$ and $r(E)$. Notice also that the buckets of A, C and D do not contain any functions and therefore the weights associated with the respective arcs are $(1, 0)$.*

3.4 Value Functions

With each node n in the weighted AND/OR graph $C_{\mathcal{T}}(\mathcal{M})$, we associate a *probability value* $\lambda(n)$ and a *utility value* $\theta(n)$ defined on the subspaces they root.

DEFINITION **3 (values of a chance node)** *The values $\lambda(n)$ and $\theta(n)$ of a chance node $n \in C_{\mathcal{T}}(\mathcal{M})$ are defined recursively as follows: (i) if n labeled $\langle X_i, x_i \rangle$ is a terminal AND node, then $\lambda(n) = 1$ and $\theta(n) = 0$; (ii) if n labeled $\langle X_i, x_i \rangle$ is an internal AND node, then $\lambda(n) = \prod_{m \in succ(n)} \lambda(m)$ and $\theta(n) = \sum_{m \in succ(n)} \frac{\theta(m)}{\lambda(m)}$; (iii) if n labeled X_i is an inter-*

nal OR node then $\lambda(n) = \sum_{m \in succ(n)} w_\lambda(n,m) \cdot \lambda(m)$ *and* $\theta(n) = \sum_{m \in succ(n)} w_\lambda(n,m) \cdot \lambda(n,m) \cdot (w_\theta(n,m) + \theta(m))$, *where* $succ(n)$ *are the children of* n *in* $C_\mathcal{T}(\mathcal{M})$.

DEFINITION **4 (values of a decision node)** *The values* $\lambda(n)$ *and* $\theta(n)$ *of a decision node* n *are defined recursively, as follows: (i) if* n *labeled* $\langle D_i, d_i \rangle$ *is a terminal AND node, then* $\lambda(n) = 1$ *and* $\theta(n) = 0$*; (ii) if* n *labeled* $\langle D_i, d_i \rangle$ *is an internal AND node, then* $\lambda(n) = \prod_{m \in succ(n)} \lambda(m)$ *and* $\theta(n) = \sum_{m \in succ(n)} \frac{\theta(m)}{\lambda(m)}$*; (iii) if* n *labeled* D_i *is an internal OR node, then* $\lambda(n) = max_{m \in succ(n)} w_\lambda(n,m) \cdot \lambda(m)$ *and* $\theta(n) = max_{m \in succ(n)} w_\lambda(n,m) \cdot \lambda(n,m) \cdot (w_\theta(n,m) + \theta(m))$, *where* $succ(n)$ *are the children of* n *in* $C_\mathcal{T}(\mathcal{M})$.

Clearly, the λ and θ-values of each node can be computed recursively, from leaves to root. If n is the root node of $C_\mathcal{T}(\mathcal{M})$, then $\theta(n)$ is the maximum expected utility of the initial problem. Alternatively, the value $\theta(n)$ can also be interpreted as the expected utility (for chance nodes) or maximum expected utility (for decision nodes) of the conditioned subproblem rooted at n.

3.5 Policy Graphs and Decision Rules

The context-minimal AND/OR graph $C_\mathcal{T}(\mathcal{M})$ of an influence diagram $\mathcal{M}$ contains the set of all policies for $\mathcal{M}$. A policy Δ is represented in $C_\mathcal{T}(\mathcal{M})$ by a *policy graph*, which is an AND/OR subgraph, denoted by $\mathcal{G}_\Delta(\mathcal{M})$, such that: (a) it contains the root s of $C_\mathcal{T}(\mathcal{M})$; (b) if a non-terminal *chance* OR node n is in $\mathcal{G}_\Delta(\mathcal{M})$ then all of its children are in $\mathcal{G}_\Delta(\mathcal{M})$; (c) if a non-terminal *decision* OR node is in $\mathcal{G}_\Delta(\mathcal{M})$ then exactly one of its children is in $\mathcal{G}_\Delta(\mathcal{M})$; (d) if a non-terminal AND node is in $\mathcal{G}_\Delta(\mathcal{M})$ then all its children are in $\mathcal{G}_\Delta(\mathcal{M})$. Given a policy graph $\mathcal{G}_\Delta(\mathcal{M})$ with appropriate weights on its arcs, the value $\theta(s)$ of the root node s is the expected utility of the policy Δ. Therefore, the optimal policy for $\mathcal{M}$ corresponds to the policy graph with maximum expected utility.

Moreover, it is easy to see that for any decision variable $D_i \in \mathbf{D}$ its $context(D_i)$ contains the set of variables that may affect directly the decision and therefore it defines the scope of the decision rule δ_i associated with D_i. For illustration, consider the policy graph highlighted in Figure 3. The two decision rules δ_A and δ_D can be read from the graph, as follows: δ_A: $A = 0$, δ_D: $D = 1$ if $(A = 0, C = 0)$ and $D = 0$ if $(A = 0, C = 1)$, respectively.

Search algorithms that traverse the AND/OR graph can be used to compute the optimal policy graph yielding the answer to the problem as we will describe in the next section.

4 Depth-First AND/OR Graph Search

A depth-first search algorithm, called AO-ID, that traverses the context-minimal AND/OR graph and computes the values of each node in the search space is described in Algorithm 1. The following notation is used: $\mathcal{M}$ is the problem with which the procedure is called and $\mathcal{T}$ is the pseudo tree that drives the AND/OR search graph. The algorithm assumes that variables are selected according to the pseudo tree arrangement. If $\mathcal{M}$ is empty, then the result is trivially computed (line 1). Else, AO-ID selects a variable Y_i ($i.e.$, expands the OR node n labeled Y_i) and iterates over its values (line 10) to compute the

Algorithm 1: AO-ID($\mathcal{M}$): Depth-first AND/OR search.

```
1  if M = ∅ then return (1, 0);
2  else
3      choose a variable Y_i ∈ Y;
4      let n be an OR node labeled Y_i;
5      {λ(n), θ(n)} ← ReadCache(Y_i, context(Y_i));
6      if {λ(n), θ(n)} ≠ NULL then return {λ(n), θ(n)};
7      else
8          if Y_i is a decision node then {λ(n), θ(n)} ← (−∞, −∞);
9          else if Y_i is a chance node then {λ(n), θ(n)} ← (1, 0);
10         foreach y_i ∈ Domain(Y_i) do
11             let m be an AND node labeled ⟨Y_i, y_i⟩;
12             {λ(m), θ(m)} ← (1, 0);
13             foreach k = 1..q do
14                 {λ, θ} ← AO-ID(M_k);
15                 λ(m) ← λ(m) · λ;
16                 θ(m) ← θ(m) + θ/λ;
17             if Y_i is a decision node then
18                 λ(n) ← max(λ(n), w_λ(n, m) · λ(m));
19                 θ(n) ← max(θ(n), w_λ(n, m) · λ(m) · (w_θ(n, m) + θ(m)));
20             else if Y_i is a chance node then
21                 λ(n) += w_λ(n, m) · λ(m);
22                 θ(n) += w_λ(n, m) · λ(m) · (w_θ(n, m) + θ(m));
23         WriteCache(Y_i, context(Y_i), {λ(n), θ(n)});
24         return {λ(n), θ(n)}
```

OR values $\{\lambda(n), \theta(n)\}$. The algorithm first attempts to retrieve the results cached at the OR nodes (line 5). If a valid cache entry is found for the current OR node n then the OR values $\{\lambda(n), \theta(n)\}$ are updated (line 6) and the search continues with the next variable. The context-based caching uses table representation. For each variable Y_i, a *cache table* is reserved in memory for each possible assignment to its context. During search, each table entry records the λ and θ-values below the corresponding OR node (for decision nodes, the table entry also records the argmax of the corresponding θ-value).

When AO-ID expands the AND node m labeled $\langle Y_i, y_i \rangle$ the problem is decomposed into a set of q independent subproblems ($\mathcal{M}_k$), one for each child Y_k of Y_i in $\mathcal{T}$. These subproblems are solved sequentially (lines 13-16) and the solutions accumulated by the AND values $\{\lambda(m), \theta(m)\}$ (lines 15-16). After trying all feasible values of Y_i, the solution to the subproblem below Y_i remains in $\{\lambda(n), \theta(n)\}$ which are first saved in cache (line 23) and then returned (line 24).

Extracting the Optimal Decision Rules Once AO-ID terminates and returns the maximum expected utility $\mathcal{E}$, the optimal policy $\Delta^* = (\delta_1^*, ..., \delta_m^*)$ corresponding to $\mathcal{E}$ is obtained by processing the decision variables from first to last, as follows. Let D_i be the current decision variable. Its optimal decision rule, δ_i^*, is a function defined on $context(D_i)$ and maps every instantiation of $context(D_i)$ that is consistent with the previously computed optimal decision rules $(\delta_1^*, ..., \delta_{i-1}^*)$, to the corresponding cache entry recorded by AO-ID for D_i ($i.e.$, the optimal decision d_i for D_i).

Exploiting Determinism Often the functions of an influence diagram may encode deterministic relationships ($i.e.$, hard constraints). Some of these constraints are represented by the zero-probability entries of the probability tables. In this case, it is beneficial to exploit the computational power of the constraints explicitly, via constraint propagation [Dechter and Mateescu, 2007]. The approach we take for handling the

determinism is based on the known technique of *unit resolution* for Boolean satisfiability (SAT) over a knowledge base (KB) in the form of propositional clauses (CNF) representing the constraints. One way for encoding constraints as a CNF formula is the *direct encoding* [Walsh, 2000].

The changes needed in Algorithm 1 are then as follows. Upon expanding an AND node, its corresponding SAT instantiation is asserted. If unit resolution leads to a contradiction, then the current AND node is marked as dead-end and the search continues by expanding the next node on the search stack. Whenever AO-ID backtracks to the previous level, it also retracts any SAT instantiation recorded by unit resolution. Notice that the algorithm is capable of pruning the domains of future variables in the current subproblem due to conflicts detected during unit propagation. In summary,

THEOREM **1 (complexity)** *Given an ID with primal graph G and a pseudo tree $\mathcal{T}$ of G, algorithm AO-ID guided by $\mathcal{T}$ is sound and complete. Its time and space complexity is $O(N \cdot k^{w^*_{\mathcal{T}}(G)})$, where $w^*_{\mathcal{T}}(G)$ is the constrained induced-width.*

5 Experiments

In this section, we compare empirically the AND/OR search approach against state-of-the-art algorithms for exact evaluation of influence diagrams. We consider two AND/OR search algorithms that explore the context minimal AND/OR graph and exploit the determinism that may be present in the influence diagram using constraint propagation. They are denoted by AO-ID+SAT and AO-ID+BCP, respectively. AO-ID+BCP is conservative and applies only unit resolution over the CNF that encodes the determinism, at each node in the search graph, whereas AO-ID+SAT is more aggressive and detects inconsistency by running a full SAT solver. We used the **minisat** solver (*http://minisat.se/*) for both unit resolution as well as full satisfiability. For reference, we also ran the AND/OR graph search algorithm without constraint propagation, denoted by AO-ID. In all our experiments, the pseudo trees that guided the AND/OR search algorithms were generated using the *min-fill* heuristic [Dechter and Mateescu, 2007].

The competing approaches are: (i) the bucket elimination (BE) algorithm [Dechter, 2000] and (ii) the policy evaluation algorithm [Cooper, 1988] available from the Genie/Smile system (*http://genie.sis.pitt.edu*). The latter converts the influence diagram into a Bayesian network and finds the expected utilities of each of the decision alternatives by performing repeated exact inference in this network. We also note that the variable elimination algorithm by [Jensen *et al.*, 1994] which is available in the commercial Hugin shell (*www.hugin.com*) is equivalent with BE [Dechter, 2000].

Random Influence Diagrams We generated a set of 150 random influence diagrams based on the total number of nodes (N) and the number of decision nodes (d). The configurations chosen are shown in the first column of Table 1. We have from 40 to 160 nodes, 10 decision nodes and 5 utility functions (u), respectively. Each of the chance and decision variables had two parents chosen randomly, ensuring that the ID graph had no cycles and the decision nodes were connected by a directed path in the graph. The fraction of chance nodes that are assigned deterministic CPTs is a parameter, called the *deterministic ratio*. The CPTs for these nodes were randomly filled with 0 and 1; in the remaining nodes, the CPTs were randomly filled using a uniform distribution. Each utility function was defined over 3 randomly chosen variables (out of N), and its corresponding table was filled with integers drawn uniformly at random between 1 and 100, respectively. The domain size (k) of each variable is 2.

Random influence diagrams, deterministic ratio 0.50, 10 instances for each entry						
size (N,d,u,k)	(w*, h)	BE	Smile	AO-ID	AO-ID+SAT	AO-ID+BCP
(40,10,5,2)	(16, 26)	**1.12** (10)	24.32 (10)	12.65 (10)	30.47 (10)	7.88 (10)
(60,10,5,2)	(22, 34)	**57.68** (10)	86.35 (6)	542.09 (10)	250.53 (10)	182.64 (10)
(80,10,5,2)	(26, 43)	431.62 (5)	1295.51 (3)	3156.99 (6)	2294.49 (6)	**1312.71** (6)
(100,10,5,2)	(31, 48)	-	**7196.51** (1)	-	-	-
Random influence diagrams, deterministic ratio 0.75, 10 instances for each entry						
size (N,d,u,k)	(w*, h)	BE	Smile	AO-ID	AO-ID+SAT	AO-ID+BCP
(40,10,5,2)	(16, 26)	**0.61** (10)	1.89 (10)	2.08 (10)	1.19 (10)	0.87 (10)
(60,10,5,2)	(22, 34)	66.53 (10)	82.16 (8)	392.37 (10)	50.51 (10)	**23.66** (10)
(80,10,5,2)	(26, 43)	358.85 (5)	74.52 (4)	2448.75 (10)	**129.00** (10)	164.06 (10)
(100,10,5,2)	(31, 48)	-	88.57 (2)	-	1024.18 (10)	**675.09** (10)
Random influence diagrams, deterministic ratio 0.90, 10 instances for each entry						
size (N,d,u,k)	(w*, h)	BE	Smile	AO-ID	AO-ID+SAT	AO-ID+BCP
(40,10,5,2)	(17, 27)	0.65 (10)	1.67 (10)	1.11 (10)	0.10 (10)	**0.03** (10)
(60,10,5,2)	(23, 34)	52.86 (10)	30.65 (9)	101.48 (10)	1.09 (10)	**0.37** (10)
(80,10,5,2)	(27, 42)	480.99 (4)	73.46 (6)	1711.81 (10)	3.06 (10)	**0.88** (10)
(100,10,5,2)	(31, 46)	516.59 (1)	26.03 (2)	-	9.12 (10)	**3.07** (10)
(120,10,5,2)	(36, 55)	-	-	-	23.92 (10)	**7.85** (10)
(140,10,5,2)	(39, 58)	-	30.98 (1)	-	97.46 (10)	**30.64** (10)
(160,10,5,2)	(43, 66)	-	-	-	140.26 (10)	**62.68** (10)

Table 1: Median CPU times in seconds on 10 examples of random influence diagrams at each size. Time limit 2 hours and 2GB of RAM. '-' stands for time-out or out-of-memory.

Each row in Table 1 contains the median CPU time in seconds, as well as the median induced width (w^*) and depth of the pseudo tree (h) obtained for 10 randomly generated diagrams with that configuration. A number in parenthesis (next to the CPU time) indicates only that many instances out of 10 were solved within the time or memory limit. Also, the table is organized into three horizontal blocks, each corresponding to a specific value of the deterministic ratio. Not surprisingly, BE and AO-ID were able to solve only the smallest instances and they ran out of memory due to higher induced widths on larger problems. On the other hand, AO-ID+SAT and AO-ID+BCP, which exploit efficiently the determinism present in the diagrams, offer the overall best performance on this domain. Both methods scaled to much larger problem instances than their competitors, especially for the 0.90 deterministic ratio. For example, on the $(100, 10, 5, 2)$ configuration with 0.90 deterministic ratio, BE solved one instance (in 516.59 sec), while Smile solved 2 out of 10 instances (in 26.03 sec). AO-ID+SAT and AO-ID+BCP solved all 10 instances of this problem class using 9.12 and 3.07 seconds, respectively. We also see that AO-ID+BCP was consistently faster than AO-ID+SAT. This was due to lightweight constraint propagation scheme used by the former. Notice that Smile is competitive with AO-ID+BCP, however it solved about half as many problem instances as AO-ID+BCP (72 versus 136).

Real-World Benchmarks These influence diagrams are based on ground instances of real-world Bayesian networks from the UCI Graphical Models repository (*available at: http://graphmod.ics.uci.edu/group/Repository*). For our purpose, we converted each of these networks into an influence diagram by choosing at random d out of N variables to act

Influence diagrams derived from real-world Bayesian networks, 10 instances for each entry, 2 hour time limit and 2GB of RAM								
network	(N, d, u, k)	(w*, h)	(literals, clauses)	BE	Smile	AO-ID	AO-ID+SAT	AO-ID+BCP
90-10-1	(90, 10, 5, 2)	(25, 48)	(200, 482)	346.43 (8)	34.81 (10)	2262.48 (8)	20.75 (10)	**6.47** (10)
90-14-1	(186, 10, 5, 2)	(32, 73)	(392, 1032)	-	35.38 (7)	-	353.39 (10)	**100.84** (10)
90-16-1	(246, 10, 5, 2)	(32, 95)	(512, 1336)	-	1292.21(5)	-	3568.53 (5)	**674.89** (9)
blockmap_05_01	(690, 10, 5, 2)	(26, 89)	(1400, 3591)	526.17 (10)	1223.95 (10)	2981.95 (4)	5.01 (10)	**0.69** (10)
blockmap_05_02	(845, 10, 5, 2)	(27, 85)	(1710, 4448)	902.13 (2)	5111.03 (3)	-	20.93 (10)	**2.30** (10)
blockmap_05_03	(995, 10, 5, 2)	(28, 120)	(2010, 5272)	892.21 (1)	4027.92 (2)	-	45.40 (10)	**3.35** (10)
hailfinder	(51, 5, 5, 11)	(10, 19)	(223, 874)	**1.15** (10)	37.29 (10)	13.35 (10)	69.64 (8)	15.17 (10)
insurance	(22, 5, 5, 5)	(9, 17)	(89, 367)	**0.45** (10)	6.56 (10)	9.68 (10)	32.09 (10)	8.51 (10)
mastermind_03_08_03	(1205, 15, 10, 2)	(25, 111)	(2440, 6506)	440.22 (6)	6125.32 (1)	5233.73 (1)	967.17 (10)	**40.96** (10)
mastermind_04_08_03	(1408, 15, 10, 2)	(31, 124)	(2836, 7628)	-	-	-	3833.30 (3)	**391.45** (8)
pathfinder	(106, 3, 5, 63)	(6, 15)	(448, 44656)	**0.14** (10)	1.72 (10)	1.60 (10)	1750.01 (10)	2.28 (10)
s386	(162, 10, 5, 2)	(21, 44)	(344, 1200)	34.70 (10)	4.27 (10)	21.02 (10)	26.82 (10)	**3.38** (10)
water	(29, 3, 5, 4)	(13, 22)	(116, 3649)	123.36 (9)	26.36 (10)	99.50 (10)	43.08 (10)	**11.13** (10)

Table 2: Median CPU times in seconds on influence diagrams derived from real-world Bayesian networks.

as decisions and adding u ternary reward functions as in the case of random influence diagrams. Table 2 displays the results obtained on this dataset, where each row shows the median CPU time over 10 instances that were generated for each network by randomizing the choice of decision nodes. In addition to the induced-width (w^*) and depth (h) of the pseudo trees, we also record the median size of the CNF encoding of the zero-probability CPT entries (column 4). As before, the numbers in parenthesis (next to the CPU time) indicate how many instances out of 10 were solved. We see that AO-ID+BCP is overall the best performing algorithm on this dataset, winning on 10 out of the 13 benchmarks tested and, in some cases, outperforming its competitors by almost two orders of magnitude ($e.g.$, *blockmap*). For example, on the *blockmap_05_02* benchmark, BE solved 2 out of 10 instances (in 902.13 sec), Smile solved 3 out of 10 instances (in 5111.03 sec), while AO-ID+SAT and AO-ID+BCP solved all 10 instances in 20.93 and 2.30 seconds, respectively. Notice again that BE and Smile are competitive with AO-ID+BCP only on the smallest problem instances ($e.g.$, *pathfinder*). The relatively worse performance of AO-ID+SAT/AO-ID+BCP can be explained by the computational overhead of constraint propagation which did not pay off in this case.

6 Conclusion

In this paper we extended the AND/OR search space for graphical models to influence diagrams and presented a depth-first AND/OR graph search algorithm for computing the optimal policy of an influence diagram. We also augmented the algorithm with constraint propagation in order to exploit the determinism that may be present in the diagram. The efficiency of our approach was demonstrated empirically on various benchmarks for influence diagrams containing a significant amount of determinism. Future work includes the extension of the AND/OR search algorithm into a Branch-and-Bound scheme in order to use domain-specific heuristic information. We also plan to apply the AND/OR search approach to Limited Memory Influence Diagrams (LIMIDs) [Lauritzen and Nilsson, 2001] as well as to the more general Plausibility-Feasibility-Utility (PFU) framework [Pralet *et al.*, 2007]. Finally, we can incorporate an adaptive caching mechanism as suggested in [Mateescu and Dechter, 2005].

Related work: Our approach is closely related to the decision graph search algorithms from [Qi and Poole, 1995]. Unlike Qi and Poole's method, our approach requires the influence diagram to respect the regularity constraint only, does not require additional information wrt the decision alternatives ($i.e.$, framing functions), does not impose any restrictions on the utility functions and the arc weights of the AND/OR graph do not involve complex computation ($i.e.$, Bayesian inference) as they are derived solely from the ID's input functions via simple arithmetic computations.

References

[Cooper, 1988] G.F. Cooper. A method for using belief networks as influence diagrams. In *UAI*, pages 55–63, 1988.

[Dechter and Mateescu, 2007] R. Dechter and R. Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106, 2007.

[Dechter, 2000] R. Dechter. A new perspective on algorithms for optimizing policies under uncertainty. In *AIPS*, pages 72–81, 2000.

[Howard and Matheson, 1984] R. A. Howard and J. E. Matheson. *Influence diagrams. The principles and applications of Decision analyis*. Menlo Park, CA, USA, 1984.

[Jensen *et al.*, 1994] F. Jensen, F.V. Jensen, and S.L. Dittmer. From influence diagrams to junction trees. In *UAI*, 1994.

[Lauritzen and Nilsson, 2001] S. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Mngmnt. Science*, 47(9):12351251, 2001.

[Mateescu and Dechter, 2005] R. Mateescu and R. Dechter. AND/OR cutset conditioning. In *IJCAI*, 2005.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[Pralet *et al.*, 2007] C. Pralet, G. Verfaillie, and T. Schiex. An algebraic graphical model for decision with uncertainties, feasibilities, and utilities. *JAIR*, 29(1):421–489, 2007.

[Qi and Poole, 1995] R. Qi and D. Poole. A new method for influence diagram evaluation. *Comp. Intell.*, 11(1), 1995.

[Shenoy, 1992] P. Shenoy. Valuation-based systems for Bayesian decision analysis. *OR*, 40(3):463–484, 1992.

[Tatman and Shachter, 1990] J.A. Tatman and R.D. Shachter. Dynamic programming and influence diagrams. *IEEE Systems, Man, and Cybernetics*, 1990.

[Walsh, 2000] T. Walsh. SAT vs CSP. In *CP*, 2000.

Using Fusion to Fill in the Gaps in Old Scientific Discoveries' Notebooks

Bassel HABIB
LIP6, University of Paris 6
Paris, France
bassel.habib@lip6.fr

Claire LAUDY
THALES Research & Technology
Palaiseau, France
claire.laudy@thalesgroup.com

Jean-Gabriel GANASCIA
LIP6, University of Paris 6
Paris, France
jean-gabriel.ganascia@lip6.fr

Abstract

We are interested in using a fusion process to complete information prior to the reasoning process about scientific discoveries. In particular, using fusion to complete the set of experiments used as the source of information of the process that led Claude Bernard to his discovery about the effects of curare. Our reconstruction of the discovery process is based on his experiments as they are illustrated in his notebooks. Our main problem is the lack of some important information in his notebooks containing descriptions of his set of experiments. In order to fill in the gaps in his set of experiments, we propose to use fusion between experiments. Prior to fusion, we must ensure that the experiments are compatible according to some similarity measures and depending on the objectives of the fusion. The paper presents our domain-independent approach for similarity checking and fusion, including similarity and fusion strategies.

1 Introduction

In previous papers, we studied the process of scientific discovery [Ganascia and Habib, 2007] and [Habib and Ganascia, 2008]. The aim of our study is to construct computer programs that simulate, at a grosser or finer level of approximation, the path that have been followed by Claude Bernard on his road to important discoveries including his discovery on the effects of curare. Many works from Cognitive Science and AI focus on modeling scientific reasoning. For instance, the work on DENDRAL and Meta-DENDRAL [Buchanan *et al.*, 1969], on AM [Davis and Lenat, 1982], on MOLGEN [Stefik, 1981], on BACON and related programs [Langley *et al.*, 1987] and on KEKADA [Kulkarni and Simon, 1988].

The focus of our research is to study discoveries that occur in experimental sciences. Since the research leading to such discoveries sometimes spans months or years, it is not practical to gather continuous protocols of the process. Thus, we must seek other sources for insights into the processes: for example, scientists' recollections, published papers on the discovery, and accounts from diaries and laboratory notes.

Our reconstruction of the process that led Claude Bernard to the discovery of the effects of curare is based on his notebooks. In most experimental sciences it is customary for scientists to record the details of their experimental activity on a daily basis in a laboratory notebook or log. That is why logs may contain reasons for carrying out an experiment, observations, hypotheses and conclusions drawn from the data.

The abduction [Harman, 1965] plays a crucial role in Claude Bernard's investigations by keeping or changing his initial hypotheses according to the consequences observed through his empirical experiments. We aim at the abductive reasoning about his scientific approach [Josephson and Josephson, 1996] by constructing a causal network that links observations obtained after an experiment with inferred hypotheses. Therefore, descriptions about experiments must be complete and more precise. The main problem, with using notebooks as the source of insight into the discovery process, is the gaps in these notebooks. In the case of Bernard's notebooks, gaps are due to the lack of information about hypotheses inferred from observations. Generally, gaps may be filled in by other sources such as: retrospective recollections of the discoverer during his lifetime or even by his published papers. But in the case of Claude Bernard and as we are interested in detailed experiments as they are illustrated in his notebooks, other sources are not of great use.

That is the reason we propose another way to fill in these gaps by fusing experiments' descriptions. To do so, we use a generic domain-independent approach that we presented in [Laudy and Ganascia, 2008]. The aim is to take two partial experiments and build from them a more precise and more complete experiment. But before being able to fuse two experiments, we have to make sure that they are compatible according to some similarity measures and regarding some precise objectives. If so, the result of their fusion should complete one of the experiment's description with information provided by the other one. Therefore, we will be able to complete the set of Bernard's experiments prior to our reasoning.

The paper is organized as follows. In Section 2, we provide an overview of formal representation of Claude Bernard's experiments. In Section 3, we explain how we process the similarity of two experiments and in Section 4, we emphasize on the fusion aspects. Section 5 describes our results showing the fusion of two selected experiments. Finally, the conclusion summarizes our approach and describes our future directions.

2 Knowledge Representation

2.1 Epistemological studies on Claude Bernard's manuscripts

As previously introduced, the focus of our work is on Claude Bernard's discovery about the effects of curare. Our work is based on data gathered from his notebooks and manuscripts between 1845 and 1875. Since Claude Bernard's manuscripts contain descriptions of experiments in natural language, it was necessary to abstract from these descriptions a number of attributes (experimental criteria), which are rich enough to reflect the complexity of the original descriptions, and sufficiently representative of their variability. An attribute is created if this potential attribute intervenes in a significant proposition of available experiments.

Claude Bernard's manuscripts have been, in a previous study, the subject of an epistemological study, which consists of several steps:

- The transcription of these manuscripts using a text editor. These manuscripts contain experiments using curare or strychnine as a toxic substance;
- The sorting of this work in a chronological order;
- The formalization of an table in which Claude Bernard's experiments are annotated according to several experimental criteria (attributes) such as : weight, age, dose, animal, preparation/manipulation, point of insertion, date, ideas of experiments, observations, hypotheses and references.

2.2 Lack of important information in descriptions of experiments

The identification of the main attributes allowed us to formalize Bernard's experiments. This is a preliminary step to the simulation of these experiments in a virtual laboratory previously built [Habib and Ganascia, 2008]. Prior to the simulation, Claude Bernard's experiments are classified into several sets of experiments. The classification of experiments is done according to one precise criterion; for instance, the set of experiments using dogs as experimental animals, or even the set of experiments including some nerve manipulations, etc. This classification is a methodological problem, because it constitutes an important step in the process of empirical discovery that concerns us, but it is not systematic, and even less, automatic.

Since Claude Bernard does not write down all the details about preparation, observations or even less about the inferred hypotheses, some experiments are not complete comparing to others in the same set of experiments. Hence comes the idea to complete experiments' descriptions by fusing them with descriptions about other experiments from the same set, which are compatible.

Fusion allows us, on the one hand, to reduce the number of experiments within a set of experiments and thus, to reduce the number of possible simulations in a particular set of experiments since each experiment may be the object of a simulation. On the other hand, fusion allows to complete descriptions about some experiments with information of a great interest in our reasoning process. After the fusion step, information includes not only the complete set of observations resulting from experiment but also the hypotheses inferred by Claude Bernard.

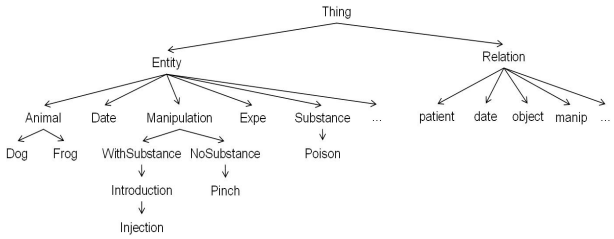


Figure 1: Type hierarchy for Bernard's experiments

2.3 Using conceptual graphs to represent experiments

The Conceptual Graphs model was developed by JF Sowa in [Sowa, 1984]. The model encompasses a formalism for knowledge representation and integrates linguistical, psychological and philosophical aspects. It was conceived in order to develop a logical system, able to represent natural language in a simple manner and allowing deductions and inferences.

The conceptual graphs model is essentially composed of an ontology (called support) and the graphs themselves. The ontology defines the different types of concepts and relations which are used in the conceptual graphs. To describe Claude Bernard's experiments using conceptual graphs, we first had to define the support on which the description will be based. Therefore, we used the ontology that Claude Bernard himself defined during his work (see [Habib and Ganascia, 2008] for more details). The support defines a set of type labels as well as a partial order over the type labels and the support defines the lattice of the conceptual types. The conceptual types are, for instance, Experiment, Poison, Muscle, etc. Figure 1 depicts a subset of the support that we formalized in order to represent Claude Bernard's experiments.

Figure 2 depicts an example of an experiment description, stored as a conceptual graph. The concepts are reprensented in boxes whereas the relationships that exist between the different concepts and objects of the real world are represented by the nodes in ovals. For instance, an *Experiment* has an *Animal* (here the *Dog* named *dog1*), as *patient*.

3 Discrimination between the Experiments

Before to fuse two experiments, one has to determine whether they are compatible or not. The compatibility of two experiments depends on the objectives that we have when we want to fuse them. For instance, sometimes Claude Bernard does not write down all the observations of an experiment because some of them were already observed during an earlier experiment. In such cases, we aim at fusing experiments that have almost similar preparation phases in order to complete the observations. Then the similarity of experiments will be processed regarding the preparation phase. For instance, we will emphasize on similar animals and similar poison. On the contrary, if our aim was to aggregate all the different effects of

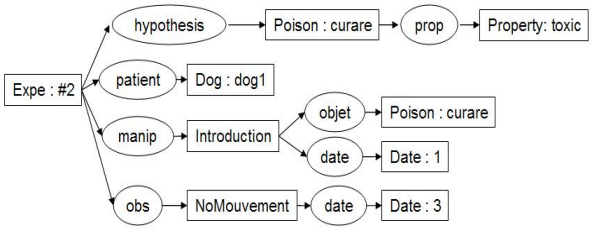


Figure 2: Example of a first experiment

curare, we would process similarity only regarding the poison with no restriction on the animal, point of insertion, observations, etc.

Regarding the whole process of grouping similar experiments and fusing them, the discrimination phase can be viewed as a classification. It also reminds of the issue of conceptual clustering. Conceptual clustering was defined by Michalki and Stepp in [Michalski and Stepp, 1983]. Given a set of objects associated with their descriptions, conceptual clustering allows to define a set of classes that group these objects together.

To determine whether two experiments are sufficiently similar to be fused, we use a similarity measure. Several similarity measures between conceptual graphs have been proposed. Some, as [Sorlin *et al.*, 2003] rely on the similarity of the structure and values of the two graphs taken as a whole. Other works (e.g. [Gandon *et al.*, 2008]) concern the semantic distance between conceptual types. Studies were also performed regarding the issues of classifying a set of conceptual graphs ([de Chalendar *et al.*, 2000] for instance). This is also a way to find the conceptual graphs which are, more or less, similar.

Our approach relies on combining these different approaches. Relying on [de Chalendar *et al.*, 2000] and [Gandon *et al.*, 2008], we define a similarity measure of two concept nodes that depends on the distance between their conceptual types, the distance between their individual values and the similarity of their immediate neighborhood. The similarity between two graphs is then computed, regarding the best matching of their nodes, as [Sorlin *et al.*, 2003] does it.

One of our goals is to compare two experiments using only local comparisons that are "weak" in terms of processing time. Therefore, we propose to compare the different pairs of concepts of the two graphs. The global graph structure of the experiments' descriptions will be handle during the fusion process.

In the next sections, we use the following notations:

- C denotes the set of concepts with conceptual types defined on a support S;
- $c_1 \in C$ and $c_2 \in C$ are two concept nodes;
- $c_1 = [T_1 : v_1]$ and $c_2 = [T_2 : v_2]$ with $T_1 \in S, T_2 \in S$, the two conceptual types of c_1 and c_2.

3.1 Similarity between two concepts

To measure the similarity between two concepts, we propose to compare their conceptual types, their individual markers as well as their neighborhood. The study of the neighborhood gives clue about the context in which a concept is used.

The similarity measure $sim : E \times E \rightarrow [0, 1]$ is expressed as follows:

$$sim(c_1, c_2) = p_1(T_1, T_2) sim_{Type}(T_1, T_2) * \left(\frac{p_2(T_1,T_2) sim_{Ref}(v_1,v_2) + p_3 sim_{rel}(c_1,c_2) - p_4 diss_{Rel}(c_1,c_2)}{p_2+p_3} \right)$$

- p_1, p_2, p_3 and p_4 are weights that allow to give more importance to some elements with regard to the others;
- sim_{Type}, sim_{Ref}, $sim_{RelComm}$ and $diss_{Rel}$ are local similarity/dissimilarity measures that we detail hereafter.

Similarity between conceptual types: $\mathbf{sim_{Type}}$

The similarity measure, between two conceptual types, depends on their distance in the lattice of concepts. Among the different studies that exist, concerning this problem, we are particularly interested in the distance between types proposed by [Gandon *et al.*, 2008].

Our objective is to fuse the different experiments in order to make them more precise. Therefore, unlike most of the existing measures that use the nearest common parent of the two types to be compared as key feature, we will use the nearest common subtype as key type in our measure.

The distance between two types is defined as follows:

$\forall (t_1, t_2) \in S \times S$

$$dist(t_1, t_2) = min_{\{t \leq t_1, t \leq t_2\}} \Big(l_S(t_1, t) + l_S(t_2, t) \Big)$$

with $\forall (t, t') \in S \times S, t \leq t'$

$$l_S(t, t') = \sum_{t_i \in \langle t, t' \rangle, t_i \neq t} \left[\frac{1}{2^{prof(t_i)}} \right]$$

where $\langle t, t' \rangle$ is the shortest path between t and t' and $prof(t)$ is the depth of t in the support.

Given this distance, the similarity between t_1 and t_2 is given by $1 - dist(t_1, t_2)$.

Similarity between two referents: $\mathbf{sim_{Ref}}$

The similarity between the values of two concepts depends on the conceptual types of the concepts and the application domain.

A lot of distances exist on different types of data and are specific to different application domains. The similarity measure between two referents can be based on any of them.

Similarity regarding neighborhoods: $\mathbf{sim_{Rel}}$

In order to compare the context in which the two concepts are expressed, we propose to compare their immediate neighborhood. Intuitively, the similarity measure of two concepts given the common neighboring relations is processed by measuring the proportion of relations linked to the concepts and that have the same conceptual type:

$$sim_{RelComm}(c_1, c_2) = \frac{2 * nbRelComm(c_1, c_2)}{nbRel(c_1) + nbRel(c_2)}$$

with $nbRelComm(c_1, c_2)$: the number of relations shared by c_1 and c_2, regarding their conceptual types;
and $nbRel(c)$: the total number of relations linked to the concept c.

Dissimilarity regarding neighborhoods: $\mathbf{diss_{Rel}}$
As for common neighboring relations, we compare the neighboring relations of two concepts that are different:

$$diss_{Rel}(c_1, c_2) = \frac{nbRelDiff(c_1, c_2)}{nbRel(c_1) + nbRel(c_2)}$$

$nbRelDiff(c_1, c_2)$: the number of relations that are not shared between c_1 and c_2;
$nbRel(c)$: the total number of relations linked to the concept c.

3.2 Similarity between two experiments

The similarity of two experiments depends on the different matchings that exist between the concepts of the two graphs. It is processed given the similarity of the matching concepts.

Given a matching of the concepts of the two graphs G_1 and G_2, the similarity between G_1 and G_2 is computed as follows:

$$sim_{match}(G_1, G_2) = \frac{\sum_{(c_1,c_2)\in app} sim_{concept}(c_1, c_2)}{min(|C_1|, |C_2|)}$$

- C_1 (resp. C_2) is the set of concepts of G_1 (resp. G_2) and $|C_1|$ (resp. $|C_2|$) is the number of concepts in the graph G_1 (resp. G_2);
- $c_1 \in C_1$ (resp. $c_2 \in C_2$) is a concept of G_1 (resp. G_2).

The global similarity measure between two graphs G_1 and G_2 is then computed by maximizing the similarity of the different possible matchings:

$$sim(G_1, G_2) = \max_{\forall match \subseteq V_1, V_2} sim_{match}(G_1, G_2)$$

4 Fusion of Claude Bernard's Experiments

As a second step of the conceptual clustering, the naming allows to define a description of each class. The fusion phase can be viewed as a particular type of naming where the fused experiment description describes the cluster of experiments.

4.1 Maximal join as fusion operator

In [Laudy *et al.*, 2007], we presented a framework for high-level information fusion based on the use of the conceptual graphs formalism. Our approach is generic. In [Laudy and Ganascia, 2008], We used conceptual graphs to represent TV program descriptions. The descriptions were coming from different sources of information and were related to the same TV program. In this work, we represent Claude Bernard's experiment descriptions. The descriptions that we want to fuse relate to quite similar but different experiments. Furthermore, they are all coming from the same source of information: Claude Bernard's notebooks.

The fusion process relies on the conceptual graphs model. We use the maximal join operation defined by Sowa in order to fuse information. The maximal join operation allows to fuse two compatible sub graphs of two conceptual graphs.

The maximal join operation copies all the information that is present in the initial graphs in the new one. Intuitively, when one wants to join two graphs maximally, the first step is to look for two compatible sub graphs in the two initial ones. The initial graphs are then joined, according to the compatible sub graphs. Furthermore, two concepts are compatible if:

- their conceptual types share a common sub-type different from $\perp$;
- their referents conform their most general subtype; and
- either one of their referent is undefined, or their referents are identical.

The maximal join keeps the most specific elements of two compatible sub graphs and complete one graph according to the information contained in the other one. Furthermore, it gives several results, which depict the different ways of combining the information, that is to say, the different fusion hypotheses. An example is given in section 5.

4.2 Using fusion strategies to handle noisy data

The maximal join is a fusion operator. However, as stated in [Laudy and Ganascia, 2008], using only the maximal join is not sufficient in order to fuse information coming from real systems. Real data is noisy and knowledge about the domain is often needed in order to fuse two different but compatible values into a single one. Observations such as a person named "J. Smith" and a one named "Mr. John Smith" are not equals, but our background knowledge let us believe that the two observations rely to the same person.

It is necessary to extend the notion of compatibility between different concepts in the maximal join operation by introducing domain knowledge. The notion of compatibility between concepts is extended from compatibility of conceptual types only to compatibility of individual markers as well. We introduced the notion of fusion strategies. They are rules encoding domain knowledge and fusion heuristics. The definition of the fusion strategies is divided into two parts: the definition of the compatibility conditions between two concepts and the process of the fused value of two concepts.

Let $\mathcal{S}$ be a lattice of conceptual types and l be a set of individual markers. E is the set of concept nodes defined on $\mathcal{S} \times l$, G_1 and G_2 are two conceptual graphs defined on $\mathcal{S}$ and c_1 and c_2 are concepts defined on E such that $c_1 \in G_1$ and $c_2 \in G_2$. A fusion strategy $strategy_{fusion}$ is defined as follows:

$$strategy_{fusion} = \begin{array}{ll} if & f_{comp}(c_1, c_2) \\ then & f_{fusion}(c_1, c_2) \\ else & \{c_1, c_2\} \end{array}$$

where $f_{comp} : E \times E \rightarrow \{true, false\}$ is a function testing the compatibility of two concept nodes,
and $f_{fusion} : E \times E \rightarrow E$ is a fusion function upon the concepts nodes of the graphs.

The fusion strategies applied on two concept nodes result either in a fused concept node if the initial nodes are compatible, or in the initial nodes themselves if they are incompatible.

5 Case Study

5.1 Context

Within the context of understanding the process of scientific discovery followed by Claude Bernard, the goal here, as said

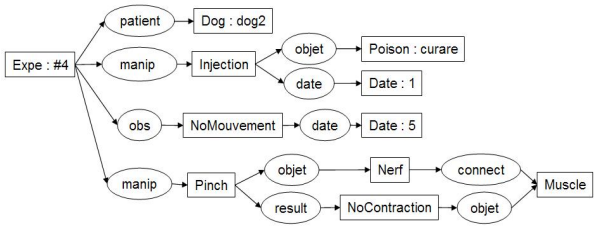


Figure 3: Example of a second experiment

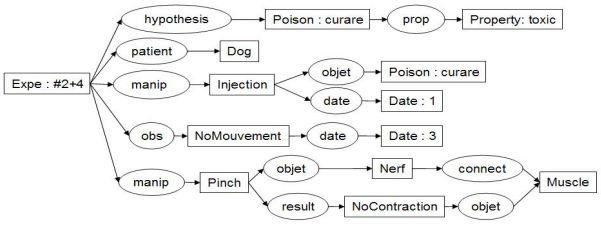


Figure 4: Fusion of two experiment's descriptions

before, is to reduce the number of experiments and, therefore, to reduce the number of simulations of his experiments.

Therefore, our first step consisted of the classification of the experiments into several classes (sets of experiments).

We will take here the example of the class of experiments that Claude Bernard achieved on dogs, using curare as the toxic substance. This set of experiments includes ten experiments. The selection of this set of experiments on dogs, using curare, can be processed semi-automatically using our similarity measure combined with a set of weights that allows to have high scores of similarity only in case two experiments share *Dog* as the same sub-type of *Animal* and the same *Poison* name. After that, one has only to check the poison used in each class of experiments.

Once the selection of the set of experiments is done, we use our similarity measures and process the similarity between the experiments. In our example, the similarity measures include the given animal (a dog), manipulations (introduction of a toxic substance), the poison used (the curare) and the observations (whether the curare affects the animal or not). Including the observations in the similarity measure allows to ensure that we will aggregate the hypotheses that relate to experiments that have the same conclusions or observations. As a result of the similarity step, we could devide our set of experiments into two subsets, the first one *set1* (see Table 1) contains the experiments for which the curare affects the experimental animal (six experiments including experiments 2 and 4 for which we will show the result of the fusion). The second subset *set2* (see Table 1) is the one for which the curare has no effect (four experiments).

The last step is to apply fusion strategies on the experiment descriptions within the same subset of experiments. The fusion will allow, as said before, to aggregate all the hypotheses given by Claude Bernard, regarding a same subset of experiments. Therefore, after the fusion of the experiments, only one simulation will be sufficient to validate or invalidate several hypotheses.

5.2 From natural language descriptions to conceptual graphs

As said before, the preliminary step of the process when we want to simulate Bernard's reasoning is to formalize his experiments. This is part of the work realized by the epistemological study and that we completed in order to transform the table into a set of conceptual graphs. In the following sections, we illustrate our approach on a concrete example that uses two of Bernard's experiments on curare as follows:

"Experiment 2 : Small dog, 12 days-old. Arrow of curare lodged in the tissue of the thigh. 3 minutes later, death without screams or convulsions. Immediately after death, reflex movements are abolished. The heart beats a few more moments. At the autopsy, nothing can explain the death."

"Experiment 4 : Small dog, 12 days-old. Dissolution of five centigrams of curare in water and injected by the anus in the stomach. 5 minutes later, death with the same symptoms as in the experiment 2. Immediately after death, we can not produce any reflex movement. The nerves in the legs being naked, cut or pinched do not give any contraction in the muscles. At the autopsy, nothing can explain the death."

The corresponding conceptual graphs are shown in Figures 2 and 3. For a matter of readability the figures only depict a subset of the experiment descriptions.

5.3 Similarity between experiments

The aim is to complete the observations and/or the details of the preparation of the animal for the experiments. Therefore, we will emphasize on the similarity of the preparation phase, and particularly on the type of animal, the poison used and the manipulation that is done. We use a set of weights for p_1, p_2, p_3 and p_4 found experimentally by analysing a subset of the experiment descrptions. Intuitively, experiments that share the same *Observation* are very close and add up a high score of similarity. Experiments that shre the same *Animal*, *Manipulation*, *Poison* ... add up a medium score of similarity. Furthermore, as we aim at completing the experiment descriptions, we decided not to take into account the neighbourhood of each concept.

The similarity between 2 concepts referents is processed aaccording to the Leventshtein edit distance [Levenshtein, 1966] if the referents are strings and according to a normalized distance between numerical values otherwise.

We compared the experiments belonging to the different sets. Table 1 shows the results of the similarity rate between the experiment 2 and the other experiments from the same set of experiments using dogs.

5.4 Completing descriptions of experiments

Figures 2 and 3 show the description of two experiments that are very similar regarding our similarity measure. We used our fusion platform to fuse them. Figure 4 shows the result of the fusion process. The experiment descriptions have been completed regarding the observations that were made during the second experiment.

On the one hand, our process allows to fuse experiments regarding the observations. Thanks to that, the implicit observations that Claude Bernard didn't rewrite from one experiment to an other one are used in the simulation. On the other hand, we can also fuse the experiment descriptions regarding the hypothesis. Then, one simulation will be sufficient to validate or invalidate several hypothesis. As the simulation phase is very time consuming, it saves a considerable amount of time within the global study of scientific discovery.

Table 1: The similarity rate of the experiment *exp2*.

set1	exp2	exp4	exp7	exp9	exp20	exp21
exp2	1	0.82	0.83	0.91	0.97	0.895
set2	exp5	exp6	exp8	exp22		
exp2	0.62	0.62	0.65	0.62		

6 Conclusion & Perspectives

In this paper, we showed how we used similarity and fusion strategies to fill in the gaps in Claude Bernard's notebooks. Since he doesn't always write down every detail of preparation, observations or even inferred hypotheses, the fusing of his experiments helps us to complete them, which is of great interest for the reasoning on his discovery process.

However, to simulate the reasoning about Bernard's scientific approach, especially, in his reasoning about the process of the discovery of the effects of curare, we would like to add a generalization feature among the different experiments. From several almost similar experiments, Claude Bernard was able to generalize some aspects such as the properties of curare. For instance, from two experiments where the mode of introduction of curare is different, but where the observations are similar, one would like to automatically deduce that the introduction mode doesn't affect the curare properties.

The studies performed by [de Chalendar *et al.*, 2000] for instance, aim at classifying conceptual graphs. After constructing classes of conceptual graphs, one of the goals is to find a single conceptual graph to represent each set of classified graphs. This graph is more general than each graph in the class it describes. We will rely on these works in order to propose a method for generalizing knowledge from a set of experiments. We aim at mixing generalization and fusion processes. In do so, we will not only deduce general knowledge from a set of different related experiments, but also take advantage of fusion to complete the generalization of several partial experiment descriptions.

References

[Buchanan *et al.*, 1969] B. G. Buchanan, E. A. Sutherland, and E. A. Feigenbaum. Heuristic dendral: A program for generating explanatory processes in organic chemistry. *Machine Intelligence 4*, 1969.

[Davis and Lenat, 1982] R. Davis and D. Lenat. *AM: Discovery in Mathematics as Heuristic Search in Knowledge System in Artificial Intelligence, Part one*. McGraw-Hill, New-York, 1982.

[de Chalendar *et al.*, 2000] G. de Chalendar, B. Grau, and O. Ferret. Conceptual graphs generalization. In *RFIA 2000. 12ème Congrés Francophone AFRIT-AFIA*, Paris, 2000.

[Ganascia and Habib, 2007] J-G. Ganascia and B. Habib. An attempt to rebuild c. bernard's scientific steps. *The Tenth International Conference on Discovery Science*, 2007.

[Gandon *et al.*, 2008] F. Gandon, O. Corby, I. Diop, and M. Lo. Distances sémantiques dans des applications de gestion d'information utilisant le web sémantique. In *Semantic similarity workshop, EGC 2008*, Sophia Antipolis, France, 2008.

[Habib and Ganascia, 2008] B. Habib and J-G. Ganascia. Using ai to reconstruct claude bernard's empirical investigations. In *The 2008 International Conference on Artificial Intelligence (ICAI'08)*, pages 496–501, Las Vegas, USA, July 2008.

[Harman, 1965] G. H. Harman. *The inference to the best explanation*. pages 88–95, 1965. The Philosophical Review.

[Josephson and Josephson, 1996] J. Josephson and S. Josephson. *Abductive inference: computation, philosophy, technology*. 1996. Cambridge University Press.

[Kulkarni and Simon, 1988] D. Kulkarni and H. A. Simon. The processes of scientific discovery: The strategy of experimentation. *Cognitive Science*, 12:139–175, 1988.

[Langley *et al.*, 1987] P. Langley, H. A. Simon, G. L. Bradshaw, and J. M. Zytkow. Scientific discovery: Computational explorations of the creative processes. *The MIT Press*, 1987.

[Laudy and Ganascia, 2008] C. Laudy and J-G. Ganascia. Information fusion in a tv program recommendation system. In *11th International Conference on Information Fusion*, pages 1455–1462, Cologne, Germany, July 2008.

[Laudy *et al.*, 2007] C. Laudy, J-G. Ganascia, and C. Sedogbo. High-level fusion based on conceptual graphs. In *10th International Conference on Information Fusion*, Québec, Canada, July 2007.

[Levenshtein, 1966] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[Michalski and Stepp, 1983] R. S. Michalski and R. E. Stepp. Learning from observation: Conceptual clustering. *Machine Learning: An Artificial Intelligence Approach*, pages 331–363, 1983.

[Sorlin *et al.*, 2003] S. Sorlin, P-A Champin, and C. Solnon. Mesurer la similarité de graphes étiquetés. In *9èmes Journées Nationales sur la résolution pratique de problèmes NP-Complets*, pages 325–339, 2003.

[Sowa, 1984] J. F. Sowa. *Conceptual Structures. Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA, 1984.

[Stefik, 1981] M. Stefik. Planning with constraints (molgen part 1). *Artificial Intelligence*, 16:2:111–140, 1981.