logic, and in particular, LITE may be used for some existing automated theorem prover.

LITE has an implementation in the LINCKS system in the sense that a LINCKS database is a structure in LITE. This strong correlation is a promising start-point for developing and implementing an explicit LITE reasoner.

LITERATURE

[All83] Allen, J.F., Towards a General Theory of Action and Time, *Artificial Intelligence*, 23(2), pp 123-154, 1983.

[Bac88] Bacchus, F., Tenenberg, J., Koomen, J.A., *A Non-Reified Temporal Logic*, 1988.

[Bel77] Bell, J., Machover, M, *A Course in Mathematical Logic*, Elsevier Science Publishers B.V., Netherlands, 1977.

[Bur84] Burgess, J.P., Basic Tense Logic, *Handbook of Philosophical Logic*, vol II, pp 89-133, D. Reidel Publishing Company, 1984.

[For85] Forbus, K. D., The Role of Qualitative Dynamics in Naive Physics, *Formal Theories of the Commomsense World*, pp 185-226, Ablex Publishing Corporation, New Jersey, USA, 1985.

[Geo87] Georgeff, M. P., Actions, Processes and Causality, *proceedings of the 1986 workshop on Reasoning About Actions & Plans*, pp 99-159, Morgan Kaufmann Publishers Inc., California, USA, 1987.

[Gre69] Green, C. C., Applications of theorem proving in problem solving, *Proceedings of the 1st IJCAI*, pp 219-239, 1969.

[Hay85] Hayes, P. J., The Second Naive Physics Manifesto, *Formal Theories of the Commonsense World*, pp 1-36, Ablex Publishing Corporation, New Jersey, USA, 1985.

[McC60] McCarthy, J., Programs with Common Sense, *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, London 1960.

[McC85] McCarthy, J., Programs with Common Sense, *Readings in Knowledge Representation*, pp 300-307, Morgan Kaufmann Publishers Inc., California, USA, 1985.

[McD82] McDermott, D., A Temporal Logic for Reasoning about Processes and Plans, *Cognitive Science*, pp 101-155, 1982.

[Pad86] Padgham, L., A Description of LINCKS: Linköpings Intelligent Knowledge Communication System, *Proceedings of IFIP Working Conference on Methods and Tools for Office Systems*, Pisa, Italy, 1986.

[Pad88] Padgham, L., NODE: a database for use by intelligent systems, *Proceedings of the International Symposium on Methodologies for Intelligent Systems* Torino, Italy, 1988.

[Rei86] Reichgelt, H., *Semantics for Reified Temporal Logic*, D.A.I. research paper No 299, 1986.

[San89] Sandewall, E., A Decision Procedure for A Theory of Actions and Plans, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pp 501-512, 1989.

[Sho87] Shoham, Y, Temporal logics in AI: Semantical and ontological considerations, *Artificial Intelligence*, 33(1), pp 89-104, 1987.

[Sho88] Shoham, Y, *Reasoning about Change*, MIT Press, 1988.

[Win88] Winslett, M., Reasoning About Action Using a Possible Models Approach, *proceddings of the 7th AAAI*, pp 89-93, 1988. 1989.

Valuation-Based Systems for Propositional Logic

Prakash P. Shenoy
School of Business, University of Kansas, Summerfield Hall, Lawrence, KS 66045-2003
Tel: (913)-864-7551, Fax: (913)-864-5328, Bitnet: pshenoy@ukanvm

ABSTRACT

This paper describes a valuation-based system for propositional logic. The solution method of valuation-based systems can efficiently check for inconsistencies in a knowledge-based system. This method is capable of detecting all contradictions that can be detected using the expressive power of propositional logic. The computational complexity of this method depends on the sizes of the valuations and on the graphical structure of the valuation-based system.

Key Words: Valuation-based systems, consistency in rule-based systems, satisfiability problem, theorem proving, knowledge representation, combination, marginalization.

1. INTRODUCTION

A valuation-based system is a knowledge-based system in which knowledge is represented by functions called valuations [Shenoy, 1989]. In valuation-based systems, inferences are made using two operations called combination and marginalization. The process of making inferences can be simply described as finding the marginal of the joint valuation for each variable in the system.

This paper presents a new efficient computational technique for checking for inconsistencies in valuation-based systems. (We use the word inconsistency in the logical sense: a set of propositions is inconsistent if contradiction is entailed). This technique is capable of detecting all contradictions that can be detected using the expressive power of propositional logic. The computational complexity of this technique depends on the sizes of the valuations and on the graphical structure of the valuation-based system.

The problem of consistency is NP-complete in the worst case. Hence there are no guarantees that the method will always work. However, this does not mean that real-world valuation-based systems always conform to the worst case. In such cases, it is desirable to have an efficient method for detecting inconsistencies. If each valuation relates only a small number of variables and the subsets of variables thus related can be arranged in a Markov tree without adding large subsets, then consistency can be checked using only local computation. The computational complexity of this local computational scheme is $O((n+k)s)$ where n is number of variables, k is the number of valuations, and s is the cardinality of the frame of the largest subset of variables in the Markov tree.

The issue of consistency in valuation-based systems is an important one. Most commercial rule-based languages do not check whether the rule-base is consistent or not. For example, in Texas Instruments' PERSONAL CONSULTANT system, if two rules such as If X=x then Y=y, and If X=x then Y=~y are entered with variable Y as the goal, the system will first ask the user for the value of X. If the user responds by stating that X=x, the system then responds by concluding either Y=y or Y=~y depending on the order in which the rules were entered in the rule-base!

The method we propose is distinct from previous approaches to this problem. Suwa *et al.* [1982] and Nguyen *et al.* [1985] restrict their analyses to pairs of rules with contradictory conclusions. As Ginsberg [1988] has pointed out, such pairwise comparisons cannot guarantee detection of all potential inconsistencies. Ginsberg's [1988] method called 'knowledge-base reduction' is based on techniques developed by de Kleer [1986] for assumption-based truth maintenance systems. His method flags all potential inconsistencies. The task we address is a bit simpler. We do not attempt to detect potential inconsistencies. The method we present detects actual inconsistencies. Thus, two rules such as If X=x then Y=y, and If X=x then Y=~y are not inconsistent until X=x is posited. The method described here is derived from the valuation-based system for discrete optimization described in Shenoy [1990b].

If a knowledge-based system is inconsistent, the solution method of valuation-based system also is capable of isolating a minimal set of inconsistent valuations. Furthermore, if a knowledge-based system is consistent, the solution method of valuation-based systems also is capable of generating a proof, i.e., an assignment of truth values to each variable that is consistent with the knowledge in the knowledge-base. Page limitations prevents us from describing these capabilities here. See Shenoy [1990] for a demonstration of these claims.

© 1990 by Elsevier Science Publishing Co., Inc.
Methodologies for Intelligent Systems, 5
Z.W. Ras, M. Zemankova, and M.L. Emrich, Editors

An outline of this paper is as follows. In Section 2, we define valuations and proper valuations which are used to represent knowledge. In Section 3, we define two operations on valuations called combination and marginalization. These operations are used to make inferences from the knowledge-base. In Section 4, we give a formal definition of a consistent valuation-based system. In Section 5, we describe a method for checking consistency of a valuation-based system and describe its computational complexity. In section 6 we make some concluding remarks.

2. KNOWLEDGE REPRESENTATION

We will represent knowledge by functions, called valuations, from the space of configurations to the space of values.

Consider a variable X. We use the symbol $\mathcal{W}_X$ for the set of possible values of X. We assume that one and only one of the elements of $\mathcal{W}_X$ can be the true value of X. We call $\mathcal{W}_X$ the *frame for X*.

Let $\mathcal{X}$ denote the set of all variables. In this paper we will be concerned only with the case where $\mathcal{X}$ is finite. We will also assume that all the variables in $\mathcal{X}$ have finite frames.

We will often deal with non-empty subsets of variables in $\mathcal{X}$. Given a non-empty subset h of $\mathcal{X}$, let $\mathcal{W}_h$ denote the Cartesian product of $\mathcal{W}_X$ for X in h, i.e., $\mathcal{W}_h = \times\{\mathcal{W}_X \mid X \in h\}$. We can think of the set $\mathcal{W}_h$ as the set of possible values of the joint variable h. Accordingly, we call $\mathcal{W}_h$ the *frame for h*. Also, we will refer to elements of $\mathcal{W}_h$ as *configurations of h*. We will use this terminology even when h consists of a single variable, say X. Thus we will refer to elements of $\mathcal{W}_X$ as *configurations of X*. We will use lower-case, bold-faced letters such as **x**, **y**, etc. to denote configurations. Also, if **x** is a configuration of g and **y** is a configuration of h and $g \cap h = \varnothing$, then (**x**,**y**) will denote a configuration of $g \cup h$.

In (categorical) rule-based systems, knowledge is represented by rules. For example, we may have a rule relating two variables X and Y: If X=x then Y=y. In our framework, we will represent knowledge using functions called valuations.

Suppose $h \subseteq \mathcal{X}$. A *valuation for h* is a function $H: \mathcal{W}_h \rightarrow \{1, 0\}$. We will refer to the elements of the set $\{1, 0\}$ as *values*. The value 1 represents truth and the value 0 represents falsehood. Thus a valuation for h is a function from the set of configurations of h to the set of truth values. For example, the rule If X=x then Y=y that relates two variables X and Y whose frames are, respectively, $\mathcal{W}_X = \{x, \sim x\}$ and $\mathcal{W}_Y = \{y, \sim y\}$, can be represented by the valuation V for {X, Y} defined as follows: $V(x,y) = 1$, $V(x,\sim y) = 0$, $V(\sim x, y) = 1$, $V(\sim x,\sim y) = 1$. Note that this is basically the truth-table representation of the rule interpreted as a conditional.

In order to represent the notion of consistent knowledge, we will define proper valuations. Suppose $h \subseteq \mathcal{X}$. A valuation H for h is said to be *proper* if there exists a configuration **x** of h such that H(**x**) = 1, and *improper* otherwise. Thus a proper valuation cannot be identically equal to 0 for all configurations.

The motivation behind the above definition is clear. Earlier, we had defined a frame as a set of configurations exactly one of which is true. Thus it should not be possible for a consistent piece of knowledge to rule out all configurations.

3. COMBINATION AND MARGINALIZATION

In this section, we define two basic operations for valuations called combination and marginalization. Combination corresponds to logical conjunction and represents aggregation of knowledge. Marginalization is less familiar and represents crystallization of knowledge.

Before we define the two operations, we need to introduce some notation. *Projection* of configurations simply means dropping extra coordinates; if (w,x,y,z) is a configuration of {W,X,Y,Z}, for example, then the projection of (w,x,y,z) to {W,X} is simply (w,x), which is a configuration of {W,X}. Formally, if g and h are sets of variables, $h \subseteq g$, and **x** is a configuration of g, then we will let $\mathbf{x}^{\downarrow h}$ denote the projection of **x** to h. The projection $\mathbf{x}^{\downarrow h}$ is always a configuration of h.

Combination. Suppose G and H are valuations for g and h, respectively. The valuation $G \otimes H$ for $g \cup h$ is defined as follows:

$$(G \otimes H)(\mathbf{x}) = G(\mathbf{x}^{\downarrow g})\, H(\mathbf{x}^{\downarrow h})$$

for all $\mathbf{x} \in \mathcal{W}_{g \cup h}$. If $G \otimes H$ is not a proper valuation then we shall say that G and H are *not combinable*. If $G \otimes H$ is a proper valuation, then we shall say that G and H are *combinable* and that $G \otimes H$ is the *combination of G and H*.

Note that if either G or H is improper, then $G \otimes H$ is improper. However, if both G and H are proper, then $G \otimes H$ may be proper or improper. Whether $G \otimes H$ is proper or not depends on if the knowledge represented by G and H are consistent or not when aggregated. Intuitively, combination corresponds to aggregation of knowledge.

Note that the combination operator is commutative and associative; $G \otimes H = H \otimes G$, and $(G \otimes H) \otimes K = G \otimes (H \otimes K)$. Thus, when we combine several valuations $G_1, \ldots, G_k$, we can write $G_1 \otimes \ldots \otimes G_k$ or simply $\otimes\{G_i \mid i = 1, \ldots, k\}$ without indicating the order in which the combination is carried out.

Consider two variables X and Y whose frames are, respectively, $\mathcal{W}_X = \{x, \sim x\}$ and $\mathcal{W}_Y = \{y, \sim y\}$. Let G_1, G_2, and G_3 be three valuations for {X}, {X,Y}, and {X,Y} respectively, given as follows:

$$G_1(x) = 1,\ G_1(\sim x) = 0;$$
$$G_2(x,y) = 1,\ G_2(x,\sim y) = 0,\ G_2(\sim x,y) = 1, G_2(\sim x,\sim y) = 1;$$
$$G_3(x,y) = 0,\ G_3(x,\sim y) = 1,\ G_3(\sim x,y) = 1, G_3(\sim x,\sim y) = 1.$$

G_1 represents the proposition X=x; G_2 represents the conditional: If X=x then Y=y; and G_3 represents the conditional: If X=x then Y=~y. Table 1 shows that $G_1 \otimes G_2 \otimes G_3$ is an improper valuation. Therefore the set of valuations $\{G_1, G_2, G_3\}$ is inconsistent.

Table 1. The combination of G_1, G_2 and G_3.

w	$G_1(\mathbf{w}^{\downarrow\{X\}})$	$G_2(\mathbf{w})$	$G_3(\mathbf{w})$	$(G_1 \otimes G_2 \otimes G_3)(\mathbf{w})$
(x,y)	1	1	0	$G_1(x)G_2(x,y)G_3(x,y) = 0$
(x,~y)	1	0	1	$G_1(x)G_2(x,\sim y)G_3(x,\sim y) = 0$
(~x,y)	0	1	1	$G_1(\sim x)G_2(\sim x,y)G_3(\sim x,y) = 0$
(~x,~y)	0	1	1	$G_1(\sim x)G_2(\sim x,\sim y)G_3(\sim x,\sim y) = 0$

Marginalization. Suppose G is a valuation for g and suppose $h \subseteq g$. Then the *marginal of G for h*, denoted by $G^{\downarrow h}$, is the valuation for h defined as follows:

$$G^{\downarrow h}(\mathbf{x}) = \mathrm{MAX}\{G(\mathbf{x},\mathbf{y}) \mid \mathbf{y} \in \mathcal{W}_{g-h}\}$$

for all $\mathbf{x} \in \mathcal{W}_h$.

It follows from the above definition that $G^{\downarrow h}$ is proper if and only if G is proper. Intuitively, marginalization corresponds to crystallization of knowledge. If G is a valuation for g representing some knowledge about variables in g, and $h \subseteq g$, then $G^{\downarrow h}$ represents knowledge about variables in h implied by G if we disregard variables in g–h.

Note that if $k \subseteq h \subseteq g$, and G is a valuation on G, then $(G^{\downarrow h})^{\downarrow k} = G^{\downarrow k}$. In words, the order of deletion of variables in the marginalization operation does not matter.

Consider the valuations G_1 and G_2 as defined above representing proposition X=x and conditional If X=x then Y=y, respectively. If we combine G_1 and G_2 and marginalize the combination for {Y}, the resulting valuation $(G_1 \otimes G_2)^{\downarrow\{Y\}}$ is given by $(G_1 \otimes G_2)^{\downarrow\{Y\}}(y) = 1$ and $(G_1 \otimes G_2)^{\downarrow\{Y\}}(\sim y) = 0$, i.e., $(G_1 \otimes G_2)^{\downarrow\{Y\}}$ represents the proposition Y=y (see Table 2). Thus combination and marginalization to {Y} gives the same result as the *modus ponens* form of inference in propositional logic (X=x, and If X=x then Y=y, ∴ Y=y).

Table 2. Modus ponens represented as combination and marginalization. G_1 represents proposition X=x and G_2 represents conditional If X=x then Y=y. $(G_1 \otimes G_2)^{\downarrow\{Y\}}$ represents the conclusion Y=y.

w	$G_1(w^{\downarrow\{X\}})$	$G_2(w)$	$(G_1 \otimes G_2)(w)$
(x,y)	1	1	1
(x,~y)	1	0	0
(~x,y)	0	1	0
(~x,~y)	0	1	0

w	$(G_1 \otimes G_2)^{\downarrow\{Y\}}(w)$
y	$\mathrm{MAX}\{(G_1 \otimes G_2)(x,y), (G_1 \otimes G_2)(\sim x,y)\} = 1$
~y	$\mathrm{MAX}\{(G_1 \otimes G_2)(x,\sim y), (G_1 \otimes G_2)(\sim x,\sim y)\} = 0$

Similarly, other forms of logical inference such as *modus tollens* (Y=~y, and If X=x then Y=y, ∴ X=~x), *disjunctive syllogism* (X=x or Y=y, and X=~x, ∴ Y=y), and *disjunctive elimination* (X=x or Y=y, If X=x then Z=z, and If Y=y then Z=z, ∴ Z=z) are shown to be special cases of combination and marginalization (for the appropriate variable) in Tables 3, 4 and 5, respectively.

4. CONSISTENT VALUATION-BASED SYSTEMS

In this section, we will formally define what we mean by a consistent valuation-based system.

A *valuation-based* system consists of a finite set of variables $\mathcal{X}$, a finite frame $\mathcal{W}_X$ for each variable X in $\mathcal{X}$, and a finite collection of valuations $\{V_1, ..., V_k\}$ where each valuation V_i is for some subset h of $\mathcal{X}$.

Let $\mathcal{H}$ denote the set of subsets of $\mathcal{X}$ for which valuations exist in the valuation-based system. For simplicity of exposition, we will assume that each variable in $\mathcal{X}$ is included in some element of $\mathcal{H}$, i.e., $\cup\mathcal{H} = \mathcal{X}$ (if not, we can always disregard such variables). Note that there could be more than one valuation defined for a subset of variables. Using the language of graph theory, the set $\mathcal{H}$ is called a *hypergraph on* $\mathcal{X}$ and each of its elements is called a *hyperedge*.

Thus, a valuation-based system (VBS) can be denoted formally by the quadruple $\{\mathcal{X}, \{\mathcal{W}_X\}_{X\in\mathcal{X}}, \{V_1, ..., V_k\}, \mathcal{H}\}$ representing variables, frames, valuations, and hypergraph, respectively.

Suppose $\rho = \{\mathcal{X}, \{\mathcal{W}_X\}_{X\in\mathcal{X}}, \{V_1, ..., V_k\}, \mathcal{H}\}$ is a valuation-based system. We shall say that ρ is *consistent* if $\otimes\{V_1,...,V_k\}$ is a proper valuation and *inconsistent* otherwise. We will refer to $\otimes\{V_1,...,V_k\}$ as the *joint valuation*. Note that the joint valuation is a valuation for $\mathcal{X}$.

If a valuation-based system has, say, 50 variables, and each variable has, say, 2 configurations then the frame of the joint variable $\mathcal{X}$ will have 2^{50} configurations. Thus, in this case, it will be computationally infeasible to explicitly compute the joint valuation in order to verify whether it is consistent or not. In the next section, we describe an efficient method for verifying if a VBS is consistent or not without explicitly computing the joint valuation.

5. A METHOD FOR CHECKING FOR CONSISTENCY

In Section 3, we stated that a marginal $G^{\downarrow h}$ is proper if and only if G is proper. Thus, one way of checking whether or not a VBS is proper is to verify whether the marginal of the joint valuation for some subset of joint variables is proper or not. In this section, we describe a method for computing exactly the marginal of the joint valuation for some subset without explicitly computing the joint valuation.

Table 3. Modus tollens represented as combination and marginalization. G_4 represents proposition Y=~y and G_2 represents conditional If X=x then Y=y. $(G_4 \otimes G_2)^{\downarrow\{X\}}$ represents the conclusion X=~x.

w	$G_4(w^{\downarrow\{Y\}})$	$G_2(w)$	$(G_4 \otimes G_2)(w)$
(x,y)	0	1	0
(x,~y)	1	0	0
(~x,y)	0	1	0
(~x,~y)	1	1	1

w	$(G_4 \otimes G_2)^{\downarrow\{X\}}(w)$
x	0
~x	1

Table 4. Disjunctive syllogism represented as combination and marginalization. G_5 represents proposition X=~x and G_6 represents disjunction X=x or Y=y. $(G_5 \otimes G_6)^{\downarrow\{Y\}}$ represents the conclusion Y=y.

w	$G_5(w^{\downarrow\{X\}})$	$G_6(w)$	$(G_5 \otimes G_6)(w)$
(x,y)	0	1	0
(x,~y)	0	1	0
(~x,y)	1	1	1
(~x,~y)	1	0	0

w	$(G_5 \otimes G_6)^{\downarrow\{Y\}}(w)$
y	1
~y	0

Table 5. Disjunctive elimination represented as combination and marginalization. G_6 represents disjunction X=x or Y=y. G_7 represents conditional If X=x then Z=z, G_8 represents the conditional If Y=y then Z=z. $(G_6 \otimes G_7 \otimes G_8)^{\downarrow\{Z\}}$ represents the conclusion Z=z.

w	$G_6(w^{\downarrow\{X,Y\}})$	$G_7(w^{\downarrow\{X,Z\}})$	$G_8(w^{\downarrow\{Y,Z\}})$	$(G_6 \otimes G_7 \otimes G_8)(w)$
(x,y,z)	1	1	1	1
(x,y,~z)	1	0	0	0
(x,~y,z)	1	1	1	1
(x,~y,~z)	1	0	1	0
(~x,y,z)	1	1	1	1
(~x,y,~z)	1	1	0	0
(~x,~y,z)	0	1	1	0
(~x,~y,~z)	0	1	1	0

w	$(G_6 \otimes G_7 \otimes G_8)^{\downarrow\{Z\}}(w)$
z	1
~z	0

Suppose $\rho = \{\mathcal{X}, \{\mathcal{W}_X\}_{X\in\mathcal{X}}, \{V_1, ..., V_k\}, \mathcal{H}\}$ is a valuation-based system. Let h_1 denote some element of $\mathcal{H}$. We will now describe a method for computing the marginal $(\otimes\{V_1,...,V_k\})^{\downarrow h_1}$.

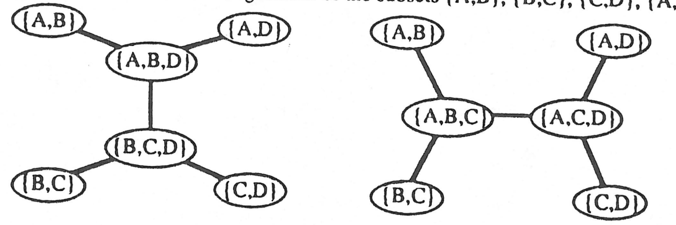
The computation of the marginal proceeds in two phases. In the first phase, we arrange the subsets of variables in $\mathcal{H}$ in a "Markov tree". In the second phase, we "propagate" the valuations $\{V_1, ..., V_k\}$ in the Markov tree using a local message-passing scheme resulting in the computation of the desired marginal.

5.1. Finding a Markov Tree Arrangement

A *Markov tree* is a topological tree whose vertices are subsets of variables with the property that when a variable belongs to two distinct vertices, then every vertex lying on the path between these two vertices contains the variable.

First, the only information we need in this phase is the set $\mathcal{H}$ of subsets of variables for which we have valuations in the VBS. Second, in the process of arranging a set of subsets in a Markov tree, we may have to add some subsets to the hypergraph $\mathcal{H}$. Third, in general, there are many Markov tree arrangements of a hypergraph. Figure 1 shows two Markov tree arrangements of the subsets $\{A,B\}$, $\{B,C\}$, $\{C,D\}$, $\{A,D\}$.

Figure 1. Two Markov tree arrangements of the subsets $\{A,B\}$, $\{B,C\}$, $\{C,D\}$, $\{A,D\}$.



The computational efficiency of the second phase depends on the sizes of the frames of the vertices of the Markov tree constructed in the first phase. However, finding an optimal Markov tree (a Markov tree whose largest frame is as small as possible) has been shown to be an NP-complete problem (Arnborg *et al.* [1987]). Thus we have to balance the computational efforts in the two phases. We should emphasize, however, that this is strictly a computational effort question. If computational effort is not an issue, then it does not matter which Markov tree is used for propagating the valuations. All Markov trees give the same final answer, i.e., the marginal of the joint valuation for some subset.

The method described below for arranging a hypergraph in a Markov tree is due to Kong [1986] and Mellouli [1987].

Suppose $\mathcal{H}$ is a hypergraph on $\mathcal{X}$. To arrange the subsets in $\mathcal{H}$ in a Markov tree, we first pick a sequence of variables in $\mathcal{X}$. As we shall see, each sequence of the variables gives rise to a Markov tree arrangement. Mellouli [1987] has shown that an optimal Markov tree arrangement can be found by picking some sequence. Of course, since there are an exponential number of sequences (n! to be exact), finding an optimal sequence is, in general, a difficult problem.

Consider the following set of instructions in pseudo-basic:

```
INITIALIZATION: $u \leftarrow \mathcal{X}$, $\mathcal{H}_0 \leftarrow \mathcal{H}$, $\mathcal{V} \leftarrow \varnothing$, $\mathcal{E} \leftarrow \varnothing$.
FOR i = 1 to n by 1;
    Pick a variable in set u and call it $X_i$
    $u \leftarrow u - \{X_i\}$
    $g_i \leftarrow \cup\{h \in \mathcal{H}_{i-1} \mid X_i \in h\}$.
    $f_i \leftarrow g_i - \{X_i\}$.
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{h \in \mathcal{H}_{i-1} \mid X_i \in h\} \cup \{f_i\} \cup \{g_i\}$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{\{g_i,h\} \mid h \in \mathcal{H}_{i-1}, h \neq g_i, X_i \in h\} \cup \{\{f_i,g_i\}\}$
    $\mathcal{H}_i \leftarrow \{h \in \mathcal{H}_{i-1} \mid X_i \notin h\} \cup \{f_i\}$
    IF $\mathcal{H}_i$ consists only one subset THEN GOTO STOP
ENDFOR
STOP
```

After the execution of the above set of instructions, it is easily seen that the pair $(\mathcal{V}, \mathcal{E})$ is be a Markov tree arrangement of $\mathcal{H}$ where $\mathcal{V}$ denotes the set of vertices of the Markov tree and $\mathcal{E}$ denotes the set of undirected edges. Note that at each iteration of the above sequence of

instructions, subsets g_i and f_i are added to the set of subsets if they are not already members of $\mathcal{H}$.

We shall say that in the ith iteration of the FOR-loop in the above set of instructions, the variable X_i that is picked from set u is *marked*. Note that the subsets in $\mathcal{H}_i$ only contain unmarked variables.

A heuristic called *one-step-look-ahead* has been suggested by Kong [1986] for finding a good Markov tree. This heuristic tells us which variable to mark next. As the name of the heuristic suggests, the variable that should be marked next is an unmarked variable X_i such that the cardinality of $\mathcal{W}_{f_i}$ is the smallest. Thus the heuristic attempts to keep the sizes of the frames of the added vertices as small as possible by focussing only on the next subset added. See Zhang [1988] for other heuristics for efficient Markov tree construction.

5.2. Propagating Valuations in Markov Trees

Suppose we have arranged the hypergraph $\mathcal{H}$ in a Markov tree. Let $\mathcal{H}'$ denote the set of all subsets in the Markov tree. Clearly $\mathcal{H}' \supseteq \mathcal{H}$. To simplify the exposition, we will assume that there is exactly one valuation for each subset $h \in \mathcal{H}'$. (If h is a subset that was added during the Markov tree construction process, then we can associate the vacuous valuation (the valuation whose values are all 1) with it. On the other hand, if subset h had more than one valuation defined for it, then we can combine all of these valuations to obtain one valuation).

First, note that the Markov tree defines a neighborhood structure for each subset $h_i \in \mathcal{H}'$. Subset h_j is a *neighbor* of h_i if there is an edge $\{h_i,h_j\}$ in the Markov tree. Let N(h) denote the set of subsets in $\mathcal{H}'$ that are neighbors of h. Second, let us designate h_1 (the subset for which the marginal is desired) as the *root* of the Markov tree. Thus for each subset $h \neq h_1$ in the Markov tree, there will be a unique neighbor of h that will lie on the path from h to h_1. We will refer to this neighbor as h's *parent* and denote it by $\pi(h)$, and we will refer to h's other neighbors (if any) as its *children*.

In the propagation process, each subset (except the root h_1) transmits a valuation to its parent. We shall refer to the valuation transmitted by subset h_i to its parent $\pi(h_i)$ as a *message* and denote it by $M^{h_i \rightarrow \pi(h_i)}$. Suppose $\mathcal{H}' = \{h_1, \ldots, h_p\}$ and let H_i denote the valuation associated with subset h_i. Then, the message transmitted by a subset h_i to its parent is given by

$$M^{h_i \rightarrow \pi(h_i)} = (\otimes\{M^{h \rightarrow h_i} \mid h \in (N(h_i) - \{\pi(h_i)\})\} \otimes H_i)^{\downarrow(h_i \cap \pi(h_i))} \tag{1}$$

In words, the message transmitted by a subset to its parent consists of the combination of all the messages it receives from its children plus its own valuation suitably marginalized. Note that the combination operation that is performed in (1) is on the frame $\mathcal{W}_{h_i}$.

Expression (1) is a recursive formula. We need to start the recursion somewhere. Note that if subset h_i has only one neighbor (its parent), then $(N(h_i) - \{\pi(h_i)\}) = \varnothing$ and the expression in (1) reduces to

$$M^{h_i \rightarrow \pi(h_i)} = (H_i)^{\downarrow(h_i \cap \pi(h_i))} \tag{2}$$

Thus the *leaves* of the Markov tree (the subsets that have no children) can send messages to their parents right away. The others wait until they have heard from all their children before they send a message to their parent.

After the root h_1 has received a message from each of its children, it combines all the messages it receives from its children with it own valuation, i.e., it computes

$$(\otimes\{M^{h \rightarrow h_1} \mid h \in N(h_1)\}) \otimes H_1. \tag{3}$$

The following theorem states that the result of the computation in (3) is indeed the desired marginal.

Theorem 1. The marginal of the joint valuation of ρ for h_1 can be computed as in (3), i.e., $(\otimes\{V_i \mid i=1,\ldots,k\})^{\downarrow h_1} = (\otimes\{M^{h \rightarrow h_1} \mid h \in N(h_1)\}) \otimes H_1$.

The essence of the propagation method described above is to perform combinations of valuations on smaller frames instead of combining all valuations on the global frame associated with $\mathfrak{X}$. To ensure that this method gives us the correct answers, the smaller frames have to be arranged in a Markov tree.

If the VBS ρ is consistent, then the marginal of the joint for h_1 is of course proper. If ρ is inconsistent, then this will be manifested by the marginal of the joint for h_1 being improper. Depending on which subsets of valuations are inconsistent, inconsistency may be detected earlier during the combination operation (in expression (1)) at some vertex h_i.

Shenoy [1990] describes an example in detail illustrating the construction of a Markov tree and illustrating the propagation of valuations.

6. CONCLUSIONS

We have described an efficient method for checking whether a knowledge-base is consistent or not.

In the propagation scheme described in Section 5.2, messages were only sent one way towards the root. If messages are sent back down from the root to the leaves, we could compute the marginals for every subset in the tree [Shenoy and Shafer, 1990]. And if we included all singleton subsets in the Markov tree, then we would have marginals of the joint valuation for each variable in the system. As we explained earlier in Section 3, finding the marginal of the joint valuation is a form of logical inference. Thus the method we have described can also be used to make inferences from a valuation-based system. Shenoy [1989] describes such a language that uses valuations to encode knowledge and uses the combination and marginalization operations to make inferences from the knowledge-base.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grant IRI-8902444.

REFERENCES

Arnborg, S., Corneil, D. G. and Proskurowski, A. (1987), Complexity of finding embeddings in a k-tree, *SIAM Journal of Algebraic and Discrete Methods*, 8, 277-284.

Ginsberg, A. (1988), Knowledge-base reduction: A new approach to checking knowledge bases for inconsistency and redundancy, *Proceedings of 7th National Conference on AI (AAAI-88)*, St. Paul, MN, 1, 585-589.

de Kleer, J. (1986), An assumption-based TMS, *Artificial Intelligence*, 28, 127-162, 1986.

Kong, A. (1986), Multivariate belief functions and graphical models, Ph.D. thesis, Department of Statistics, Harvard University, Cambridge, MA.

Mellouli, K. (1987), On the propagation of beliefs in networks using the Dempster-Shafer theory of evidence," Ph.D. dissertation, School of Business, University of Kansas, Lawrence, KS.

Nguyen, T.A., Perkins, W.A., Laffey, T.J. and Pecora, D. (1985), Checking an expert systems knowledge base for consistency and completeness, *Proceedings of the 9th International Joint Conference on AI (IJCAI-85)*, Los Angeles, CA, 1, 375-378.

Shenoy, P. P. (1989), A valuation-based language for expert systems, *International Journal for Approximate Reasoning*, 3(5), 383-411.

Shenoy, P. P. (1990), Consistency in valuation-based systems, Working Paper No. 216, School of Business, University of Kansas, Lawrence, KS.

Shenoy, P. P. (1990b), Valuation-based systems for discrete optimization, *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, Boston, MA, to appear.

Shenoy, P. P. and Shafer, G. (1990), Axioms for probability and belief function propagation, *Uncertainty in Artificial Intelligence*, 4, edited by R. Shachter, T. Levitt, J. Lemmer and L. Kanal, North-Holland, to appear.

Suwa, M., Scott, A. C. and Shortliffe, E. H. (1982), An approach to verifying completeness and consistency in a rule-based expert system, *AI Magazine*, 3(3), 16-21.

Zhang, L. (1988), Studies on finding hypertree covers of hypergraphs, Working Paper No. 198, School of Business, University of Kansas, Lawrence, KS.

HIERARCHICAL PLANNING USING ACTION TAXONOMIES*

Hua Shu
Dept. of Computer and Information Science
Linköping University, S-582 81 Linköping, Sweden
huash@ida.liu.se

Abstract

This paper proposes a formulation of hierarchical planning using action taxonomies and action structures. The problem of coordinating *abstraction levels* and *planning levels* is addressed. *Action taxonomies*, characterizing decomposition and inheritance relations between actions with temporal constraints associated, are used to represent the abstraction levels of the actions in a domain. The planning levels are represented in terms of *hierarchical action structures*. A plan refinement algorithm extends the abstract plan from one abstraction level to the lower one. Especially the planning levels may locally interleave with the abstraction levels to choose the specification of an inheritance abstraction during the plan refinement.

1 Introduction

Abstraction is a way of suppressing detail in order to focus on important aspects of a problem first, instead of considering everything at once. The basis of hierarchical planning is the use of abstraction both in the planning process and in the domain description to avoid combinational explosion and obtain computational tractability. *Abstraction levels* are determined by the degree to which the world is granulated. *Planning levels* [9] are various stages on which a plan is developed. For many hierarchical planners, their planning levels do not correspond to the abstraction levels of the domain knowledge; few planners have prescribed structures formalizing the *abstraction levels* of the domain knowledge. Typically, They account hierarchical structures used for the planning process rather than the abstraction levels of the domain knowledge. Such planners often suffer from problems such as *hierarchical promiscuity* [1] and planning ineffectiveness. After all, they are impeded to take full advantage of abstraction technique.

We use a hierarchical structure, *action taxonomy*, to represent the abstraction levels of the knowledge about actions. An augmented action taxonomy is an integration of decompositional and inheritance hierarchies, associated with temporal constraints between actions. Planning is regarded as a two-phase process by which we can *synchronize* the abstraction and planning levels. *Projection* first results in an abstract action based on a behavior description. *Refinement* then generates detailed plans using an action taxonomy. Actions are described as transformations between partial states, and plans are represented using *action structures*. The *coherence* of an action structure implies the *validity* of the abstract plan. The evolution of the abstract plans is represented using *hierarchical action structure* which is also the data structure for the coordination between planning levels and abstraction levels.

*This work was supported by the Swedish Board of Technical Development and the IT4 research program.

[1]exhibited by a planner which ensures neither that all relevant information at the proper abstraction level is available for the actions at a planning level, nor that subsequent expansions to lower abstraction levels will not change the truth values of some features referred the previous abstraction level [9].

© 1990 by Elsevier Science Publishing Co., Inc.
Methodologies for Intelligent Systems, 5
Z.W. Ras, M. Zemankova, and M.L. Emrich, Editors