

Title	A Deductive Object-Oriented Database System for Situated Inference in Law
Author(s)	Wong, Stephen; Tojo, Satoshi
Citation	IEEE Transactions on Knowledge and Data Engineering, 8(3): 496-503
Issue Date	1996-06
Type	Journal Article
Text version	publisher
URL	http://hdl.handle.net/10119/4651
Rights	Copyright (c)1996 IEEE. Reprinted from IEEE Transactions on Knowledge and Data Engineering, 8(3), 1996, 496-503. This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of JAIST's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org . By choosing to view this document, you agree to all provisions of the copyright laws protecting it.
Description	

- 1) the basis of each,
- 2) the limitations of each (for example, the argument-type scheme does not link slot classes by the subclass relation),
- 3) the benefits of keeping the schemes separate (uniform differentiation), and
- 4) the benefits of having all available to the system user (alternate search techniques).

Also, we emphasize that the relation elements, such as *functional* and *separable*, play an important role in representing—with slot classes and individual slots—the underlying properties of semantic relations.

In Cyc, each of the separate slot hierarchies can be implemented in its own “context” or microtheory [7], and slot retrieval and storage can be made within the context selected by the user. In effect, we can “turn-on” any of three separate slot organization schemes and search for existing slots in any of them. There is, however, additional effort required in representing a new slot since it can be placed in up to three different hierarchies. Nevertheless, the different slot hierarchies have well defined principles and can facilitate knowledge use by making it easier for system users to find slots they need.

REFERENCES

- [1] I.I. Bejar, R. Chaffin, and S. Embretson, *Cognitive and Psychometric Analysis of Analogical Problem Solving*. New York: Springer-Verlag, 1991.
- [2] R. Chaffin and D.J. Herrmann, “Relation Element Theory: A New Account of the Representation and Processing of Semantic Relations,” *Learning and Memory: The Ebbinghaus Centennial Conf.*, D. Gorfein and R. Hoffman, eds. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1987.
- [3] D.A. Cruse, *Lexical Semantics*. Cambridge: Cambridge Univ. Press, 1986.
- [4] J. Doyle and R.S. Patil, “Two Theses of Knowledge Representation: Language Restrictions, Taxonomic Classification, and the Utility of Representational Services,” *Artificial Intelligence J.*, vol. 48, no. 3, pp. 261-297, 1991.
- [5] *Relational Models of the Lexicon*, M.W. Evens, ed. Cambridge: Cambridge Univ. Press, 1988.
- [6] R. Fikes and T. Kehler, “The Role of Frame-Based Representation in Reasoning,” *Comm. ACM*, vol. 28, no. 9, pp. 904-920, Sept. 1985.
- [7] R.V. Guha and D.B. Lenat, “Enabling Agents to Work Together,” *Comm. ACM*, vol. 37, no. 7, pp. 127-142, July 1994.
- [8] M.N. Huhns and L.M. Stephens, “Plausible Inferencing Using Extended Composition,” *Proc. 11th Int'l Joint Conf. Artificial Intelligence, IJCAI-89*, pp. 1420-1425, Detroit, Aug. 1989.
- [9] D.B. Lenat and R.V. Guha, *Building Large Knowledge-Based Systems*. Reading, Mass.: Addison-Wesley, 1990.
- [10] D.B. Lenat and R.V. Guha, “The Evolution of CycL, The Cyc Representation Language,” *SIGART Bulletin*, vol. 2, no. 3, pp. 84-87, June 1991.
- [11] A. Newell, “The Knowledge Level,” *AI J.*, vol. 19, no. 2, pp. 87-127, 1982.
- [12] K. Pittman and D.B. Lenat, “Representing Knowledge in Cyc-9: An Introduction,” MCC Technical Report no. CYC 175-93P, Microelectronics and Computer Technology Corp., Austin, Tex., Dec. 1993.
- [13] L.M. Stephens and Y.F. Chen, “Principles for Organizing Semantic Relations in Large Knowledge Bases,” Technical Report no. LMS-95-10, Dept. of Electrical and Computer Eng., Univ. of South Carolina, Oct. 1995. On Web page: <http://www.ece.sc.edu/Labs/dai.html>.
- [14] M.E. Winston, R. Chaffin, and D. Herrmann, “A Taxonomy of Part-Whole Relations,” *Cognitive Science*, vol. 11, pp. 417-444, 1987.
- [15] W.A. Woods, “Understanding Subsumption and Taxonomy: A Framework for Progress,” *Principles of Semantic Networks*, J.F. Sowa, ed., pp. 45-94. San Mateo, Calif.: Morgan Kaufmann, 1991.
- [16] W.A. Woods and J.G. Schmolze, “The KL-ONE Family,” *Computer & Mathematics*, vol. 23, no. 2-5, pp. 133-177, 1992.

A Deductive Object-Oriented Database System for Situated Inference in Law

Stephen Wong and Satoshi Tojo

Abstract—Deductive Object-Oriented Databases and Situation Theory are two important areas of research in the fields of database and of linguistics. AI and Law is a new field attracting both AI researchers and legal practitioners. Our research brings together the former two fields with the aim of designing knowledge applications in the latter. This is achieved through a formal model for legal reasoning, *SM*, and a deductive object-oriented database system, *QUIXOTE*. The purpose of this paper is to introduce the key features of this formal model, based on situation theory, and to describe how this database system can implement this abstract model for complex legal reasoning applications. Concrete examples from legal precedents are used to illustrate these advanced features.

Index Terms—Deductive object-oriented databases, AI and law, knowledge base management systems, situation theory.

1 INTRODUCTION

THE research issues in Artificial Intelligence (AI) and Law include the interpretation of open-textured concepts, reasoning by cases and rules, creating computational decision making models that embody the norms of society, and drawing arguments under opposing viewpoints and different situations. Typically, a legal reasoning system draws arguments by interpreting judicial precedents (old cases) or statutes (legal rules) encoded in its knowledge base, and a more advanced system includes both kinds. Surveys on the leading projects can be found in [13], [14], [7], [1]. Most implementations are written in AI programming languages, such as Prolog or Lisp, and contain only small sets of cases and rules. They cannot access and manipulate large amounts of data and lack database management services such as concurrency control, nested transactions, and data persistence. Reasoning in law, however, is a knowledge-intensive endeavor. The lack of tools to scale up these legal reasoning prototypes is a major handicap to the growth and potential contributions of this interdisciplinary field. On the other hand, the database (DB) community has yet to develop tools which are expressive enough to satisfy the data modeling needs of the AI and Law researchers.

Legal reasoning systems has been a key research activity in the Fifth Generation Computer System (FGCS) project [11], [12]. This project devised a formal model of legal argumentation, *SM*, [16], based on situation theory [3], [2], and developed a Deductive Object-Oriented Database (DOOD) System, *QUIXOTE* [18], whose representation language can map the conceptual formulation into a computational form on the Parallel Inference Machines (PIM) [15]. The legal reasoning system developed includes a control program and a set of knowledge bases. The control program is written in the parallel logic programming language, KL1 [5]. The set of knowledge bases includes a dictionary of legal ontologies, a database of old cases, and a database of statutes. In this paper, we dis-

- S. Wong is with the Department of Radiology, University of California-San Francisco, Box 0628, San Francisco, CA 94143.
E-mail: steven_wong@radmacl.ucsf.edu.
- S. Tojo is with the Japan Advanced Institute of Science and Technology, Asahidai 15, Tatsunokuchi-machi, Nomi-gun, Ishikawaken 923-12, Japan.
E-mail: tojo@jaist.ac.jp.

Manuscript received July 6, 1993; revised Aug. 18, 1994.

For information on obtaining reprints of this article, please send e-mail to: transkde@computer.org, and reference IEEECS Log Number K96040.

cuss the specific features of the *QUIXOTE* system, that can be used to support situated inference and to manage legal databases of various sorts. In addressing the complex issues of AI and law, this study has brought together two previously unrelated fields, deductive object-oriented database and situation theory. This work, to our knowledge, is the first attempt to provide an advanced knowledge base management system tool to build large scale knowledge systems for legal reasoning applications. This paper is organized as follows. Section 2 describes the modeling of legal knowledge and reasoning at the abstraction level, using the theory of situations. Section 3 discusses the realization of this formulation at the database level using *QUIXOTE*. Section 4 illustrates situated inference mechanisms supported by this database system, and presents legal examples. We discuss related work and conclude the paper in Section 5.

2 FORMAL REPRESENTATION OF LEGAL KNOWLEDGE

As our formulation of legal inference is based on *situation theory*, we call it a *situation-theoretic model (SM)*. A legal concept exhibits *open texture* in that it is precisely defined *only* for those cases which have been decided by a court. The interpretation of such vague and discretionary legal concepts depends on the situations surrounding new cases. Many problems in natural language understanding are also ascribed to such situation dependency, and various semantics have been proposed, e.g., situations [2] and DRT [9]. One advantage of situation theory is its uniform way of representing various kinds of *situatedness*, i.e., $s \models \sigma$, the interpretation of a phrase or sentence, σ , under the scope of a situation s . Our observation is that legal situations can be defined abstractly in terms of a set of infons or sentences about a case. The presumption is that abstract situations and the constraints between them would describe the *logical* flow of information in real situations [6] and would therefore be useful to the design of legal reasoning systems.

2.1 General Terms

The ontologies of *SM* include objects, parameters, relations, infons, and situations. An object designates an individuated part of the real world: a constant or an individual in the sense of classical logic. A parameter refers to an arbitrary object of a given type. An n -placed relation is a property of an n -tuple of argument roles, r_1, \dots, r_n , or slots into which appropriate objects of a certain type can be anchored or substituted. An infon σ is written as $\langle\langle Rel, a_1, \dots, a_n, i \rangle\rangle$, where *Rel* is a relation, each argument term a_k is a constant object or a parameter, and i is a polarity indicating 1 or 0 (true or false). If an infon contains an n -place relation and m argument terms such that $m < n$, we say that the infon is *unsaturated*. If $m = n$, it is *saturated*. Any object assigned to fill an argument role of the relation of that infon must be of the appropriate type or must be a parameter that can only anchor to objects of that type. Argument roles that must be filled to result in a saturated infon is dependent upon what the relation is [16].

An infon that has no free parameters is called a *parameter-free* infon; otherwise, it is a *parametric* infon. If σ is an infon and f is an anchor for some or all of the parameters that occur free in σ , we denote, by $\sigma[f]$, the infon that results by replacing each v in the domain of f that occurs free in σ with its value (object constant) $f(v)$. If I is a set of parametric infons and f is an anchor for some or all of the parameters that occur free in I , then $I[f] = \{\sigma[f] \mid \sigma \in I\}$. In addition, an abstract situation is said to be *coherent* if it does not support both an infon and its negation. If an infon is of polarity 1, its negation is of polarity 0. Two abstract situations s and s' are said to be *compatible* if their union is a coherent situation. The situations within a legal case are presumed to be compatible with one another, but no such presumption can be made across different cases.

An *SM* is a triplet $\langle \mathcal{P}, \mathcal{A}, \models \rangle$, where \mathcal{P} is a collection of abstract

situations including judicial precedents, a new case, c_n , and a world, w , that is a unique maximal situation of which every other situation is a part; \mathcal{A} are the defendant and plaintiff agents; and \models is the support relation. The latter satisfies the following conditions [6]:

CONDITION 2.1 (Supports Relation):

- 1) For any $s \in \mathcal{P}$, and any atomic infon σ , $s \models \sigma$ if and only if (iff) $\sigma \in s$.
- 2) For any s , any σ, β, a For any s that contains (as constituents) all members of u , $s \models (\exists x \in u)\sigma$ iff there is an anchor, f , of a parameter, x , to an element of u , such that $s \models \sigma[f]$, and b) $s \models (\forall x \in u)\sigma$ iff for all anchors, f , of x to an element of u , we have $s \models \sigma[f]$.
- 3) For any $s \in \mathcal{P}$, and any set of infons I , $s \models I$ if $s \models \sigma$ for every infon σ in I .

The notation $s \models \sigma$ thus denotes a proposition about σ whose truth values are situation-dependent, whereas $w \models \beta$ asserts that β is universally true. In addition, let v be a parameter. By a *condition* on v we mean any finite set of parametric infons. (At least one of these should involve v , otherwise, the definition is degenerate). We define a new parameter, $v \parallel C$, called a *restricted parameter*. $v \parallel C$ will denote an object of the same basic type as v , that satisfies the requirements imposed by C . This amounts to our placing a more stringent requirement on anchors.

2.2 Concept Matching

We introduce certain specific terms, *relevance level*, *infon matching*, and *situation matching*, to extend the general *SM* terms into the legal domain. In a legal event, an agent would consider some facts (infons) to be more relevant than others in reaching an argument. To estimate such weighting on facts, *SM* assigns every infon in an old case with a level of relevance. For example, the restricted parameter $\hat{\sigma} = \sigma \parallel \langle\langle \text{relevance-level}, \sigma, \lambda, 1 \rangle\rangle$, where λ denotes a certain weight of relevance. One distinction of legal reasoning is the matching of the new facts with those of precedents to generate similar arguments which may hold in the new case [11], [1]. No two events are exactly alike, but the idea of precedent-based matching presupposes that a prior decision will control subsequent facts that are like the first. Yet, given the lack of absolute identity, the decision-maker of the new case must evaluate the determinant of likeness. To this end, *SM* adopts a concept of structural matching. Since cases are composed of infons, the model first defines the matching relation between these basic units of information. A case infon is always parameter-free.

CONDITION 2.2. For c_n and an old case c_o , $\sigma_n = \langle\langle Rel_1, a_1, \dots, a_n, i_1 \rangle\rangle \in c_n$, $\sigma_o = \langle\langle Rel_2, b_1, \dots, b_m, i_2 \rangle\rangle \in c_o$, a) (Exact Infon Matching): $\sigma_n \approx_{iem} \sigma_o$ iff i) $m = n$; ii) $i_1 = i_2$; iii) Rel_1 and Rel_2 are of the same type; iv) for every argument a_i of a non-infon type, there exists b_k which is of the same role or type and has not been matched with another argument; v) for every a_i of an infon type, there exists b_o that satisfies the same set of conditions, and b) (Partial Infon Matching): $\sigma_n \approx_{ipm} \sigma_o$ if $m \leq n$ and all argument terms of σ_o are matched.

Where a *Rel* b intends to denote $w \models \langle\langle Rel, a, b, 1 \rangle\rangle$. Clearly, \approx_{iem} is an equivalence relation while \approx_{ipm} is an asymmetric relation. Infon-matching relations are the building blocks for defining situation-matching relations.

CONDITION 2.3. a) (Exact Situation Matching) For any $s_n \subseteq c_n$, $s_o \subseteq c_o$, $s_n \approx_{sem} s_o$ iff for every σ of $s_o \models \sigma$, there exists ρ of $s_n \models \rho$ such that $\sigma \approx_{iem} \rho$, and vice versa; b) (Partial Situation Matching) For any $s_n \approx_{spm} s_o$ iff for every σ of $s_o \models \sigma \parallel \langle\langle \text{relevance-level}, \sigma, l, 1 \rangle\rangle$ s.t. $l \geq l_o$, there exists ρ of $s_n \models \rho$ s.t. $\sigma \approx_{ipm} \rho$.

When there is no confusion, we write \approx_s to denote a matching relation between situations and \approx_i between infons.

2.3 Situated Inference Rules

A legal reasoning process can be modeled as an inference tree of four layers. The bottom layer consists of a set of basic facts and hypotheses, the second involves case rules of individual precedents, the third involves case rules which are induced from several precedents or which are generated from certain legal theory, and the top layer concerns legal rules derived from statutes. An individual or local case rule is used by an agent in an old case to derive plausible legal concepts and propositions. These rules vary from case to case, and their interpretation depends on particular views and priorities. An induced case rule has a broader scope and is generalized from a set of precedents. Legal rules are general provisions and definitions of crimes. The applicability of these rules is independent of the view of either side (plaintiff or defendant) and every item of information (infor) included is of equal relevance. Though it rarely happens, it may be possible for an agent to skip one or two case rule layers in attaining a legal goal. Further, a local case rule is as follows:

RULE 2.1 (Local Rule): For $c \in \mathcal{P}$, $cr : c \models \sigma \Leftarrow c \models I/B_{cr}$.

Where I is called the antecedent of the rule, σ is the consequent infor, and cr is the label of the rule, which is not itself part of the rule but which serves to identify the rule. Sometimes, we simply write $cr : c \models \sigma \Leftarrow I/B_{cr}$. Both σ and I are parameter-free. The reliability and the scope of application of a local rule will be subject to a set of *background conditions*, B_{cr} . The conditions include information such as an agent's goal and hypotheses; these are crucial in debate to establish the degree of certainty and the scope of applicability of that rule. Usually, it becomes necessary to take background conditions into account and investigate what they are when the conclusion drawn from the case rule leads to conflict with others or a change in circumstances that weakens the applicability of that rule.

Denote I' and σ' as two sets of parametric infons such that all parameters that occur in the latter also appear in the former. An induced rule and a legal rule are represented as:

RULE 2.2 (Induced Rule): For any $c_1, \dots, c_k \in \mathcal{P}$, $c = c_1 \cup c_2 \cup \dots \cup c_k$, $ir : c \models \sigma' \Leftarrow I'/B_{ir}$.

RULE 2.3 (Legal Rule): $lr : w \models \sigma' \Leftarrow I'/B_{lr}$.

where all cases are coherent and ir and lr are the rule labels.

2.4 Substitution and Anchoring

When a situation of a new case, c_n , supports a similar antecedent of a local rule of c_o , one can draw a conclusion about the new case similar to the consequent of that rule. That is,

RULE 2.4 (Local Rule Substitution): For $c_n, c_o \in \mathcal{P}$, $cr^s : c_n \models \sigma\theta$ if $cr : c_o \models \sigma \Leftarrow I/B_{cr}$ and $c_n \models I'/\{B_{cr}\theta \cup B_n\}$ such that $I' \approx_s I$.

Where cr^s is the label of the new rule, B_n is the original background of the new case, I' , and the combined condition after the substitution, $B = B_{cr}\theta \cup B_n$, is coherent. The function θ forms a *link* that connects c_n with c_o and replaces all terms (objects and relations) in σ and B_{cr} that also occur in I with their matched counterparts in I' . Fig. 1 presents a substitution diagram that does not include the background conditions. Referring to Rule 2.4, the substitution merely replaces terms and does not change the polarities of infons. Also, the information of case matching B_n is not related to $B\theta$ and thus does not create compatibility problems. It thus follows that $\{c_n \cup B\}$ is coherent.

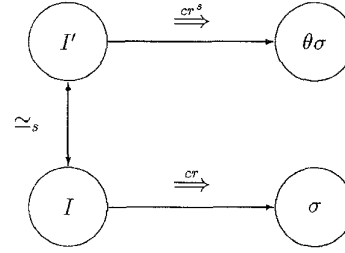


Fig. 1. Case substitution.

In a court case, both sides (plaintiff and defendant) are normally ignorant about the assumptions and hypotheses of each other's claims. An essential technique, used to reveal such 'hidden' information, is cross-examination. Incorporating the background conditions into legal constraints (case and legal rules) allows us to capture this essential feature of legal reasoning for knowledge-based applications. Rather than substitution, a consequent is derived from a legal rule.

RULE 2.5. a) (Induced Rule Anchoring) For $c_n, c_1, \dots, c_k \in \mathcal{P}$, such that $c = \{c_1, c_2, \dots, c_k\}$, $ir^s : c_n \models \sigma[f]$ if $ir : c \models \sigma \Leftarrow I/B_{ir}$ and $c_n \models I[f]/\{B_{ir}[f] \cup B_n\}$, b) (Legal Rule Anchoring) for $c_n \in \mathcal{P}$, $lr^s : c_n \models \sigma[f]$ if $lr : w \models \sigma \Leftarrow I/B_{lr}$ and $c_n \models I[f]/\{B_{lr}[f] \cup B_n\}$

where, c_n , again, is the new case and B_n is the background condition of this new case. Fig. 2 shows a legal inference example of *SM* in the forward reasoning manner. For simplicity, this inference involves only local and legal rules. The black circles, I_1, I_2' , and σ , denote the situations of a new case, c_n while situations I_1 and I_2 are of old cases. Two immediate arguments, β_1 and β_2 are drawn using local rules cr_1 and cr_2 . Together with $\{\sigma\}$, the goal $\gamma[f]$ is anchored or attained by applying the legal rule lr . From the case coherency condition, it follows that all concepts of a single goal tree must share the same legal perspective: the plaintiff's or the defendant's.

This figure indicates that the matching relation of I_1 is stronger than that of I_2 . One can also probe into background conditions, linked by appropriate rule labels, of these arguments to retrieve the underlying hypotheses and legal theories.

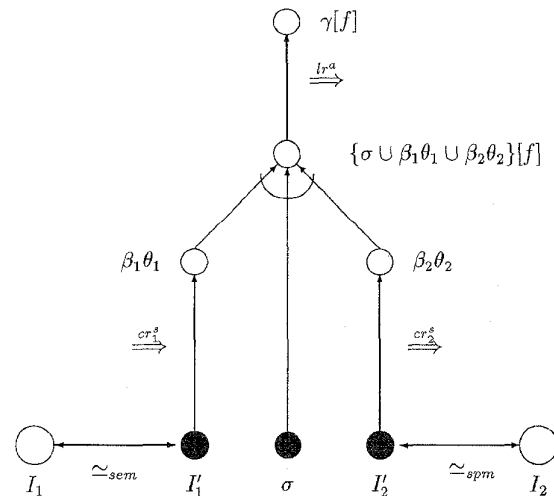


Fig. 2. A visual representation of legal inference.

3 MODELING OF LEGAL KNOWLEDGE IN QUIXOTE

This section introduces the *QUIXOTE* language and shows how it can be used to map the *SM* concepts in computable form. The *QUIXOTE* language [18] is a hybrid of the deductive object-oriented database (DOOD) language [4] and constraint logic programming (CLP) language [8]. It resembles F-logic [10], in that it is a DOOD language that includes powerful extensions into logic programming such as subsumption, complex objects, and modules. Compared with conventional CLP language, *QUIXOTE* has a symbolic constraint domain which makes it suitable for describing the legal situations depicted in legal documents written in, albeit tight and formal, natural language. A typical *QUIXOTE* database includes the following:

- 1) the subsumption relations among basic objects,
- 2) the submodule relations among modules, and
- 3) rules.

Our legal reasoning system consists of three distinct databases, namely, a dictionary, a case base, and a statute base. Accordingly, we first introduce the *objects* and *modules* of *QUIXOTE*, explain the data structure of a legal dictionary, and then describe the use of *QUIXOTE* rules to represent case-based rules and statutes. In *QUIXOTE*, the concepts of situation theory are rephrased as follows:

situation theory	QUIXOTE
situation	module
infor	attribute term
relation name	basic object
type	subsumption
role	label
supporting relation (\models)	membership in module (\in)

3.1 Objects, Modules, and Matching

3.1.1 Basic Objects, Complex Objects, and Modules

The legal dictionary has two parts: the *concept lattice* and the definition of *relations* (viz. the tuples of roled slots with predicates). We first introduce the *QUIXOTE* notion of *objects* and *subsumption* relation for forming the concept lattice, and, thereafter, we describe *QUIXOTE*'s *attribute terms* to represent the relations and infons of situation theory. Object terms, *Obj*, in *QUIXOTE* consist of a set of *basic objects*, *Bobj*, a set of *complex objects*, *Cobj*, and a set of *variables*, *Var*. We denote the *subsumption* relation, \triangleright , as a partial relation between basic objects such that for any $a, b \in \text{Bobj}$, $a \triangleright b$ means that a is more specific than b , or, intuitively, a is $_a$ b . The following is an example of the subsumption relations among basic objects. (In *QUIXOTE* syntax, \triangleright is represented by " \triangleleft ".)

infant \triangleright person, baby \triangleright person,
 person \triangleright creature, lion \triangleright creature,

Together with the basic objects \perp (bottom) and \top (top), we have $\forall x \in \text{Bobj}$, $\perp \triangleright x$, $x \triangleright \top$. Thus, a concept lattice of basic objects, $\langle \text{Bobj}, \triangleright \rangle$, is a finite bounded complete partial order. A *complex object* is of the form $o[l_1 = v_1, l_2 = v_2, \dots]$, where $o \in \text{Bobj}$ and for any $l_i \in \text{Bobj}$, $v_i \in \text{Obj}$, l_i is also called a *label*. The order of labels is not strict, e.g., $o[l=a, m=b]$ and $o[m=b, l=a]$ are treated as being identical objects. The subsumption relation between basic objects is extended to the relation between complex objects, or between complex and basic objects, as follows:

$$\begin{aligned}
 h[l_1 = v_1, \dots] \triangleright h[l'_1 = v'_1, \dots] \text{ iff } h \triangleright h', \forall i \exists j, l_j = l'_i, v_j \triangleright v'_i \\
 h[l_1 = v_1, \dots] \triangleright o \text{ iff } h \triangleright o
 \end{aligned}$$

Similarly, the database operations, *meet* and *join*, between complex objects are defined as the greatest lower bound and least upper bound, since the basic objects compose a complete lattice. For example, the following relation holds when we have *strangle* \triangleright *homicide* and *poison* \triangleright *homicide*.

strangle[agent=tom] \triangleright homicide,
 poison[agent=tom, coagent=mary] \triangleright homicide[coagent=mary]

while there is no subsumption relation between *poison*[agent=tom] and *homicide*[coagent=mary]. For object terms with variables, co-reference relation is considered in the definition as discussed here [17]. For example, $o[l_1 = X, l_2 = X] \triangleright o[l_1 = X, l_2 = Y]$. An *attribute term* is an object term with attached property specifications, i.e., a set of " $l=v$." Such a term for a complex object has the following form.

$$\frac{\text{basic object } o \quad \text{intrinsic } [l_1 = v_1, \dots]}{\text{complex obj. } o} \Big/ \frac{\text{extrinsic } [l_2 = v_2, \dots]}{o}$$

We distinguish the properties of a complex object from those of an attribute term. The former is called an *intrinsic* attribute while the latter is called an *extrinsic* attribute. The label-valued relations of attribute terms are:

$$\begin{aligned}
 o/[l=X] \text{ iff } o \mid \{o.l == X\} \\
 o/[m \rightarrow U] \text{ iff } o \mid \{o.m \triangleright U\} \\
 o/[n \leftarrow V] \text{ iff } o \mid \{o.n \triangleright V\}
 \end{aligned}$$

where $O \mid C$ denotes an object term O with constraint C . We introduce the *dotted term* notation, $o.l$, where o is an object term and l is a label, to specify the value of the l (extrinsic) attribute of o . By default, the properties of an object are inherited from related objects via the subsumption relation, If $o \triangleright p$, then $\forall l, o.l \triangleright p.l$. On the other hand, for complex objects, the values of intrinsic attributes override those of extrinsic ones, e.g., when *death*[cause=suicide] holds, *death*[cause=murder].cause (= murder) is not subsumed by *death*.cause (= suicide) although we have *death*[cause=murder] \triangleright *death*. These attribute terms can be used to represent *SM* infons. Let us consider the following relation (see Section 2.1):

abandon/[agent=Agent, object=Coagent, place=Loc] \\
 | {abandon \triangleright act, Agent \triangleright human, Coagent \triangleright human, \\
 Loc \triangleright location}.

This is a *QUIXOTE* representation of the sentence, "Agent's act of abandoning Coagent at a certain place, where both Agent and Coagent are human." The subsumption relation stated in the constraints denotes the type specification in situation theory, such that the corresponding *SM* representation is:

$$\langle \text{abandon:action}, \dot{x}_{agt}:human, \dot{y}_{obj}:human, \dot{l}_{obj}:location \rangle$$

where *abandon* is of the *action* type, *agt* (agent) and *cgt* (coagent) are of the *human* type. The dictionary maintains legal relations of distinct names, and its object lattice includes the subsumption hierarchy between the relation names.

A *QUIXOTE* legal database consists of a hierarchy of modules. Each module is identified by an object term called a *module identifier* and consists of a set of rules. The rules of one module are inherited by its submodules. The submodule relation, \triangleright_s , is a partial relation between module identifiers that specifies rule inheritance among modules. For example, if *case1* \triangleright_s *case2*, all rules and facts in module *case2* are inherited by module *case1*. (In *QUIXOTE* syntax, \triangleright_s is represented by " \triangleright_s ".) We called *case1* a *submodule* of *case2* and *case2* a *supermodule* of *case1*. Module and rule inheritance are powerful devices for classifying and modeling situation-dependent knowledge. Identical objects must have equal properties within a module, but are allowed to have distinct properties between different unrelated modules. For instance, the following piece of code is consistent, provided that *sit_1* and *sit_2* are not related.

Sit_1 :: homicide/[agent=tom]; sit_2 :: \\
 homicide/[agent=mary];

3.1.2 Realization of Concept Matching

The concept of infon matching, stated in Condition 2.2, is realized in *QUIXOTE* as follows.

OPERATION 3.1 (Infon Matching): For any two attribute terms o_1 and o_2 ,

- 1) o_3 exists, such that $o_1 \diamond o_3$, and $o_2 \diamond o_3$ in a given concept lattice, then o_1 and o_2 are interpreted as being *partially matched infons*, and
- 2) if the basic object parts of two attribute terms are found to be identical, the two attribute terms are interpreted as being *exactly matched infons*.

Under Operation 3.1, for example, *abandon* and *leave* are *partially matched* if the legal dictionary contains: *abandon* \diamond *act*, *leave* \diamond *act*, and *abandon*/[agent=jim] is *exactly matched* with *abandon*/[agent=tom].

OPERATION 3.2 (Situation Matching): For any m_1 and m_2 ,

- 1) if, for every attribute term in m_1 , there is one and only one attribute term in m_2 that can match it exactly, and vice versa, then the two modules are interpreted as being *exactly matched situations*, and
- 2) if, for any o_1 in m_1 whose *relevance* value subsumes a given object (viz. the threshold level), there is an attribute term o_2 in m_2 , that can be partially matched with o_1 , the two modules are interpreted as being *partially matched situations*.

For example, if two modules contain:

```
m_n :: {abandon/[agent=mary, object=june],
        leave/[agent=mary, object=june]};
m_o :: {abandon/[agent=jim, object=tom],
        leave/[agent=jim, object=tom]};
```

QUIXOTE would assert that m_n is *exactly matched* with m_o . Consider another pair of descriptions:

```
m_n :: {abandon/[agent=mary], leave/[agent=mary,
        object=june]};
m_o :: {abandon/[agent=jim, object=tom] |
        {abandon.relevance == 13}, leave/[agent=jim,
        object=tom] | {leave.relevance == 12},
        poor/[agent=jim] | {poor.relevance == 11}};
```

where $11 \leq 12 \leq 13$, we have:

- 1) if the threshold value is 12, then m_n is *partially matched* with m_o , and
- 2) item if the threshold value is 11, then m_n is *not partially matched* with m_o .

3.2 Situated Inference Rules

A *QUIXOTE* rule takes the following form:

$$\overbrace{m_0}^{\text{head}} :: H \mid \overbrace{HC}^{\text{head_constraints}} \Leftarrow \overbrace{m_1, B_1, \dots, m_n, B_n}^{\text{body}} \parallel \overbrace{BC}^{\text{body_constraints}} ;;$$

where H or B_i is a literal while HC and BC are sets of subsumption constraints. An object term, m_i , is called a *module identifier*. The above rule exists in the module m_0 . Intuitively, this means that if every B_i holds in a module m_i under the constraints BC , then H and constraints HC hold in m_0 , where H and B_i s are *object terms* or *attribute terms*. HC works as constraints in the sense of conventional CLP language [8], while BC is processed abductively. Constraints in *QUIXOTE* are sets of formulas in terms of a subsumption relation among object terms and dotted terms. Each formula has the form $\langle term \rangle$, $\langle op \rangle$, $\langle term \rangle$ where $\langle term \rangle$ is an object term or a dotted term and $\langle op \rangle$ is $=$, \diamond , or \blacktriangleright . If the head constraints and module identifiers of a rule can be omitted, the body constraints, BC , of that rule then constitute the background conditions.

3.2.1 Case Representation

We give a sample case below, which is simplified from an actual legal precedent [11].

Mary's Case: On a cold winter's day, Mary abandoned her son Tom on the street because she was very poor. Tom was just four months old. Jim found Tom crying on the street and started to drive Tom by car to the police station. However, Jim caused an accident on the way to the police station. Tom was injured. Jim thought that Tom had died in the accident and left Tom on the street. Tom froze to death.

In *QUIXOTE* format, the aforementioned case contains objects, such as *mary*, *tom*, *jim*, *accident*, and *cold*, as well as several events, such as *abandon*, *find*, *make*, *injure*, *leave*, *death*, and *causes*. The relevance levels of these events are indicated through explicit attributes with ordering values.

```
&subsumption;; l1 =< l2 =< l3;;
&rule;; mary_case :: {mary, tom, jim, accident,
        cold, poor/[agent=mary, relevance=11],
        abandon/[agent=mary, object=tom,
        relevance=12], find/[agent=jim,
        object=tom/[state=crying], relevance=11],
        accident/[agent=jim, relevance=12],
        baby/[agent=tom, age=4months],
        injure/[agent=jim, object=tom, by=accident,
        relevance=12], leave/[agent=jim,
        object=tom, relevance=13],
        death/[agent=tom, cause=cold,
        relevance=13]};
```

There were many interpretations of the responsibilities of the actions of Mary and Jim. A lawyer might reason that: "If Mary hadn't abandoned Tom, Tom wouldn't have died. In addition, the cause of Tom's death was not injury but freezing. Therefore, there exists a causality between Tom's death and Mary's abandonment." Another lawyer would, however, argue differently: "There is a crime committed by Jim, for his abandonment of Tom. And in addition, Tom's death is indirectly caused by Jim's abandonment. Therefore, there exists a causality between Tom's death and Jim's abandonment." These contradictory claims are documented, together with the final verdict as decided by the judge, as a judicial precedent. The next subsection, shows how to model these conflicting arguments using *case rules*.

3.2.2 Case, Induced, and Legal Rules

The deduction of legal arguments in *QUIXOTE* observes the following convention.

$$\overbrace{Head}^{\text{result}} \Leftarrow \overbrace{B_1, B_2, \dots, B_n}^{\text{facts}} \parallel \overbrace{Background_conditions}^{\text{lawyer's interpretation}}$$

Namely, B_i s in the above are the facts that were accepted by both the plaintiff and defendant beforehand, and the set of *Background Conditions* is the interpretation of causal relations between events, scopes of an agent's intention, and so on. For example, we can represent the following case-based rules in Mary's case (see Rule 2.1).

```
c >- cr1;;
cr1 :: responsible_for_injury/[agent=jim,
        to=tom] <= accident/[agent=jim],
        injury/[agent=tom] || {injury.cause=<=
        accident};;
```

c >- $cr1$, claims that $cr1$ is an extended case of c , including the case rule. This rule claims: when there existed *jim's accident* and *tom's injury* as facts, and if the injury's cause was ascribed to the *accident*, *jim* is responsible to *tom* for the injury. Similarly, $cr2$ is another example of a case rule, again from Mary's case.

```
c >- cr2;;
cr2 :: responsible_for_protection_for_weak/
```

```
[agent=jim, to=tom]
<= accident/[agent=jim], baby/[
agent=tom], injury/[agent=tom],
leave/[agent=jim, object=tom]
|| {injury.cause=<= accident};;
```

The idea of an *induced rule* is to abstract some of ground terms in case rules, either by common sense knowledge or by legal theories. For example, if there are several similar accident cases, the attorneys may draw the following generalization, because the causality between the accident and the injury is agreed by both sides (refer to Rule 2.2):

```
ir1 :: responsible_for_injury/[agent=X, to=Y]
<= accident/[agent=X], injury/[agent=Y]
|| {X =<= person, Y =<= person};;
```

In the above rule, restrictions on variables X and Y are given in the background conditions, such that they have to satisfy certain roles. The following ir2 is yet another, more abstract, form of ir1.

```
ir2 :: responsible/[agent=X, to=Y, for=Inj]
<= Acc/[agent=X], Inj/[agent=Y, cause=Acc]
|| {Acc =<= accident, Inj=<=physical_damage,
X =<= person, Y =<= person};;
```

In ir2, traffic accident and injury are abstracted to variable Acc and Inj, and are subsumed by their super concepts in the legal dictionary.

Legal rules are written in a form having free parameters. Consider the following penal code (Japanese Penal Code, Article 199): "In case an intentional action of person A causes the death of person B and the action is not presumed to be legal, A is responsible for the crime of homicide." Its *QUIXOTE* representation is (see Rule 2.3):

```
lr1 :: responsible_for_homicide/[agent=A, to=B]
<= Action/[agent=A], illegal/[act->Action],
death/[agent=B, cause->Action]
|| {Action =<= intend, A =<= person, B =<=
person};;
```

where illegal/[agent=A, action -> Action] claims that the action Action done by A, such as self-defense, is not found to be legal. The statute for the legality of self-defense is described as follows (Japanese Penal Code, Article 38):

```
lr2 :: illegal/[act->Action] <= Action
|| {Action =<= intend};;
```

4 QUERY AND INFERENCE IN *QUIXOTE*

4.1 Constraints and Answer with Assumption

QUIXOTE supports two kinds of constraints: *head constraints* and *body constraints*. During execution, the following transformation is performed first.

$$m_0 :: H \mid HC \Leftarrow m_1 : B_1 \mid C_1, \dots, m_n : B_n \mid C_n \parallel \overline{Dot_Cstr} \cup \overline{Oterm_Cstr};$$

$$m_0 :: H \mid \overbrace{HC \cup \overline{Oterm_Cstr} \Leftarrow m_1 : B_1, \dots, m_n : B_n}^{\text{head constraints}} \parallel \overbrace{\overline{Dot_Cstr} \cup C_1 \cup \dots \cup C_n}^{\text{body constraints}};$$

Constraints (subsumption relation) on object terms (*Oterm_Cstr*) are merged to the head constraint (*HC*), and are used as background conditions for the applicability of the rule, while constraints that includes a dotted term (*Dot_Cstr*) remain as the body constraint, and constraints on each object in the body (*C_i*) are merged into the body constraint. To reply to a query, *QUIXOTE* often returns answer substitutions with a set of constraints among dotted terms called *assumptions*. An assumption is a set of unsatisfied constraints during derivation, such that they can be considered as being missing information. The control program or the user will then decide whether to fill in the missing information, or to invoke another query. Except for constraints among dotted terms, *QUIXOTE* works like a conventional CLP language [8].

However, dotted term constraints in the body constraints work as assumptions if they are not satisfied in the head constraints. In this respect, *QUIXOTE* supports abductive queries to partial information databases, and such partiality differs from *incompleteness* in databases represented as null values or Skolem constants.

The formal derivation in *QUIXOTE* is explained as follows. Let G_m be a set of goals in the m th stage of an execution, the next set of goals is derived from the rule $H \mid HC \Leftarrow B \parallel BC : G_{m+1} = (G_m - \{G\})\theta \cup B\theta$, where there is a most general unifier θ between H and G . Thus, the current goal, $H\theta (= G\theta)$, is removed from G_m , and new goals that are in the body part of the rule $B\theta$ are added. When $G_n = \emptyset$, execution ends. The conclusion is the set of resolved head constraints: $C_{m+1} = (C_m \cup HC)\theta$, while a set of assumptions, or the remaining unsatisfied constraints, A_i , becomes: $A_{n+1} = (A_m \cup BC)\theta - C_{m+1}$. A_i is the accumulation of body constraints $BC\theta$, some of them being removed from this accumulation when they are satisfied in $HC (= C_{m+1})$, and the final set of assumption, A_n , becomes the *abductive reason* for the conclusion. As an example, the following code says that there is a crime and the judgment result is guilty if self-defense is illegal, but innocent if self-defense is legal.

```
case::crime;;
case::judgment[result=guilty]
<= crime/[self_defense->illegal];;
case::judgment[result=innocent]
<= crime/[self_defense->legal];;
```

The first clause tells us of the existence of an object, *crime*, but nothing about the properties of its *self_defense* attribute. The second clause means that if *crime* exists in the case and the *self_defense* property is subsumed by *illegal*, judgment[result=guilty] holds. When we initiate a query $?- \text{case:judgment[result=Result]}$, that is, the judgment result of the crime, *QUIXOTE* returns the following two independent answers.

```
Result=guilty if case:crime.self_defense=<=illegal
Result=innocent if case:crime.self_defense=<=legal
```

Each answer assumes that the *self_defense* property of *crime* coming from the body of the second or third clause. Neither constraint is satisfied by the head constraint, which is empty in this example, so they are accumulated as assumptions.

4.2 Inference of Legal Knowledge

We list a small case example (a traffic accident) and use it to show how the induced rule ir1 is invoked.

```
N_case :: injure/[agent=tom];;
n_case :: traffic_accident/[agent=jim];;
&subsumption;; traffic_accident =<= accident;;
injure =<= physical_damage;; person ==
{jim, tom};;
&submodule;; ir2 -<= n_case;;
```

Now, consider the following query, $?- \text{n_case:responsible/[agent=jim, to=X, for=Y]}$. This query may be read as "Is Jim responsible to someone, X, for something (represented variable Y)?" *QUIXOTE* returns the following answer: IF $\text{n_case:injure.cause} == \text{traffic_accident}$ THEN $Y == \text{injure}$, $X == \text{tom}$. This answer says that if the cause of the injury is the traffic accident in this case, then Jim is responsible. Consider the following case, *hanako_case*, where *QUIXOTE* invokes a sequence of case and legal rules to draw a conclusion, as shown in Fig. 2.

```
Hanako_case :: {hanako, taro, jiro,
death/[agent=taro, age=4months],
baby/[agent=taro age=4months],
injury/[agent=taro], abandon/[agent=hanako,
object=taro],
accident/[agent=jiro], leave/[agent=jiro,
object=taro]};;
```

Using Operation 3.2, *QUIXOTE* would partially match *hanako_case* with *mary_case* (see Section 3.2.1) with the threshold relevance value, 12. That is, there is a rule substitution, θ , on *cr2* (see Rule 2.4): $\theta = [\text{hanako}/\text{mary}, \text{taro}/\text{tom}, \text{jiro}/\text{jim}]$, where 'x/y' stands for a substitution of y for x. *cr2_s*, as generated, is represented as follows:

```
cr2_s :: responsible_for_protection_for_weak/
[agent=jiro, to=taro]
<= accident/[agent=jiro], baby/[agent=taro],
injury/[agent=taro], leave/[agent=jiro],
object=taro] || {injury.cause=< accident};;
```

The concept of *anchoring*, mentioned in Section 2.4, is realized in *QUIXOTE* by invoking either induced case rules or statutes within a case description. Let us suppose the following submodule relation:

```
&submodule;; w >- hanako_case;; w >- lr3;; w >- cr2s;;
```

with the following subsumption relations.

```
&subsumption;; leave =< abandon;; abandon =< intend;;
```

In addition, we need one more rule that is derived from common sense: *weak* =< *baby*;;, then with the query:

```
?-
w:responsible_for_death_by_abandonment_of_weak/
[agent=X, to=taro],
```

meaning that "Is someone responsible for the death of Taro by abandoning the weak person?" *QUIXOTE* returns with two answers as follows.

```
** Answer 1 **
IF w:injury.cause =< accident
w:leave.agent >= responsible
_for_death_by_abandonment_of_weak.agent
w:death.cause =< leave THEN X =< jiro
** Answer 2 **
IF w:injury.cause =< accident
w:abandon.agent >= responsible
_for_death_by_abandonment_of_weak.agent
w:death.cause =< abandon THEN X =< hanako
```

The first answer interprets the causality in Hanako's case as: if the cause of Taro's death is some event under Jiro's leaving Taro, then Jiro is responsible for the homicide. The second answer states yet another interpretation, i.e., Hanako is responsible if Taro is killed by Hanako's intended abandonment. This rather confusing response arises from the fact that there were two deeds, *leave* and *abandon*, both of which can be regarded as being abandonment, i.e., both belong to the same class in the legal dictionary. To verify this, one can further query the database with new constraint:

```
?-w:D || {D =< abandon}.
** Answer 1 **
D == leave
** Answer 2 **
D == abandon
```

Thus, this section has shown that *QUIXOTE*:

- 1) returns answers with assumptions when there are unsatisfied background conditions for applying legal and case rules,
- 2) proposes all the alternative solutions to the query program for unsatisfied background conditions, and
- 3) accepts queries with additional information that has not yet been stored in its databases.

These features confirm the knowledge processing capability of *QUIXOTE* in supporting situated inference within an OODB framework and in managing persistent legal data.

5 CONCLUSIONS

In this paper, we have outlined the motivation behind this study, presented the basic features of a formal model for legal reasoning, and a deductive object-oriented database for implementing this model. The foundation of this model, *SM*, is based on the theory of situations and clearly defines the notions of open-texture concepts and situated inference in the legal domain. The purpose of this model is to study the fundamental issues of AI and Law at the abstraction level, to help design better and more robust legal reasoning systems. In Section 2, we introduced the key features but leave a more detailed description in a future paper. In Sections 3 and 4, we described how *QUIXOTE*, a deductive object-oriented database system, is used to implement *SM* for our legal reasoning applications. In addition, we have illustrated the features of *QUIXOTE* with implemented legal examples. To the best of our knowledge, this is the first reported work that brings together two previously unrelated fields, namely, deductive object-oriented databases and situation theory, to design knowledge systems for solving complex problems and for modeling human intellectual behavior. It is also the first attempt to enhance the reasoning capability and application scale of the current generation of legal reasoning systems with an advanced database tool.

QUIXOTE provides a single language for both query and programming purposes, and it exhibits the inference features of deduction, object-oriented, and constraint logic programming. Most legal reasoning systems are small programs that lack the database management capability to access and store large volumes of data, presenting a stumbling block to the growth of this knowledge-intensive field. The DOOD approach is proposed here to satisfy such needs. In addition, research into legal reasoning systems is closely related to a broader and more complex field, natural language processing (NLP). The ability of DOOD systems, such as *QUIXOTE*, to model abstract concepts of situation theory in a database environment may pave the way for the natural language processing community to tackle concrete, demanding problems, such as building a comprehensive dictionary database of general linguistic concepts.

ACKNOWLEDGMENTS

This work was a part of the Fifth Generation Computer Systems (FGCS) project of Japan. The authors would like to thank the following ICOT members in the knowledge base management group and legal reasoning group: Drs. K. Nitta, K. Yokota, and H. Yasukawa.

REFERENCES

- [1] K. Ashley, *Modeling Legal Argument*. Cambridge, Mass.: MIT Press, 1990.
- [2] J. Barwise and J. Perry, *Situations and Attitudes*. Cambridge, Mass.: MIT Press, 1983.
- [3] J. Barwise, *The Situation in Logic*, CSLI Lecture Notes 17, CSLI, Stanford Univ., 1988.
- [4] S. Ceri, G. Gottlob, and L. Tanca, *Logic Programming and Databases*. Berlin: Springer-Verlag, 1990.
- [5] T. Chikayama, "Operating System PIMOS and Kernel Language KL1," *Proc. Int'l Conf. Fifth Generation Computer Systems*, ICOT, pp. 73-88, Tokyo, June 1992.
- [6] K. Devlin, *Logic and Information I*, Cambridge Univ. Press, 1991.
- [7] A.v.d.L. Gardner, *An Artificial Intelligence Approach to Legal Reasoning*. Cambridge, Mass.: MIT Press, 1987.
- [8] J. Jaffer and J.-L. Lassez, "Constraint Logic Programming," *Proc. Fourth IEEE Symp. Logic Programming*, 1987.
- [9] H. Kamp, "A Theory of Truth and Semantic Representation," *Methods in the Study of Language Representation* J. Groenendijk, T. Jansson, and M. Stockhof, eds. Amsterdam: Math Carter, 1981.
- [10] M. Kifer and G. Lausen, "F-Logic—A Higher Order Language for Reasoning About Objects, Inheritance, and Schema," *Proc. ACM*

- SIGMOD Int'l Conf. Management of Data, pp. 134-146, Portland, June 1989.
- [11] K. Nitta, Y. Ohtake, S. Maeda et al., "HELIC-II: A Legal Reasoning System on the Parallel Inference Machine," *Proc. Int'l Conf. FGCS, ICOT*, pp. 1,115-1,124, Tokyo, June 1992.
- [12] K. Nitta, S. Wong, and Y. Ohtake, "A Computational Model for Trial Reasoning," *Proc. Fourth Int'l Conf. AI and Law*, Amsterdam, June 1993.
- [13] Special issue: AI and Legal Reasoning, Part 1, 2, E.L. Rissland, ed., *Int'l J. Man-Machine Studies*, vol. 34, no. 6, June 1991.
- [14] M. Sergot, "The Representation of Law in Computer Programs: A Survey and Comparison," *Knowledge Based Systems and Legal Applications*, T.J.M. Bench-Capon, ed., pp. 3-68. Academic Press, 1991.
- [15] K. Taki, "Parallel Inference Machine PIM," *Proc. Int'l Conf. FGCS, ICOT*, pp. 50-72, Tokyo, June 1992.
- [16] S. Wong, "A Situation-Theoretic Model for Trial Reasoning," *Proc. Sixth Int'l Symp. Legal Knowledge and Legal Reasoning Systems*, pp. 32-54, Tokyo, Oct. 1992.
- [17] H. Yasukawa and K. Yokota, "Labeled Graphs as Semantics of Objects," *Proc. SIGDBS and SIGAI of Information Processing Society of Japan*, Oct. 1990.
- [18] K. Yokota, H. Tsuda, Y. Morita, S. Tojo, and H. Yasukawa, "Specific Features of a Deductive Object-Oriented Database Language *QUIXOTE*," *Proc. Workshop Combining Declarative and Object-Oriented Databases*, ACM SIGMOD, Washington, D.C., May 29, 1993.

Towards the Correctness and Consistency of Update Semantics in Semantic Database Schema

Joan Peckham, *Member, IEEE*,
Fred Maryanski, *Senior Member, IEEE*,
and Steven A. Demurjian, *Member, IEEE*

Abstract—This paper discusses a paradigm and prototype system for the design-time expression, checking, and automatic implementation of the semantics of database updates. Here, enforcement rules are viewed as the implementation of constraints and are specified, checked for consistency, and then finally mapped to object-oriented code during database design. A classification of enforcement rule types is provided as a basis for these design activities, and the general strategy for specification, analysis, and implementation of these rules within a semantic modeling paradigm is discussed. SORAC (semantic, objects, relationships, and constraints), a prototype database design system at the University of Rhode Island, is also described.

Index Terms—Data modeling, database updates, constraint maintenance, schema checking, data consistency, active databases.

1 INTRODUCTION

THERE are many modern applications that require databases for the storage of persistent, complex, and interrelated data. Semantic relationships among data object types serve as predictors of the paths of query and update throughout database systems. Thus, they must be carefully characterized to provide correct data access and data consistency. For example, CAD (computer aided design) systems need complex structures that are interrelated through built-in relationships, such as IS-A and PART-OF, modeling the structure and behaviors of parts and the roles they play in the total design [3], [19], [23]. Real-time systems [33], [35] need reliable estimates of the time needed for an update and the actions that the update propagates. A careful characterization of the updates that occur over interrelated objects permits analysis of this parameter. In the requirements analysis phase of database security design, [14], [22], a complete description of interrelated data is needed to identify inference dependencies among data items, whereby more sensitive information can be inferred from less sensitive information [29].

This work addresses these needs through an integration of semantic [16], [30] and object-oriented [20], [41], [44] database techniques, and was also inspired by the semantic characteristics of the structural [42] and network [12] models. Although these two models describe a relatively low level of semantics, we have borrowed the philosophy of providing a careful set of relationship structures with clearly specified update semantics, and moved it to a higher conceptual level in the system.

The primary constructs for schema design are object types,

- J. Peckham is with the Department of Computer Science and Statistics, University of Rhode Island, Kingston, RI 02881-0816. E-mail: joan@cs.uri.edu.
- F. Maryanski is with the Provost's Office, University of Connecticut, Storrs, CT 06269-2086. E-mail: fred@chancellor.ypa.uconn.edu.
- S.A. Demurjian is with the Computer Science and Engineering Department, University of Connecticut, Storrs, CT 06269. E-mail: steve@cse.uconn.edu.

Manuscript received Oct. 7, 1993.

For information on obtaining reprints of this article, please send e-mail to: transkde@computer.org, and reference IEEECS Log Number K96041.