Agent-based Structures, Agent Ontology Preservation and Enterprise Modeling

Cyrus. F. Nourani

projectmetaai@cs.com

ABSTRACT

Design techniques with software agents and *Abstract Intelligent Implementations are presented.* Agent morphisms are defined and applied to preservation principles. The object level definitions for individual modules can be automatically programmed by source abstract syntax tree to target abstract syntax tree morphisms. AII techniques are applied to define an Ontology Preservation Principle for Heterogeneous KB Design and implementation.

The stages of conceptualization, design and implementation are defined by AI agents and Mediators. Multiagent implementations are applied to software design techniques, which incorporates object level nondeterministic knowledge learning and knowledge representation developed in [12]. Objects, message passing actions, and implementing agents are defined by syntactic constructs, with agents appearing as functions. By defining specified agent activators events and activity are computed for the AII agents. The proposed AII techniques provide a basis for an approach to automatic implementations from intelligent syntax trees. Interpretability is defined by mediators implementing objects and agents

1. Abstract Modeling with Computing Agents

The paper on which the enclosed abstract is based presents a formal basis to agent ontology modeling and ontology preservation is specific designs. The notion of abstract implementation defined by this author in [1,4] are either algebraic or model-theoretic (algebraic logic) definitions. We refer to specifications of the form <O,A,R>, abbreviating <Objects, Actions, and Relations>, as presentations. A design realization applies a triple I: <I[O],I[A],I[R]> with agents with specific ontologies, where I is an agent ontology preserving mapping. Informally the process of design realization was defined by the author to be that of encoding the algebraic structure of the conceptualization of a problem with agents onto the algebra that specified a specific system. Thus design realization is via morphisms on agent-based algebras. A multiagent System design might be defined as an "enterprise" by <O,A,R> is implemented by agents that characterize the implementation function

I: <O,A,R> $\rightarrow$ <I[O],I[A],I[R]> is to be defining a mapping

I: <Alg[A],Alg[F]> $\rightarrow$ <Alg[I(A)],Alg[I(F)]>

We refer to Alg[A] and Alg [F] are what we call *ontology algebras.* The implementation mapping I defines wrappers to resources in a manner preserving the ontology algebra. Ontology algebras are multi-sorted algebras defining multiagent systems defined by formal agents, e.g. hysteretic or knowledge level agents and agent morphisms. A formal definition is provided. *The Ontology Preservation Principle:* The AII is correct only if it preserves the ontology algebras. It will be abbreviated by AIIOPP. Widerhold's domain knowledge base algebra DKB consists of matching rules linking domain ontology. There are three operations defined for DKB. The operations are Intersection- creating subset ontology and keeping sharable entries. Union- creates a joint ontology merging entries. Difference- creates a distinct ontology and removing shared entries. Mapping functions must be shown to preserve ontology. Applying AIIOPP we can state specific preservation principles as follows. The DKB Preservation Principle- AII implementations must preserve ontology under Intersection, Union, and Difference operations. . The algebras Alg[A] and Alg[F] define wrappers for the mediators as functions for interacting with resources. A wrapper is a tool to access known resources and translate their objects.

1.1 Agents

Starting with hysteretic agents [5], the agent has an internal state set I, which the agent can distinguish its membership. The agent can transit from each internal state to another in a single step. Actions by agents are based on I and board observations. There is an external state set S, modulated to a set T of distinguishable subsets from the observation viewpoint. An agent cannot distinguish states in the same partition defined by a congruence relation. A sensory function s :S $\rightarrow$ T maps each state to the partition it belongs. Let A be a set of actions which can be performed by agents. A function action can be defined to characterize an agent activity action:T $\rightarrow$A.

There is also a memory update function mem: I x T $\rightarrow$ I. To define agent at arbitrary level of activity knowledge level agents are defined. All excess level detail is eliminated. In this abstraction an agent's internal state consists entirely of a database of sentences and the agent's actions are viewed as inferences based on its database. The action function for a knowledge level agent maps a database and a state partition t into the action to be performed by an agent in a state with database and observed state partition t. action: Dx T$\rightarrow$ A

The update function database maps a state and a state partition t into a new internal database.

database: D x T → D

A knowledge-level agent is an environment is an 8-tuple shown below. The set D in the tuple is an arbitrary set of predicate calculus databases, S is a set of external states, T is the set of partitions of S, A is a set of actions, see is a function from S into T, do is a function from A S into S, database is a function from D x T into D, and action is a function from D x T into A. <D,S,T,A,see,do,database,action> Knowledge level agents are hysteretic agents.

1.2AgentMorphisms

Let HA be a set of sextuples defining hysteretic agents. Define HA morphims by a family of functions defined component-wise on the sextuple above.

Definition 1.1 A HA morphism is a function F : HA → HA' defined component-wise by F[i]: I→ I'; F[S]: S → S', F[T]: T →T', F[A]: A →A'; F[s]: S→ T'; F[d]: A' x S' → S' and F[internal]: I' x T'→ I'. □

Definition 1.1 implies F defines new hysteretic agents from HA by a morphism. The definition might become further transparent in view of definitions. Component-wise definitions for morphism might be viewed as functions on a multi-sorted signature carrying the sextuple. Similar morphisms can be defined for knowledge level agents.

1.3 Agents, Languages, and Models

By an intelligent language we intend a language with syntactic constructs that allow function symbols and corresponding objects, such that the function symbols are implemented by computing agents in the sense defined by this author in (Nourani 1993c, 96a). Sentential logic is the standard formal language applied when defining basic models. The language $\mathcal{L}$ is a set of sentence symbol closed by finite application of negation and conjunction to sentence symbols. Once quantifier logical symbols are added to the language, the language of first order logic can bedefined.

A Model $\mathcal{A}$ for is a structure with a set A. There are structures defined for $\mathcal{L}$ such that for each constant symbol in the language there corresponds a constant in A. For each function symbol in the language there is a function defined on A; and for each relation symbol in the language there is a relation defined on A. For the algebraic theories we are defining for intelligent tree computing in the forthcoming sections the language is defined from signatures as in the logical language is the language of many-sorted equational logic.

The signature defines the language by specifying the function symbols' arities. The model is a structure defined on a many-sorted algebra consisting of S-indexed sets for S a set of sorts. By an intelligent language we intend a language with syntactic constructs that allow function symbols and corresponding objects, such that the function symbols are implemented by computing agents. A set of function symbols in the language, referred to by AF, is the set modeled in the computing world by AI Agents with across and/or over board capability. Thus the language defined by the signature has designated function symbols called AF. The AF function symbols define signatures which have specific message paths defined for carrying context around an otherwise context free abstract syntax. A set of function symbols in the language, referred to by AF, are agents with nontrivial capability.

Definition 1.2 We say that a signature is intelligent iff it has intelligent function symbols. We say that a language has intelligent syntax if the syntax is defined on an intelligent signature. □

Definition 1.3 A language L is said to be an intelligent language iff L is defined from an intelligent syntax. □ The above mathematical basis might be applied to the KQML agent language paradigms. However the author has not had the time an occasion to explore the applications. The example of intelligent languages [3] we could present are composed from <O,A,R> triples as control structures. The A's have operations that also consist of agent message passing. The functions in AF are the agent functions capable of message passing. The O refers to the set of objects and R the relations defining the effect of A's on objects. Amongst the functions in AF only some interact by message passing. There is a new frontier for theoretical development of the <O, A, R> algebras and that of the AII foundations. <O,A,R> is a pair of algebras, <Alg[A],Alg[F]>(see section 3), connected by message passing and AII defines techniques for implementing such systems. To define AII we define homorphisms on intelligent signature algebras.

Definition 1.4 An I-homorphism is a homoprphism defined on algebras with intelligent signature I. □

To define agent specific designs we apply HA-morphisms via the following definition.

Definition 1.5 Let A and B be I-algebras with signatures containing an agent signature HA. A HA-homomorphism from A to B is an I-homomorphism with defined HA-morphism properties. □

2. Agent Ontology Preservation Theorems

Let us apply the definition for HA agents and HA morphisms to state a preservation theorem. Let A and B be I-algebras with the signature I containing HA agents. Let Alg[B] be an I-algebra defined from B implementing [1] a specified functionality defined by A. An AII is an implementation for Alg[A] by Alg[B].

Definition 2.6 Let A and B be I-algebras with intelligent signature I containing agents. An I-ontology is an I-algebra with axioms for the agents and functions on the signature. □

Theorem 2.1 Let A and B be I-algebras with the signature I containing HA agents. The AII with HA morphisms

defined from A to B preserve I-ontology algebras iff defined by HA-homorphisms. □

Theorem 2.2 Let A and B be I-algebras with the signature I containing KL agents. The AII with KL morphisms preserve I-ontology algebras iff defined by KL-homomorphisms. □

DKB mappings are specific AII's were the ontology algebra operations are the same at source and target. the DKB mappings are proved AIIOPP consistent.

3. Meditors and Ontologies

A *mediator* is a software module that exploits encoded knowledge about certain sets or subsets of data to create information for a higher layer of applications. and the definition goes on to state `It should be small and simple, so that it can be maintained by one expert or, at most, a small and coherent group of experts' Mediator instantiation is to populate a domain-independent service or tool with domain-specific knowledge. We define *Mediator Specifications* with agent ontologies consisting of a tuple engine agent-based computing system <A,F> := <Design_Agents,CoAgents>, consisting of Design_Agents := <O,A,RNA> and CoAgents;= <O,F,RFA>. The design is depicted by the following figure. RNA are normal actions and RFA the faults, exceptions, and remedial functions.

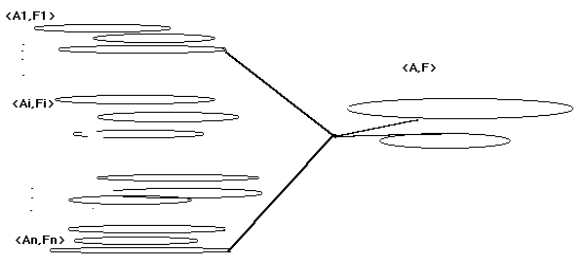

Figure 1-The pairs <Ai,Fi> are modules composed to define <A,F>.

The modules are defined from multiple objects. Actions could be in form of operations or message communication from one object to another. A set of computing agents forms Design_Agents and a dual set forms CoAgents. CoAgents are agents running parallel checking faults and unplanned events presenting alternatives. The algebras Alg[A] and Alg[F] define wrappers for the mediators as functions for interacting with resources. A wrapper is a tool to access known resources and translate their objects. The Design_Agents corresponds to an algebra Alg[A] of Normal Activities and CoAgents to an algebra Alg[F] for unplanned events, computing faults recovery.

References

[1] Abstract Implementation Techniques for A.I. By Computing Agents,: A Conceptual Overview, Technical Report, March 3, 1993, Proc. SERF-93, Orlando, Florida, November 1993. Published by the University of West Florida Software Engineering Research Forum, Melbourne, FL.

[2]G.Wiederhold: "Interoperation, Mediation and Ontologies"; Proc.Int.Symp. on Fifth Generation Comp Systems, ICOT, Tokyo, Japan, Vol.W3, Dec.1994, pages 33-48.

[3] Nourani, C.F.," Intelligent Languages- A Preliminary Syntactic Theory," May 15, 1995, Mathematical Foundations of Computer Science;1998, 23rd International Symposium, MFCS'98, Brno, Czech Republic, August 1998, Jozef Gruska, and Jiri Zlatuskanbsp;(Eds.): Lecture Notes in Computer Science;1450, Springer, 1998, ISBN 3-540-64827-5, 846 pages.

[4] Nourani, C.F." AII and Heterogenous Software Design, May 10, 1995, MAAMAW'97, Eighth European Workshop on MODELLING AUTONOMOUS AGENTS IN A MULTI-AGENT WORLD May 1997, University of Karlskrona/Ronneby, Dept of Computer Science and Business Administration Ronneby, SWEDEN. Specific track on WWW announced papers only.

[5] Genesereth, M.R. and Nilsson, N.J., Logical Foundations of Artificial Intelligence,"Morgan-Kaufmann,1987.

[6] Nourani, C.F.," Slalom Tree Computing," 1994, AI Communications, December 1996, IOS Press, Amsterdam.

[7]ADJ- Goguen, J.A., J.W. Thatcher, and E.G. Wagner "An Initial Algebra Approach To Specification, Correctness and Implementation of Abstract Data Types, in Current Trends in Programming Methodology, Vol IV, 1978, R. Yeh, editor, Prentice-Hall, Englewood-Cliffs, NJ, pages80-149.

AffilationsAcademia USA last appointment UCSB

ScientificURL
http://members.fortunecity.com/crisfn/metaai.html.doc

Fax 415-430-2167 x1342

Telephone 310-754-6000 x3036

.