Intelligent Tree Computing, Decision Trees And Soft OOP Tree Computing

Cyrus F. Nourani

Project METAAI@CompuServe.com
1986,Revised August 1997
USA Academic Address UCSB

Abstract

Intelligent tree computing techniques are defined with applications to decision tree computing defining decisive agent computing. New linguistics abstraction theories are presented. Treating objects as abstract data types and a two level programming approach to OOP allows us to define Pullup abstractions to treat incompatible objects. A basis for a tree computing theory for abstract objects with intelligent languages and intelligent algebraic tree rewriting is presented. The formulation leads to theoretical results that provide the basis for parallel algebraic tree rewrite computing with intelligent trees. Tree completion theorems are presented, and techniques for generating initial intelligent models are developed. We also have soundness and completeness theorems for the algebraic tree computing theory for abstract objects and its preliminary model theory. New frontiers for object level computing with tree rewriting is presented.

Keywords Intelligent Trees, OOP TreeRewriting, Model Theory, Intelligent Syntax Languages and Intelligent Models, Abstract Objects.

Introduction

The term "agent" has been recently applied to refer to AI constructs that enable computation on behalf of an AI activity (GenNils 987,34). It also refers to computations that take place in an autonomous and continuous fashion, while considered a highlevel activity, in the sense that its definition is hardware and software independent, thus implementation independent (Nourani 1994, 1993a). The present paper develops the techniques and theory of computing with trees on signatures that bear agent functions on trees. Our results for computability of initial models by subtree replacement systems (Nourani 1995a, 1992a) are developed further for as a foundation for computing on trees to be applicable to intelligent free tree computing.

Applications, as a theory of computing, to artificial intelligence, and object level programming, are stated in brief. Further research areas are put forth at the concluding section pointing us to new methods and theories of computing. We present the concepts of intelligent syntax, intelligent languages, and their applications to AI and programming. Algebraic tree rewriting is defined on intelligent trees by presenting the concepts and definition of algebraic tree information content and mutual tree information content within a forest.

At the forest suddenly an OOPI tree rewrite theorem presents itself, defining a correspondence between intelligent tree rewriting and tree intelligence preservation. Next, tree rewriting with intelligent trees is formalized by defining canonical intelligent initial models and results that intelligent algebraic tree rewriting leads to intelligent initial models.

Thus models for intelligent theories emerge from the algebraic intelligent tree rewriting. Intelligent algebraic tree completion theorems and initial model rewrite theorems are put forth for intelligent trees. To bring the techniques to a climax a soundness and completeness theorem is proved for intelligent tree rewriting as a formal algebraic and model-theoretic computing technique. The abstract data type community might find a recent manuscript (Heinrich 1993) on nondeterministic data types relevant to the present paper. However, it has missed this author's tree model theory papers since 1979 and the current intelligent OOP tree computing project since 1992-93. The important direction for soft computing is its application to decision analysis and decision systems. The applications of soft computing to management decision making are equally significant. There are real problems in decision analysis that are faced by organizations today for which a sound computing basis with A.I. and software is presented, identifying new roles for soft computing with agents, decision trees and objects. The methods and tools allow us to deal with high levels of complexity, organization and manipulation of information, applying tree agents and intelligent object modules for decision trees. Lifting the limitations in the application of formal methods by inventing intelligent tree computing. Challenges in decision systems such as incorporation of learning, dynamic knowledge bases, are met with hybrid intelligent computing trees.

Computing On Trees

Recent views

In order to present some motivation for the methods proposed certain model-theoretic concepts are reviewed and some new techniques are presented. The Henkin style proof for Godel's completeness theorem is implemented by defining a model directly from the syntax of theories(Henkin 1949). A model is defined by putting terms that are provably equal into equivalence classes, then defining a free structure on the equivalence classes. The computing enterprise requires more general techniques of model construction and extension, since it has to accommodate dynamically changing world descriptions and theories. The models to be defined are for complex computing phenomena, for which we define generalized diagrams. The techniques in (Nourani 1987,1991a,1995b) for model building as applied to the problem of AI reasoning allows us to build and extend models through diagrams. We had invented G-diagrams(Nourani 19 87,1991a,1995b) to build models with a minimal family of generalized Skolem functions. The minimal set of function symbols is the set with which a model can be inductively defined. The models defined are Initial and computable (Nourani 1995a,1992a). The G-diagram methods allowed us to formulate AI world descriptions, theories, and models in a minimal computable manner. Thus models and proofs equational theories of computing problems are presented by a pair (Σ,E), where Σ is a signature (of many sorts, for a sort set S) (ADJ 1973,1975) and E a set of Σ-equations. Let T<Σ> be the free tree word algebra of signature Σ. From (ADJ 1973,1987) given the set S of sorts, an S sorted signature or operator domain Σ is a family Σw,s of sets, for w in S* (the set of all finite strings from S, including the empty string λ, and s in S. Let Σ be an S-sorted signature. Then a Σ-algebra A is a family As of sets s in S together with a function

σ<A> As1 x... x Asn → As

for each σε Σw,s with w=s1 s2...sn; for σ ε Σλ,s we have σA ε As (also written (σ:→ As. Σλ,s is the set of (names of) constants of sort s. Call σε Σw,s an operator symbol of arity w,s; of rank w; and of sort s.

Let Σ denote the set of all operator symbols in the S-sorted signature. Now let T<Σs>, for s in S be the smallest family of sets contained in ΣU{'(,)}* satisfy the following two conditions, where {'(,)'} is a two element set disjoint from Σ to take on formal parentheses:

(0) Σλ,s ⊆ T<Σsi>

(1) if Σ in Σw,s, w=s1...sn and ti ε T<Σ,si>, then ' σ (t1...tn)' ε T<Σs>.

The family is then made into a Σ-algebra by defining the operations:

(0) for σ εΣλ,s , σ T =σ εT<Σ,s>;

(1) for σεΣw,s, w=s2....sn and ti in T<Σ,si>, σT(t1,...,tn) = σ'(t1...tn)' is in T<Σ,s>.

The quotient of T<Σ>, the word algebra of signature Σ, with respect to the Σ congruence relation generated by E, will be denoted by T<Σ,E>, or T<P> for presentation P. T<P> is the "initial" model of the presentation P. The Σ-congruence relation will be denoted by ≡P. One representation of T(P) which is nice in practice consists of an algebra of the canonical representations of the congruence classes. This is a special case of generalized standard models defined here.

In what follows g t1...tn denotes the formal string obtained by applying the operation symbol g in Σ to an n-tuple t of arity corresponding to the signature of g. Furthermore, gC denotes the function corresponding to the symbol g in the algebra C. We present some definitions from our papers (Nourani 1995a,1992a,1994c) that allow us to define standard models of theories that are ΣCTA's. The standard models are significant for tree computational theories that we had presented in (Nourani 1995a,1992a) and the intelligent tree computation theories developed by the present paper.

for AI and computing problems can be characterized by models computable by a set of functions. This allows us to program with objects and functions "running" on G-diagrams. To allude to our AI planning (Fiks-Nils 1971) as an example, the planning process at each stage can make use of GF-diagrams(Nourani 1995b,1995c), G-diagrams with free Skolemized trees(Nourani-Hoppe 1994), by taking the free interpretation, as tree-rewrite computations for the possible proof trees that correspond to each goal satisfiability This paper is towards a theory of computing and programming that could provide a foundation for computing inevitable with the current and forthcoming programming techniques.

Algebraic Tree Computation

The computing and reasoning enterprise require more general techniques of model construction and extension, since it has to accommodate dynamically changing world descriptions and theories. The techniques in (Nourani 1991 a,1995b) for model building as applied to the problem of AI reasoning allows us to build and extend models through diagrams. This requires us to focus attention on the notion of generalized diagram. A technical example of algebraic models defined from syntax had appeared in defining initial algebras for equational theories of data types (ADJ 1973,1975) and our research in (Nourani 1992a,Nourani 1994). In such direction for computing models of

Definition 2.2 Let M be a set and F a finite family of functions on M. We say that (F,M) is a monomorphic pair, provided for every f in F, f is injective, and the sets {Image(f):f in F} and M are pairwise disjoint.

This definition is basic in defining induction for abstract recursion-theoretic hierarchies and inductive definitions. We had put forth variants of it with axiomatizations in (16). What is new in our papers is the definition of generalized standard models with monomorphic pairs. This definition was put forth by the present author around 1982 (Nourani 1994c) for the computability problems of initial models.

Definition 2.3 A standard model M, with base M and functionality F, is a structure inductively defined by <F,M> provided the <F,M> forms a monomorphic pair.

We will review these definitions in the sections to follow.

G-diagrams for Initial Models

The generalized diagram (Gdiagram) (Nourani 1991a,1995b) is a diagram in which the elements of the structure are all represented by a minimal set of function symbols and constants, such that it is sufficient to define the truth of formulas only for the terms generated by the minimal set of functions and constant symbols. Such assignment implicitly defines the diagram. This allows us to define a canonical model of a theory in terms of a minimal family of function symbols.

The minimal set of functions that define a G-diagram are those with which a standard model could be defined by a monomorphic pair. Formal definition of diagrams are stated here, generalized to G-diagrams, and applied in the sections to follow.

Definition 2.5 Let M be a structure for a language L, call a subset X of M a generating set for M if no proper substructure of M contains X, i.e. if M is the closure of X U {c(M): c is a constant symbol of L}. An assignment of constants to M is a pair <A,G>, where A is an infinite set of constant symbols in L and G: A→ M, such that {G(a): a in A} is a set of generators for M. Interpreting a by g(a), every element of M is denoted by at least one closed term of L(A). For a fixed assignment <A,G> of constants to M, the diagram of M, D<A,G>(M) is the set of basic (atomic and negated atomic) sentences of L(A) true in M. (Note that L(A) is L enriched with set A of constant symbols.)

Definition 2.6 A G-diagram for a structure M is a diagram D<A,G>, such that the G in definition 2.5 has a proper definition by a known function set.

Remark: The minimal set of functions above are those by which a standard model could be defined by a monomorphic pair for the structure M.

Thus initial models could be characterized by their G-diagrams. Further practical and the theoretical characterization of models by their G-diagrams are presented by this author in (Nourani 1995b). This builds the basis for some forthcoming formulations that follow, and the tree computation theories that we had put forth in (Nourani 1992a, 1994). Methods of constructing initial models by algebraic tree rewriting for the intelligent languages is to be developed from our approach in (Nourani 1995a,1985a,1995d). We showed how initial algebras can be defined by subtree replacement and tree rewriting (Nourani 1995a,1994). Our papers in 1979 pointed out the importance of the constructor signatures in computational characterization of initial models, and sufficient conditions for constructibility. These are the minimal set of functions which by forming a monomorphic pair with the base set, bring forth an initial model by forming the free trees that define it.Thus an initial free model is formed. The models might be obtained by applying algebraic subtree replacement systems. The G-diagram for the model is also defined from the same free trees. The conditions of the theorems are what one expects them to be: the canonical subset are to be closed under constructor operations, and the operations outside the constructor signature on canonical terms are to yield canonical terms.

Intelligent Syntax Languages

Intelligent Syntax

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 1993a). A set of function symbols in the language, referred to by Agent Function Set, are function symbols that are modeled in the computing world by AI Agents. The objects, message passing actions, and implementing agents are defined by syntactic constructs, with agents appearing as functions. The computation is expressed by an abstract language that is capable of specifying modules, agents, and their communications. We have to put this together with syntactic constructs that run on the tree computing theories to be presented by this paper. We have to define how the syntactic trees involving functions from the AFS are to be represented by algebraic tree rewriting on trees. This is the subject treated by the next section where we define free intelligent trees. The abstract syntax put forth for to be implemented are expected at two levels. Level 1 is a language that only expresses the functionality of modules by names of objects and their message passing actions, for example by SLPX(2). Level 2 defines the functions themselves. The implementing agents, their corresponding objects and their message passing actions can also be presented by the two-level abstract syntax. From the practical stand point, the models as individual programs can be specified with well-known specifications languages.

An important technical point is that the agents are represented by function names that appear on the free syntax trees of implementing trees. This formulation will prove to be an important technical progress. The trees defined by the present approach have function names corresponding to computing agents. The computing agent functions have a specified module defining their functionality. These definitions are applied at syntax tree implementation time.

Definition 3.2 We say that a signature Σ is intelligent iff it has intelligent function symbols. We say that a language has intelligent syntax iff the syntax is defined on an intelligent signature.

Definition 3.3 A language L is said to be an intelligent language iff L is defined from an intelligent syntax.

Abstract Syntax Information

It is essential to the formulation of computations on intelligent trees and the notion of congruence that we define tree information content of some sort. A reason is that there could be loss of tree information content when tree rewriting because not all intelligent functions are required to be mutually informable. Theories are presented by axioms that define them and it is difficult to keep track of what equations not to apply when proving properties. Further, it is necessary for limiting the computational complexity of the algebraic tree rewriting to at most the R.E. sets, to allow intelligent functions only at some predefined root nodes of trees, for example. We have to take our leave for a vague statement at the present time and refer the reader to our forthcoming papers for its theoretical reasons. We already have presented the mathematics that allows us to order trees by abstract model theory and set theory for tree computations (Nourani 1995a,1985a,Heinrich 1993) What we have to define, however, is some computational formulation of information content such that it applies to the present computability on trees. Once that formulation is presented, we could start decorating the trees with it and define computation on intelligent trees.

It would be nice to view the problem form the stand point of an example. The example of intelligent languages we could present have <O,A,R> triples as control structures. The A's have operations that also consist of agent message passing. The functions in AFS 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 AFS only some interact by message passing. What's worse the functions could affect objects in ways that affect the information content of a tree. There you are: the tree congruence definition thus is more complex for intelligent languages than those of ordinary syntax trees. Let us define tree information content for the present formulation.

Definition 3.3 We say that a function f is an information string , iff there is no message passing or information exchange except onto the object that is at the range set for f, reading parameters visible at each object. Otherwise, f is said to be an information splurge. We refer to them by string and splurge functions when there is no ambiguity. Remark: Nullary functions are information strings. []

Definition 3.4 The tree intelligence degree, TID, is defined by induction on tree structures:
(o)the information content of a constant symbol function f is f;
(i)for a string function f, and tree f(t1,...,tn) the TID is defined by U TID (ti::f), where (ti::f) refers to a subtree of ti visible to f;
(ii) for a splurge function f, TID is defined by U TID (f:ti), where f:ti refers to the tree resulting from ti upon information exchange by f.

The methods of object level programming are implicit in the above definition. For example, the concept of a subtree being visible to a function refers to encapsulation methods. The theorem below formalizes these points, while we defer the reader to some subsequent sections for methods of dealing with this problem. Thus out of the forest of intelligent trees, not to overstate the significance, there appears a sudden information theoretic rewrite theorem.

Theorem 3.5 Trees on intelligent syntax, rewritten guided only by what equations state, could cause a loss of information content between trees.

Proof Trees with AFS functions by definition affect TID, thus a rewrite from a tree formed by a function g in AFS to a tree that does not have g as a function symbol causes an information loss. For example, a harmless looking equation of the form f1 (f(t)) = t, where f is in AFS causes an information loss to the resulting set of trees, from the left hand to the right hand tree t.[] Thus computing with arbitrary rewriting is irreversible and at times nonterminating. And we our facing a computing wilderness at the present time. Let us now define computing with intelligent equational theories.

Definition 3.6 We say that an equational theory T of signature IΣ is an intelligent IΣ theory iff for every proof step involving tree rewriting, the TID is preserved. We state T<IST> I- t=t' when T is an IΣ theory.

Definition 3.7 We say that an equational theory T is intelligent, iff T has an intelligent signature IΣ,and axioms E, with IΣ its intelligent signature.

A proof of t=t' in an intelligent equational theory T is a finite sequence b of IΣ equations ending in t=t' such that if q=q' is in b, then either q=q' in E, or q=q' is derived from 0 or more previous equations in E by one application of the rules of inference. Write T <IST>I- t=t' for "T p proves t=t' by intelligent algebraic subtree replacement system. ".

By definition of such theories proofs only allow tree rewrites that preserve TID across a rule. These definitions now may be applied to prove tree rewriting theorems, set up the foundations for what defines intelligent tree rewriting TID and intelligent tree computation. Thus the essence of intelligent trees will not be lost while rewriting.

Next, we define a computing agent function's intelligence content from the above definition. This is not as easy as it seems and it is a matter of the model of computation applied rather than a definition inherent to intelligent syntax. Let me make it a function of intelligent syntax only, because we are to stay with abstract model theory and build models from abstract syntax. The definition depends on the properties of intelligent trees, to be defined in the following section.

Intelligent Trees

Embedding Intelligence

Viewing the methods of computation on trees presented in the sections above we define intelligent trees here.

Definition 4.1 A tree defined from an arbitrary signature Σ is intelligent iff there is at least one function symbol g in Σ such that g is a member of the set of intelligent functions AFS, and g is a function symbol that appears on the tree. []

Definition 4.2 We define an intelligent Σequation, abbreviate by IΣequation, to be a Σ-equation on intelligent Σterms. An IΣ congruence is an Σ-congruence with the following conditions:
(i) the congruence preserves IΣ equations;
(ii)the congruence preserves computing agents information content of Σ-trees []

Definition 4.3 Let Σ be an intelligent signature. Then a canonical term IΣ-algebra (IΣ-CTA) is a Σ-algebra C such that
(1)ICI is a subset of T<Σ> as S-indexed families
(2) gt1 ...tn in C implies ti's are in C and; gC (tl ,...,tn) = gtl ...tn, where gC refer to the operation in algebra C corresponding to the function symbol g.

For constant symbols (2) must hold as well, with gC = g.
(3) gt1...tn in T<AFS> implies ti's in T<AFS> and gC(t1,...,tn)=gt1...tn; for constant symbols it must hold as gC=g.

Theorem 4.4 Let C be an IΣ-algebra. Let P = (Σ,E) be a presentation. Then C is Σ-isomorphic to T<P>, iff
(i)C satisfies E;
(ii)gC(t1,...,tn)≡P g.t1...tn
(iii) gC(t1,...,tn) ≡ P gt1...tn, with gt1...tn in T<AFS> whenever ti's are in T<AFS> and gC is in AFS.
Note: (ii and iii) must also hold for constants with gC = g; ≡refers to the IΣ-congruence generated by E.

Intelligent Rewrite Models

Term rewrite model theorems for intelligent syntax

Lemma 4.5 Let R be a set of IΣ-equations. Let R be the set of algebraic IΣ-rewrite rules obtained by considering each equation l =r in R as a rule l => r, then for t,t' in T<Σ>, t => * t' iff T(R) <IST>I- t = t'.

Recall that a presentation (Σ,E) defined an equational theory of signature Σ and axioms E. Next we show how canonical models can be constructed by algebraic subtree replacement system. A definition and what we have done thus far (Nourani 1995a) gets us to where we want to go: the canonical algebraic intelligent term rewriting theorems Σ <s1.,,,sn,s> denotes the part of the signature with operations of arity (s1,...,sn) and coarity s, with Csi the carrier of algebra C of sort si. FTP refers to finite termination property and UTP to the unique termination property of tree rewriting.

Definition 4.6 Let R be a convergent set of Σ-rewrite rules, i.e. T <Σ,R> h as the FTP and UTP properties, let (t) denote the R-reduced form of t in T<Σ>. Let ICI be a subset of IT<Σ>I, for g in Σ<sl...sn,s> and ti in C si, define gC (t1,...,tn) = (g(t1,...,tn)). If this always lies in C, then C becomes a IΣ-algebra, and we say that (C,R) represents a IΣ-algebra A iff the IΣ-algebra so defined by (C,R) is IΣisomorphic to A.

The following intermediate theorem gives sufficient conditions for constructibility of an initial model for an IΣ equational presentation. It is the mathematical justification for the proposition that initial models with intelligent signature can be automatically implemented (constructed) by algebraic subtree replacement systems. The normal forms are defined by a minimal set of functions that are Skolem functions or type constructors. Thus we have the following Canonical Intelligent Model Theorems. The theorems provide conditions for automatic implementation by intelligent tree rewriting to initial models for programming with objects.

Theorem 4.7 Let Σ be an S-sorted signature, R a convergent set of Σ-rewrite rules. Let |C| be a subset of |T<Σ>|. Define gC(t1,...,tn) = (g(t1,...,tn)). Furthermore, assume that (f) = f for all f in Σ (I,s) If there exists a subset CF of Σ such that Σ such that |C| = |T<CF>| and the following conditions are satisfied for g with nontrivial arity (s1,...,sn):

1. gC(t1,...,tn) in C whenever ti in C, for ti of sort si;

2. for all g, ti in C, and g in CF, gC(t1,...,tn) = gt1,...tn; in particular for a constant g, gC =g;

3. for g in Σ - CF, gC(t1,...,tn)=t, for some t in T<CF>;

4. for g in AFS, gC(t1 ,...,tn) = t for some t in T<CF ∩AFS>

Then: (i) C is a canonical term IΣ-algebra; and (ii) (C,R) represents T <Σ,R>, R is R viewed as a set of IΣ equations.

Proof First prove by induction on complexity of terms that (C,R) defines a Σ-algebra structure on C. The basis is given by the assumption that constant symbols are trivially R-reduced, because we define σ (C). = (σ) in C, for each σ of O arity. Now define σ<C> (t1,...,tn) = (σ(t1,...tn)), where (t) denotes the R-reduced form of T. It is readily seen from the definitions that (C,R) defines a Σ-algebra. Now since each t in C is R-reduced by (1) and (2) we have σ <C>(t1,...,tn) = σ(t1,...,tn), where σ is in CF. We have already seen that for constant symbols σ, σ <C> = σ. By (3) and (4) the trees formed with function symbols apart form CF are reduced to CF terms that are AFS preserving. Thus (C,R) defines a canonical term IΣ-algebra C. This gives us (i). To show that C is isomorphic to T<Σ,$\underline{R}$> we apply a the CTA representation theorem for IΣ presentations. C satisfies $\underline{R}$ because all t in C are R-reduced, and the Σ-algebra structure on C is defined by R. Furthermore, σ<C>(t1,...,tn) <IΣ>≡ (σ(t1,...,tn)) - because σ<C>(t1,...,tn) = (σ(t1,...,tn)) and since C is a CTA Σ--algebra σ(t1,...,tn) = σt1...tn, therefore, σ<C> (t1,...,tn) = (σ t1...tn), while preserving AFS terms. Thus we have the conditions for CTA representation. By theorem 4.4, C is isomorphic to T<Σ,$\underline{R}$>.

Theorem 4.8 Let Σ be an S sorted signature, and R a convergent set of rewrite rules such that (g) = g. Define a Σ algebra structure C on T<Σ> by σ(t1,...,tn) = (g(t1,...,tn)). Let C* be the smallest sub IΣ-algebra of C .
Then C is a canonical term algebra consisting of R normal forms and (C,R) represents T <E,R>.
Proof (Similar to theorem 4.8 s proof)

Thus an initial free model with signature IΣ is formed. The model can be implemented by algebraic subtree replacement systems.

Definition 4.9 OOPI A function symbol f is **OOP intelligent** iff f is in AFS and it is decidable whether f is a string or splurge function.

Definition 4.10 The mutual intelligence content, *MIC*, of an OOPI is determined by the IΣ -congruence on T<AFS> relating the functions in AFS. It is union of the TID over the trees that are a member of the congruence class of the free T<AFS> with respect to the IΣ-congruence defined on the T<Σ,w>, where w is the arity of f. We have defined tree congruence and canonical intelligent models for this formulation in (Nourani 1995f).

Theorem 4.11 (The MIC Theorem) Let P be a presentation with intelligent signature IΣ for a computing theory T with intelligent syntax trees. Then T is

(a) A Sound logical theory iff every axiom or proof rule in T is TID preserving;

(b) A Complete logical theory iff there is a monomorphic pair defining a canonical set C and a G-diagram, such that C with R represents T<IΣ,R>, where R is the set R of axioms for P viewed as IΣ-rewrite rules.

Proof By Definition of MIC, theorems 4.8-4.10, completeness theorems for the first order logic, and completeness of induction for algebraic structures (Nourani 1994c).

This was referred to as the logical foundations MIC theorem for intelligent syntax algebraic theories from (Nourani 1995f) and is stated here for the theoretically interested reader. Thus the present computing methods have their own well defined logics.

This should be referred to as the logical foundations MIC theorem for intelligent syntax algebraic theories and OOPI. At some further forthcoming research we might take a brief walk in the consequent fields and present new areas of research for computing. There are further MIC theorems that are relevant to computing and the model theory of computing with intelligent trees. Some of these results appear in a paper entitled Double Vision Computing(1995e).

Computing On Intelligent Trees

Tree Computing For AI

We present a brief overview of the applications of our methods to AI planning problems (Nourani 1991a,1995b), as yet another exciting research area. In our methods for planning, the planning process at each stage can make use of GF-diagram by taking the free interpretation, as tree-rewrite computations of the possible proof trees that correspond to each goal satisfiability (Nourani 1991 a,1995b). The techniques can be applied to implement planning and reasoning for AI applications. While planning with GF-diagrams that part of the plan that involves free Skolemized trees is carried along with the proof tree for a plan goal. The idea is that if the free proof tree is constructed then the plan has a model in which the goals are satisfied. The model is the initial model of the AI world for which the free Skolemized trees were constructed (Nourani 1995a,b,c). How are these applications affected by our intelligent language formulation is to be addressed by our forthcoming papers.

Object Abstraction

To put together programming paradigms with objects it is often necessary to "pull up" from the two differing objects to a higher order abstract data type, to be able to achieve compatible syntax and functionality. This linguistic construct is defined by the present version of SLPX. The operation can be defined in terms of two parameters, object 1, and object2. What pullup does is to overload object1 and object2 to abstract data types with compatible functionality. The pullup linguistic construct for object level programming was put forth for the first time by this author around 1993. Pullup is a structural lift and might be viewed as the structural counterpart to overloading. A preliminary view is presented by the IOOP abstract in (Nourani 1995b).

Multiagent Tree Computing

The present formulation of computing has obvious applications to multi agent Al computing. The actual applications are to be the subject of forthcoming papers. Here we state the applicability in a paragraph or two only. In (Nourani 1993a,1993b) we have made use of AI languages that define functionality of the various parts of an AI design by triples <O,A,R> consisting of objects, actions and relations. Actions are operations or processes. The views of abstraction (Nourani 1994), object-level programming, and agent view of AI computation are the important components of inter-play in the approach to multi agent design and implementation. The intelligent syntax in that application areas expresses computing agents and their message passing actions. As an example let us review the following example with objects from (Nourani 1993a).

Object:= Spectacular_Coffee
OPS:= Serve_Coffee (Type,Table_no) I ...... Serve_Coffee (Spectacular_Brew,n) =>
Signal an available robot to fetch and serve (Spectacular_Brew,table n)

Exp:= Serve_Coffee (Angelika,Table no)
I...
Serve_coffee(Angelika,Table_no)=>if out_of_Angelika notify Table_no;
offer today's-brew <and make use of intelligent decision procedures to offer ternatives>

In the above example OPS denotes operations, EXP denotes exceptions, and the last equation defines the exception action. In this example there is a process(action) that is always checking the supply of Angelika coffee implementing the exception function. As another example, while planning for space exploration, an agent might be assigned by the onboard computer system to compute the next docking time and location, with a known orbiting space craft. Programming systems designed with multi agent techniques have an external behavior that is a function of the degree of message passing actions and parallelism conceptualized.

Single Agent Decisions

The present computational model for multi-agent system provides the following basis for single agent decisions. For each agent function there is a MIC determining its mutual information content with respect to the decision trees connected to it. A single agent makes its decisions for each operation or action by computing a plausible next move set. The plausible next move set might have dynamic properties. For example, it might consist of a set of trees bearing agent functions, which compute their next move sets to update the computing trees MIC. We envision that intelligent syntax and computing on intelligent trees is to become a promising method of computing and in some sense inevitable programming paradigms for AI over the next decade. The interplay of our model theoretic method for computational linguistics with the method of intelligent syntax put forth here and our intelligent syntax computability techniques (Nourani-Hoppe 1994) could provide us with exciting expressive languages and techniques or programming trees, by designing programs for single agent functions. Thus the intelligent tree syntax computing and its model theory as presented forms a formal basis for single agent decisions and decisive agent computation. This is an area for computing that is only emerging from the present paper and our methods of abstract intelligent implementations (Nourani 1993b). There are many exciting projects that could be put forth based on these ideas. The computational method that we have been referring to by Double Vision Computing (Nournai 1995e) is one of the consequences of the present approach. The author envisions that the method of intelligent syntax and computing on intelligent trees is to become a promising method of computing and in some sense inevitable programming paradigms for AI over the next decade. The interplay of our model theoretic method for computational linguistics (Nourani 1995f) with the method of intelligent syntax put forth here and the computability techniques could provide us with exciting expressive languages and techniques. The applications to new chess paradigms were reported in (Nourani 1997).

Concluding Comments

What is accomplished thus far: the basis for a theory of computing with intelligent languages and, intelligent OOP tree rewriting, with various completeness theorems for computing on intelligent trees was developed. We have presented intelligent syntax, intelligent trees and computing with intelligent tree rewriting. Last, but not the least, we have soundness and completeness theorems for the algebraic theory of intelligent trees. We have applied the formulation to computing with intelligent functions alluding to forthcoming application areas in AI. The present method of computing will also lead to several new directions for computing of their own rights. Applications to DAI appears in (Nourani 1994).

The method of computing with intelligent trees applied to object level programming, is another. A further area for research opened up by the present paper is Intelligent Algebraic Slalom Tree Computing (Nourani 1994b) and Artificial Algebras.

References

1. Nourani 1995a, Nourani, C.F., "How Could There Be Models For Nothing and Proofs for Free," March 25, 1993, Brief Overview, IDPL95, *Domains, Logic and Programming*, Darmstadt, Germany, May 1995.
2. Nourani 1992a, Nourani,C.F. *"Parallel Module Specification on SLPX,"* November 1990,SIGPLAN, January 1992.
3. Nourani 1987, Nourani,C.F. "Diagrams, Possible Worlds, and the Problem of Reasoning in Artificial Intelligence," *Logic Colloquium*, Padova, Italy, 1987, Proc. in Journal of Symbolic Logic.
4. Nourani 1991a,Nourani,C.F., "Planning and Plausible Reasoning in AI," *Proc. Scandinavian Conference in AI*, May 1991, Denmark, 150-157, IOS Press.
5. Fiks-Nils 1971,Fikes, R.E. and N.J. Nilsson, "Strips: A New Approach to the Application of Theorem Proving to Problem Solving," *AI 2*, 1971, pp. 189-208.
6. Nourani 1993a, Nourani,C.F., "Abstract Implementations Techniques For A.I. Systems By Computing Agents- A Conceptual Overview," February 11, 1993, *Proc. SERF-93*, Orland, FL. November 1993.
7. Nourani 1995b,Nourani,C.F."The IOOP Project Intelligent and Multi Agent Object Level Computing- *The Preliminary Overview*, May 1994, SIGPLAN, February 1995.
8. Nourani 1995c,Nourani,C.F.,"Free Proof Trees and Model-theoretic Planning", February 23, 1995, *Automated Reasoning AISB*, England, April 1995.
9. Nourani 1983, Nourani ,C. F "Equational Intensity, Initial Models, and AI Reasoning, Technical Report, 1983, : A: Conceptual Overview, in *Proc. Sixth European Conference in Artificial Intelligence*, Pisa, Italy, September 1984, North-Holland .
10. ADJ 1973, Goguen, J. A., J.W. Thatcher, E.G. Wagner and J.B. Wright," A Junction Between Computer Science and Category Theory," (parts I and II),*IBM* T.J. Watson Research Center, Yorktown Heights, N.Y. Research Report, RC4526,1973.
11. Nourani 1994a, Nourani, C.F. "A Theory For Programming With Intelligent Syntax Trees and Intelligent Decisive Agents," *Proc. 11th ECAI, Workshop on DAI Applications to A.I. Systems*, August 1994, Amsterdam.
12. ADJ 1987, Thatcher, J.W., E G. Wagner, and J.B. Wright, "Data Type Specification: Parameterization and the Power of Specification Techniques, "Proc. *10th Ann. Symposium on Theory of Computing (STOC)*, San Diego,CA. 1978, ACM, New York, 119-132..
13. ADJ 1975,Goguen, J.A. J.W Thatcher, E.G. Wagner, and J.B. Wright, "An Introduction to Categories, Algebraic Theories and Algebras,* *IBM Research Report*, RC5369, Yorktown Heights, N.Y., April 1975
14. Nourani 1985a, Nourani, C.F., "The Connection Between Positive Forcing and Tree Rewriting," *Proc. Logics In Computer Science Conference (LICS) and ASL*, Stanford University, July 1985, Proc. Journal of Symbolic Logic.
15. Nourani 1995d"AII and Heterogenous Design," *1995, MAAMAW*, Stokholm April 1997.
16. Nourani 1994b, Nourani,C.F."Slalom Tree Computing- A Tree Computing Theory For Artificial Intelligence," Revised December 1994, *AI Communications*, December 1996, IOS Press, Amsterdam.
17. Nourani 1993a, Nourani,C.F.,"A Multi_Agent Approach To Fault-Free and Fault Tolerant AI," *Proc. FLAIRS,93, Sixth Florida AI Research Symposium*, April 1993.
18. Nourani-Hoppe 1994, Nourani,C.F. and T. Hoppe,"G-diagram for Models and Free Proof Trees," *Berlin Logic Colloquium*, May 1994
19. Gen_Nils 1987, Genesereth,M.R and N.J. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan-Kaufmann, 1987
20. Henkin 1949, Henkin, L.,"The Completeness of First Order Functional Calculus," *Journal of Symbolic Logic*," vol. 14, 1949.

21. Heinrich 1993, Heinrich, H. "*Nondeterminism is Algebraic Specifications and Algebraic Programs*," Birkhauser 1993.
22. Nourani 1995e,Nourani, C.F., "Double Vision Computings' *December 1993, IAS-4*, Karlsruhe, Germany, March 1995.
23. Nourani1994c,Nourani,C.F."Types, Induction, and Incompleteness," 1982, *Technical Report (Revised 1991), EATCS Bulletin*, Page 226-, June 1994.
24. Nourani 1995f, Nourani, C.F., "Language Dynamics and Syntactic Models", August 1994, Proc. *The IXth Conference Nordic and General Linguistics*, pages 26-27, The University of Oslo, Norway, January 1995.
25. Nourani 1997, Nourani,C.F., "Multiagent Chess Games," *AAAI Chess Track*, RI, July 1997