Automated reasoning with function evaluation for COCOLOG

Suning Wang, Peter E. Caines

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

1992

2ème anniversaire

N° 1713

Programme 5
Traitement du Signal,
Automatique et Productique

AUTOMATED REASONING WITH FUNCTION EVALUATION FOR COCOLOG

Suning WANG
Peter E. CAINES

Juin 1992

RR-1713

Automated Reasoning with Function Evaluation for COCOLOG

Suning Wang

Department of Electrical Engineering, McGill University.
em:wsn@moe.mcrcim.mcgill.edu,
Currently at Spar Aerospace Ltd., 21025 Trans-Canada Highway,
Ste-Anne-de-Bellevue,
Quebec, Canada H9X 3R2, Tel.(514)-457-2150

and

Peter E. Caines

Department of Electrical Engineering, McGill University, 3480 University Street,
Montreal, P.Q., H3A 2A7 Canada and the Canadian Institute for Advanced Research,
Tel.(514)-398-7129, em:peterc@moe.mcrcim.mcgill.edu

Abstract

In [2] we introduced a logic for the study of dynamical control systems which we called a Conditional Observer and Controller Logic for finite machines (COCOLOG). In that paper the properties of consistency and completeness for each first order logical theory in the tree of such theories in a COCOLOG were established. The efficacy of automatic theorem proving is a crucial issue in the implementation of COCOLOG. In this paper, we present a function evaluation based resolution, called FE–resolution, for COCOLOG. The unique model property of COCOLOG is first proved via the complete axiomatization property and then the decidability property of any theory in a COCOLOG is deduced. Finally, completeness and complexity reduction of FE–resolution is discussed in terms of relative truthfulness and validity.

Raisonnement Automatique avec Evaluation de Fonction pour COCOLOG

Résumé

Nous présentons une méthode de démonstration automatique par résolution pour des systèmes appelés des logiques de l'observabilité et de la controllabilité conditionnelle (COCOLOG) [CW90]. Ces logiques sont créées pour le contrôle des systèmes dynamiques à entrées sorties. La méthode est basée sur l'évaluation de fonctions et s'appelle FE-résolution. La propriété d'unicité des modèles d'un système de logique COCOLOG est établie en premier : ensuite les deux propriétés d'axiomatisation complète et de décidabilité d'un système COCOLOG sont présentées. Pour finir, les problèmes de la complétude et de la réduction de la complexité de la méthode de FE-résolution sont approchées en termes de véracité et de validité.

1 Introduction

In [2], we introduced families of first order theories which we called *conditional observer and controller logic* and we gave the acronym COCOLOG to any such a family of theories. This system of theories was created for the purpose of designing observers and controllers for partially observed input-state-output finite state machines. A COCOLOG system is given in the form of a family of axiomatic theories where the transitions between theories are regulated by a meta-level agent or an oracle function. The realizability of the logic-based control framework introduced by COCOLOG is based on the assumption that at each clock instant k, the logical theory $Th(o_1^k)$ can be generated instantly, or at least any theorem P of the theory $Th(o_1^k)$ can be generated instantly. This is indeed an idealization, since the automatic generation of a proof for a given theorem in the propositional calculus by resolution can be of exponential length as a function of the length of the theorem statement (see [7]) and the situation is even worse in the first order semi-decidable case (see [10]). (It is to be stressed that this statement is specific to resolution proofs. Bibel [1] has shown that the connection method gives quadratic proofs of the pigeonhole formulas.)

In this paper we discuss techniques which we propose for automatic theorem proving for COCOLOG which are essentially extensions of the Resolution Principle introduced by Robinson [14] and the Paramodulation technique for logics with equality, see [5]. A recent thesis by K. L. Myers ([11]) has proposed and analysed methods which have some points of contact with ours, and a detailed examination of the relationship between the various methodologies is merited.

Consider a formula A of a logical theory characterized by a set of axioms Σ. In order to prove A is a theorem by a resolution based automatic theorem proving procedure, Σ is first transformed into its clausal form, denoted by $\wedge\Sigma$, which can be viewed as a conjunction of clauses. By the Deduction Theorem we have that A is a theorem of Σ if and only if $\wedge\Sigma \rightarrow A$ is a valid formula, and this holds if and only if $\neg(\wedge\Sigma \rightarrow A)$ is unsatisfiable, i.e. $\wedge\Sigma \bigvee \neg A$ is unsatisfiable. The unsatisfiability property has to be verified under all interpretations (see [10] for a formal definition of interpretation). Herbrand showed that there exists a special class of interpretations, now called Herbrand interpretations, where a formula is unsatisfiable if and only if it is false in all Herbrand interpretations. The discovery of Herbrand interpretation made it possible to systematically generate proofs based on the verifications of all Herbrand interpretations. The complete, possibly infinite, set of, Herbrand interpretations can be organized in the form

of a semantic tree (see [5]), and one version of Herbrand theorem states that a set of formulae is unsatisfiable if and only a *finite closed semantic tree* exists, where a closed semantic tree contains all possible Herbrand interpretations for that set of formulae (see [5]). There is a direct correspondence between a finite closed semantic tree and a resolution proof for a theorem. As we pointed out above, Haken [7] has shown that there is a set of propositional formulae for which the length of a resolution proof can be exponential, with respect to the length of the formulae. Furthermore, in the first order case, a *search* for a proof of a formula which is not necessarily a theorem can be non-terminating due to the semi-decidable property of first order logic. This corresponds to the existence of a possibly infinite number of Herbrand interpretations over which a verification can be infinite. From the construction of a Herbrand universe and a Herbrand interpretation (see [5]) we can see that *infiniteness* of the universe is inevitable whenever a function symbol is introduced into the formulae. Our proposed approach is to check the logical descriptions of all functions and then add the capability of function evaluation to the resolution based proof procedure. The function evaluation technique will restrict the interpretation of the function, and hence all unique up to isomorphic interpretations will apply. Therefore the extended resolution based proof procedure will generate only valid formulae *relative* to the functions, or in other words, will verify only the set of unsatisfiable formulae relative to the functions. Furthermore, this evaluability of functions actually extends to predicates and this enable us to adapt our proof procedure to *formula evaluation.*

It is commonly thought that function definition and evaluation constitute essential parts of classical, or functional, programming and that the close relationship between specification and control of data flow makes it impossible for these two parts to be separated from each other. However, the recent development of logic programming suggests the issue of problem solving can be viewed as a process of logical theorem proving, where, in principle, he programming is decomposed into logic and control in two orthogonal dimensions, see [8]. The bridge connecting functional and logical programming is the way one chooses the representations of functions versus predicates.

In this paper we focus on the issues of completeness and complexity reduction pertaining to the proposed proof methodologies. We shall not discuss the issue of choosing representations of functions versus predicates systematically so as to perform optimally with respect to any criterion. But we should make it clear that our basic motivation is (as in paramodulation) to decrease the number of axioms

in play and hence to increase the efficiency of resolution techniques which depend exponentially upon the size of the corresponding base clause set. As explained above, and in detail in Sections 3 and 4, this is bought at the cost of passing from validity to relative validity with respect to models satisfying the axioms. However, the models lost in this way are thought to be of no significance.

2 COCOLOG for Finite Machines

As we introduced in [2], a conditional observer and controller logic is a specially constructed family of first order theories. It is distinguished from other logical systems in that it consists of a collection (in fact a tree) of theories concerning a finite machine which are coordinated in a particular fashion The reader is referred to the discussion in [2, 4]. In this section, we first examine the syntactical structure of a single theory in a COCOLOG. Then we establish the unique model property for any such a theory in the COCOLOG family. From this unique model result we can then derive the *completeness* of the axiomatization and therefore the *decidability* of each theory in any given COCOLOG.

2.1 COCOLOG: Syntax and Semantics

A COCOLOG logic system consists of a partially ordered set, or family, of first order logics. Each of these logics corresponds to a node in the observer tree of a given finite machine. The family of these individual logic systems constitutes a logic-based dynamical system which evolves with its environment and updates its structure as time proceeds.

To be more precise, each of the logics is equipped with the observed input and output as the data axioms of the corresponding node in the observer tree and is able to make all logical inference steps based on data axioms. We present this family of first order logics in terms of axiomatic theories. (For an introduction to axiomatic systems see [10].) In the theory presented in this chapter, we say that we let our COCOLOG system run in *real time* with the observation and control tasks, meaning that we assume all sound inferences following from a given set of axioms are available instantaneously before the next clock instant. The issue of automatic theorem proving will be addressed later in Sections 3 and 4.

In this section, we start with an introduction of the COCOLOG *language* and then we will present the *syntax* and *semantics* of the "static" part of the

COCOLOG, i.e., the logic corresponding to the root node in the observation tree.

2.2 COCOLOG Language L

The COCOLOG language consists of a set of symbols $S(L)$ and specified formation rules (or syntax). The concerning subject of COCOLOG language is the finite machine given as $\mathcal{M} = (X^{\mathcal{M}}, U^{\mathcal{M}}, Y^{\mathcal{M}}, \Phi, \eta)$[1], where $X^{\mathcal{M}}$ is the set of *states*, $U^{\mathcal{M}}$ is the set of *controls*, $Y^{\mathcal{M}}$ is the set of *output*, Φ is a state transition function, $\Phi : X^{\mathcal{M}} \times U^{\mathcal{M}} \rightarrow X^{\mathcal{M}}$ and η is a state output function, $\eta : X^{\mathcal{M}} \rightarrow Y^{\mathcal{M}}$.

We first define $S(L)$ as follows:

$$S(L) = Apr_L \bigcup Fun_L \bigcup Var_L \bigcup Cons_L \bigcup Qua_L \bigcup Lco_L \bigcup \{\perp\}.$$

The component sets of $S(L)$ are defined as follows:

Constant Symbols

$$Cons_L = \{x^1, \cdots, x^N\} \bigcup \{y^1, \cdots, y^p\} \bigcup \{u^1, \cdots, u^m, u^*\} \bigcup \{0, 1, \cdots, k(N)\}.$$

where $k(N)$ is the upper bound on time. [2]

Variable Symbols

$$Var_L = \{x, x', x'', \cdots,\} \bigcup \{y, y', y'', \cdots,\} \bigcup \{u, u', u'', \cdots,\} \bigcup \{l, l', \cdots,\}.$$

Where the variables are intended to be varying in different *sorts* or domains, e.g., variables $x, x', x'', \cdots$ will be interpreted to represent elements in the set of states X, variables $y, y', \cdots$ will be interpreted to represent elements in the set of state output Y, and so on.

Function Symbols

$$Fun_L = \{U(\cdot), Y(\cdot), \overline{\Phi}(\cdot,\cdot), \overline{\eta}(\cdot), +_L(\cdot,\cdot), -_L(\cdot,\cdot)\}.$$

[1] We use superscrip to denote the relevant finite machine, e.g., $X^{\mathcal{M}}$ denote the set of states of machine $\mathcal{M}$.

[2] k(N) is taken to be an arbitrary large number, for example $|X|$ or $|X|^2$ since as we can see from the results presented in [3] that an initial or current state observer dag can have at most $|X|$ or $|X|^2$ non-singleton layers before it split into singleton nodes.

where the sort of each function symbol is defined as the follows:
$U(a)$: a is a symbol either in $\{u^1, \cdots, u^m, u^*\}$ or in $\{u, u', \cdots, \}$.
$Y(a)$: a is a symbol either in $\{y^1, \cdots, y^p\}$ or in $\{y, y', \cdots, \}$.
$\overline{\Phi}(a, b)$: a is a symbol either in $\{x^1, \cdots, x^N\}$ or in $\{x, x', \cdots, \}$; and b is a symbol either in $\{u^1, \cdots, u^m, u^*\}$ or in $\{u, u', \cdots, \}$.
$\overline{\eta}(a)$: a is a symbol either in $\{x^1, \cdots, x^N\}$ or in $\{x, x', \cdots, \}$.
$+_L(a, b)$ and $-_L(a, b)$: a, b are symbols either in $\{1, 2, \cdots, \}$ or in $\{l, l', \cdots, \}$.

Terms

(i) Each constant and variable symbol is a term, i.e., $Cons_L \bigcup Var_L \subseteq Term_L$
(ii) If t is a term and f is a function symbol then $f(t)$ is a term
(iii) $Term_L$ are constructed only by steps (i) and (ii) above.

Atomic Predicate Symbols

$$Apr_L = \{Eq(\cdot, \cdot), Rbl(\cdot, \cdot, \cdot)\}.$$

Quantifiers

$$Qua_L = \{\forall\}$$

Logic Connectives

$$Lco_L = \{\rightarrow\}$$

2.3 Syntax of COCOLOG L

Any *well formed formula* of L is given by the *Backus-Naur* syntactic rule, see [6]:

$$A ::= \varphi(t_1, \cdots, t_n) \mid A_1 \rightarrow A_2 \mid \bot \mid \forall v A \quad ; \; where \; \varphi(t_1, \cdots, t_n) \in Apr_L, \; t_1, \cdots, t_n \in Term$$

and the set of such formulas will be denoted Fma_L.

The other logical connectives ($\neg, \vee, \wedge, \longleftrightarrow$) and quantifier ($\exists$) are defined as follows where (and) are used wherever the meaning of the formula can be

made clearer:

$$\begin{aligned} \neg A &::= A \rightarrow \bot \\ A_1 \wedge A_2 &::= \neg(A_1 \rightarrow A_2) \\ A_1 \longleftrightarrow A_2 &::= (A_1 \rightarrow A_2) \wedge (A_2 \rightarrow A_1) \\ A_1 \vee A_2 &::= \neg A_1 \rightarrow A_2 \\ \exists v A &::= \neg(\forall v \neg A). \end{aligned}$$

We observe that, as introduced here, L is a *multi-sort* language, where variables may vary within different domains. This complication can easily be removed by using a *sort predicate* for each variable, and by replacing each quantified formula via the following equivalence operation:

$$\forall x A(x) \equiv \forall x(X(x) \longrightarrow A(x)),$$
$$\exists x A(x) \equiv \exists x(X(x) \longrightarrow A(x)),$$

where $X(x)$ is the sort predicate for variable x, indicating the membership relation of $x \in X$. This rewriting will allow variables to vary freely within a single domain and hence we get a *single-sort* language. In the rest of the paper, we will not distinguish between a formula and its rewritten version for reasons of simplicity.

2.4 Semantics of COCOLOG L

A L-structure $\mathcal{U}_L = (\mathbf{D}, I)$ is a pair, where $\mathbf{D} = \mathbf{X} \cup \mathbf{Y} \cup \mathbf{U} \cup \mathbf{I_{k(N)}}$ is the domain of interest and I is an interpretation function defined as follows [3]:

$$\begin{aligned} I(\overline{\Phi}) &= \Phi : \mathbf{X} \times \mathbf{U} \rightarrow \mathbf{X} \\ I(\overline{\eta}) &= \eta : \mathbf{X} \rightarrow \mathbf{Y} \\ I(+_L) &= +_{k(N)} : \mathbf{I_{k(N)}} \times \mathbf{I_{k(N)}} \rightarrow \mathbf{I_{k(N)}} \\ I(-_L) &= -_{k(N)} : \mathbf{I_{k(N)}} \times \mathbf{I_{k(N)}} \rightarrow \mathbf{I_{k(N)}} \\ I(c) &= \mathbf{c} \in \mathbf{D} \\ I(Eq) &= \left\{(\mathbf{t}, \mathbf{t}') \mid \mathbf{t}, \mathbf{t}' \in \mathbf{D}, \mathbf{t} = \mathbf{t}'\right\} \subseteq \mathbf{D}^2 \\ I(Rbl) &= \left\{(\mathbf{x}, \mathbf{x}', \mathbf{k}) \mid \exists \mathbf{u}_1^{\mathbf{k}} \in \mathbf{U}^{\mathbf{k}}, \Phi(\mathbf{x}, \mathbf{u}_1^{\mathbf{k}}) = \mathbf{x}'\right\} \subseteq \mathbf{X}^2 \times \mathbf{I_{k(N)}} \end{aligned}$$

[3] We distinguish symbols used in COCOLOG language and in the base finite machine $\mathcal{M}$ by the convention that bold face letters denote constant and variable symbols in the base finite machine. COCOLOG function symbols will be denoted by a bar over the corresponding function symbols in the base machine.

Where the addition, $+_{k(N)}$, and subtraction, $-_{k(N)}$, over finite integers $\{\mathbf{1}, \mathbf{2}, \cdots, \mathbf{k(N)}+\mathbf{1}\}$ are defined by the following expressions:[4]

$$\mathbf{a} +_{k(N)} \mathbf{b} = \begin{cases} \mathbf{a}+\mathbf{b} & \text{if } \mathbf{a}+\mathbf{b} \leq \mathbf{k(N)} \\ \mathbf{k(N)}+\mathbf{1} & \text{if } \mathbf{a}+\mathbf{b} > \mathbf{k(N)} \end{cases}$$

$$\mathbf{a} -_{k(N)} \mathbf{b} = \begin{cases} \mathbf{a}-\mathbf{b} & \text{if } \mathbf{a}-\mathbf{b} \geq \mathbf{1} \\ \mathbf{k(N)}+\mathbf{1} & \text{if } \mathbf{a}-\mathbf{b} < \mathbf{1} \end{cases}$$

These finite integer arithmetical operations are necessary to express control properties in terms of integer number of steps.

A $\mathcal{U}_L$-valuation is a function $V : Var_L \rightarrow \mathbf{D}$ satisfying

$$V(v) \in \begin{cases} \mathbf{X} & \text{if } v = x \\ \mathbf{Y} & \text{if } v = y \\ \mathbf{U} & \text{if } v = u \\ \mathbf{I_{k(N)}} & \text{if } v = k \end{cases}$$

and can be extended to $V : Term_L \rightarrow \mathbf{D}$ by

$$V(t) = \begin{cases} V(t) & \text{if } t \in Var_L \\ I(t) & \text{if } t \in Cons_L \\ I(f)(V(t_1), V(t_2)) & \text{if } t = f(t_1, t_2) \text{ and } f \in Fun_L \end{cases}$$

We take $V \sim_v V'$ to mean that V and V' are identical except in the value they assign to v and

$$V(v/\mathbf{x}) = V' \text{ iff } V \sim_v V' \text{ and } V'(v) = \mathbf{x}$$

The satisfaction relation $\mathcal{U}_L \models A[V]$, which stands for the property that a formula A satisfies a structure $\mathcal{U}_L$ under the valuation V, is defined recursively by:

$\mathcal{U}_L \models Eq(t, t')[V]$ iff $V(t) = V(t')$,
$\mathcal{U}_L \models Rbl(x, x', k)[V]$ iff $(V(x), V(x'), V(k)) \in I(Rbl)$,
$\mathcal{U}_L \models (A_1 \rightarrow A_2)[V]$ iff $\mathcal{U}_L \models A_1[V]$ implies $\mathcal{U}_L \models A_2[V]$,
$\mathcal{U}_L \not\models \perp [V]$,
$\mathcal{U}_L \models \forall v A[V]$ iff for all $\mathbf{x} \in \mathbf{D}$, it is the case that $\mathcal{U}_L \models A[V(v/\mathbf{x})]$.

[4]Here we follow the convention that $+_L$ and $-_L$ are used in logic language to denote addition and subtraction function symbols; $+_{k(N)}$ and $-_{k(N)}$ denote the function symbols in the finite machine or in the semantic model; + and − denote the standard integer arithmetical operations.

A formula A is *true*, written $\mathcal{U}_L \models A$, in the structure $\mathcal{U}_L$ is defined by:

$$\mathcal{U}_L \models A \quad \text{iff} \quad \text{for all } V, \text{ it is the case that } \mathcal{U}_L \models A[V];$$

conversely, A is *false*, written $\mathcal{U}_L \not\models A$, is defined by:

$$\mathcal{U}_L \not\models A \quad \text{iff} \quad \text{for all } V, \text{ it is the case that } \mathcal{U}_L \not\models A[V]$$

A formula $\mathcal{A}$ is called *valid* if it is true in all structures $\mathcal{U}_L$, i.e., $\mathcal{A}$ is valid if and only if for all $\mathcal{U}_L$, $\mathcal{U}_L \models \mathcal{A}$. A formula $\mathcal{A}$ is *satisfiable* if there exists some structure $\mathcal{U}_L$ and some $\mathcal{U}_L$ valuation V such that the satisfaction relation $\mathcal{U}_L \models \mathcal{A}[V]$ holds. Obviously a formula $\mathcal{A}$ is valid if and only if $\neg\mathcal{A}$ is unsatisfiable.

2.5 Axiomatic Theory of Th_0

A formal logic theory of a language L consists of a set of *axioms*, that is to say a set of formulas from Fma_L, which shall be required to hold in the intended models and a set of relations on Fma_L, i.e. which are called the set of *inference rules*, together with concepts of a *proof* and *theoremhood*.

There are two sets of axioms in Th_0, one is the *logical axioms* which are a set of valid formulas (i.e., true in all models) which together with the rules of inference generate all valid formulas; the other is a set of *special axioms* which specify the true facts concerning the subject that logic describes. Correspondingly $Th(o_1^k)$ is a logical theory that has the input and output data as data axioms added to the logical theory Th_0.

We include logical axioms and equality axioms in Th_0, for more details of these axioms see [10].

The special axioms for a given finite machine $\mathcal{M}$ are described as follows:

FINITE MACHINE AXIOMS

For any pair of constants $\mathbf{x}^i, \mathbf{x}^j \in X^{\mathcal{M}}, \mathbf{u}^i \in U^{\mathcal{M}}$, if $\mathbf{x}^j = \Phi(\mathbf{x}^i, \mathbf{u}^i)$ is satisfied by the given finite machine $\mathcal{M}$ then we have the following *dynamic* axiom:

$$Eq(\overline{\Phi}(x^i, u^i), x^j) \qquad \textbf{AXM}^{\textbf{dyn}}(\textbf{L})$$

The dynamic axioms state the facts specifying the state transition function of the given finite machine $\mathcal{M}$. We note that the number of dynamic axioms is equal to $|X^{\mathcal{M}}||U^{\mathcal{M}}|$.

For any pair of constants $\mathbf{x^i} \in X^{\mathcal{M}}, \mathbf{y^i} \in Y^{\mathcal{M}}$ satisfying the relation $\eta(\mathbf{x^i}) = \mathbf{y^i}$, we have the following *output* axiom:

$$Eq(\overline{\eta}(x^i), y^i) \qquad\qquad \mathbf{AXM^{out}(L)}$$

The output axioms state the facts specifying output function of $\mathcal{M}$. The number of output axioms is equal to $|X^{\mathcal{M}}|$.

The finite machine axioms given above correspond to an infinite number of models. We get a unique model (up to name change isomorphism) when we further make the specifications of $|X| = N, |Y| = p$ and $|U| = m$ in terms of axioms. Moreover, we also need to specify addition and subtraction functions for the finite domain $\{\mathbf{1}, \mathbf{2}, \cdots, \mathbf{k(N)+1}\}$ of integers.

REACHABILITY AXIOMS

We recursively define the *reachability predicate* $Rbl(\cdot,\cdot,\cdot)$ by the following axioms:

0. $\forall x \forall x', \mathsf{E}q(x,x') \longleftrightarrow Rbl(x,x',0)$
1. $\forall x \forall x', (\exists u, \mathsf{E}q(\overline{\Phi}(x,u),x')) \longleftrightarrow Rb\ell(x,x',1)$
2. $\forall x \forall x'' \forall l, \mathsf{E}q(\ell, k(N)) \vee \mathsf{E}q(\ell, k(N)\dot{+}1) \vee [\{\exists x' \exists u, Rbl(x',x'',\ell) \wedge \mathsf{E}q(\overline{\Phi}(x,u),x')\}$ $\leftrightarrow Rbl(x,x'',\ell +_L 1)]$ $\qquad\qquad \mathbf{AXM^{Rbl}(L)}$
3. $\forall x \forall x', R(x,x',k(N)+1).$

The reachability axioms specify the l step reachability relation $Rbl(x,x',l)$ among any pair of states x, x'. We note that in these formulas the variables x, x', x'' range over X, the variable u ranges over U and l ranges over the integers $0, 1, \cdots, k(N)+1$. Formally Axiom 3 make all states reachable from each other in $k(N)+1$ steps, i.e. in the number of steps that plays the rôle of infinity in our arithmetic, while Axiom 2 excludes consideration of the infinity case in order to characterize reachability on the finite numbers in the arithmetic.

Rules of Inference:

R1. MODUS PONENS

$$\frac{\mathcal{A}, \mathcal{A} \rightarrow \mathcal{B}}{\mathcal{B}} \qquad ; \; where \; \mathcal{A}, \mathcal{B} \in Fma_L$$

R2. GENERALIZATION

$$\frac{\mathcal{A}}{\forall v \mathcal{A}} \qquad ; \ where \ \ v \in Var_L$$

We write $AXM^{spe}(L)$ to denote the set of special axioms of L, i.e., $AXM^{spe}(L) = \{AXM^{dyn}(L), AXM^{out}(L), AXM^{rbl}(L)\}$. We sometimes use Σ to denote $AXM^{spe}(L)$ for simplicity.

A *proof* in L is a sequence of formulas $\mathcal{A}_1, \cdots, \mathcal{A}_k$ in Fma_L where $\mathcal{A}_i$ is either an axiom or a direct consequence of previous formulas via $R1$ or $R2$. The last formula $\mathcal{A}_k$ in the sequence is called a *theorem* and $\mathcal{A}_1, \cdots, \mathcal{A}_{k-1}$ is a *proof* of theorem $\mathcal{A}_k$

A formula $\mathcal{A}$ is a theorem of a first order theory with equality, written $\vdash_L \mathcal{A}$, if in a proof of $\mathcal{A}$ only logical axioms and equality axioms have been involved. On the other hand, $\mathcal{A}$ is called a *consequence* (or theorem) of Σ, written $\Sigma \vdash_L \mathcal{A}$, if in a proof of $\mathcal{A}$ axioms in Σ may also have been involved.

For brevity we write Th_0 for $Th_0(L)$ which stands for the set of theorems of Σ; hence we have $Th_0 = \{\mathcal{A} : \Sigma \vdash_L \mathcal{A}\}$ and we shall use the standard notation $Th_0 \vdash A$ which is customarily read as A is a *theorem* of or *provable* (*derivable*) in the theory Th_0.

A structure $\mathcal{U}_L$ of theory Th_0 is called a *model* of the theory if and only if all the axioms of Th_0 are interpreted *true* in $\mathcal{U}_L$.

2.6 Observation Dependent COCOLOGs: $Th(o_1^k)$

Corresponding to a path in an observation tree, the family of these logics are generated by receiving the observation sequence $o_1^k = \{(\phi, y_1), (u_1, y_2), \cdots, (u_{k-1}, y_k)\}$. These sequence of logical theories $Th(o_1), Th(o_1^2), \cdots, Th(o_1^k)$. are observation dependent, we call them observation dependent COCOLOGs.

2.6.1 COCOLOG Language $L(o_1^k)$ and Syntax

The language $L(o_1^k)$ is extended from the language L by adding new atomic predicates defined as follows:

$$S(L(o_1^k)) = S(L) \bigcup_{j=1}^{k} Apr_j, \ \text{ where } \ Apr_j = \left\{eCSE_j(\cdot)\right\}.$$

We define $Fma_{L_0} = Fma_L$. The set of well formed formulae $Fma_{L(o_1^k)}$ is then defined by: $A ::= \varphi_k \mid B \mid A' \rightarrow A'' \mid \forall v A'$ where $B \in Fma_{L(o_1^{k-1})}$, $\varphi_k \in Apr_k$ and $A', A'' \in Fma_{L(o_1^k)}$.

Again the properties *true* and *false* for a formula and the concept of a *model* for a theory $Th(o_1^k)$ are defined in the sense of Tarski model semantics.

2.6.2 Axiomatic Theory of $Th(o_1^k)$

We assume that at each instant k, the observer will observe $u(k-1)$ and $y(k)$, and for each u^i and y^i such that $u^i = u(k-1)$ and $y^i = y(k)$, the following formulae written as observation axioms will be used to form the axiomatic theory $Th(o_1^k)$ of $L(o_1^k)$.

OBSERVATION AXIOMS

$$Eq(Y(k), y^i) \qquad \textbf{AXM}^{\textbf{out}}(\textbf{ob}, \textbf{L}(\textbf{o}_1^{\textbf{k}}))$$

$$Eq(U(k-1), u^i) \qquad \textbf{AXM}^{\textbf{cntl}}(\textbf{ob}, \textbf{L}(\textbf{o}_1^{\textbf{k}}))$$

STATE ESTIMATION AXIOMS

The following is the general form of a set of *Axioms of Conditional State Estimation*, where $C_j(\cdot,\cdot)$ is a conditional formula expressible in terms of $Fma_{L(o_1^{k-1})}$ and $AXM^{out}(ob, L(o_1^k))$.

$$\begin{array}{lcll} C_1(Fma_{L(o_1^{k-1})}, Eq(Y(k), y)) & \longrightarrow & eCSE_k(x^1) & \textbf{AXM}^{\textbf{est}}(\textbf{ob}, \textbf{L}(\textbf{o}_1^{\textbf{k}})) \\ \vdots & \longrightarrow & \vdots & \\ C_N(Fma_{L(o_1^{k-1})}, Eq(Y(k), y)) & \longrightarrow & eCSE_k(x^N). & \end{array}$$

Here we denote by Σ_k^o the set of axioms $\Sigma \bigcup_{j=1}^k \{AXM^{out}(ob, L(o_1^j)), AXM^{cntl}(ob, L(o_1^j)), AXM^{est}(ob, L(o_1^j), +), AXM^{est}(ob, L(o_1^j), -)\}$. We define an *Observation Theory* $Th^o(o_1^k)$ of Σ_k^o at the instant k by $Th^o(o_1^k) = \{\mathcal{A} : \Sigma_k^o \vdash_L \mathcal{A}\}$. Next we consider a control theory at the instant k.

CONTROL AXIOMS

The following is the general form of a set of *Control Axioms*, where $C_j(\cdot)$ is a *conditional formula* expressible in terms of $Fma_{L(o_1^k)}$.

$$\begin{array}{lcll} C_1(Fma_{L(o_1^k)}) & \longrightarrow & Eq(U(k), u^1) & \textbf{AXM}^{\textbf{ass}}(\textbf{cntl}, \textbf{L}(\textbf{o}_1^{\textbf{k}})) \\ \vdots & \longrightarrow & \vdots & \\ (\wedge_{i=1}^{m-1} \neg C_i) \wedge C_m & \longrightarrow & Eq(U(k), u^m). & \end{array}$$

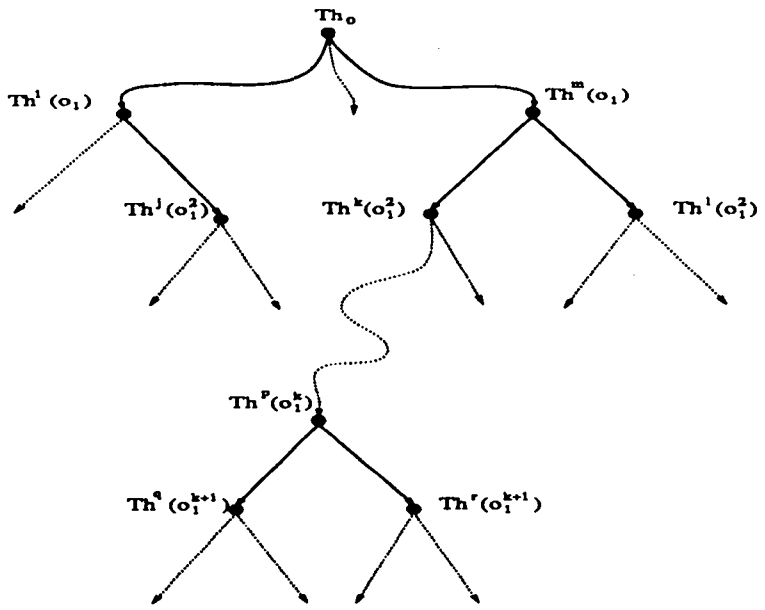


Figure 1: The Tree Structure of a COCOLOG Family

2.7 Extra-Logical Transitions Between Logical Theories

A realization of a COCOLOG is a sequence of first order theories generated by a given sequence of observations. It corresponds to a path in the COCOLOG tree structure (see Figure 1).

the *true* formulas in the nodes of this tree can be captured by a *possible world* interpretation of a *modal logic*, see [6]. Instead of modal logic, we use a family of classical first order logics to codify the state observation and control problem since we believe a modal logic representation would be too restrictive. The word restrictive is used in the following two senses: First, it can easily represent a *static* world. In other words, a modal logic cannot handle *unknowns* or the *changes* in the dynamics or the environment of the system and this prohibit the use of the logic for real time control tasks. Second, it is not necessary to *code* all the *paths* of an observation tree into a logic since a physical system cannot realize all such possibilities. Therefore the extra coding of modal logic system will simply further delay its response time. For more discussion on this see [4].

In order for the family of logics in a COCOLOG to work coherently, certain requirements have to be met. These requirements can be viewed as requirements

on the transitions between logical theories which cannot be represented in these theories themselves. Hence the extra-logical feature of the transitions must be described at a meta-level. In the following, we represent the meta-level requirement as meta-level axioms and the meta-level properties as meta-level rules of inference.

Meta-level axioms will be used to describe the assumption that there are no errors on the observation channel and the control actions sent from the logic controller will be implemented *instantly* and *correctly*. Hence there will not exist any conflict between observation and control axioms and reality.

We write $\Sigma_k^{o,c}$ as $\Sigma_k^o \bigcup_{j=1}^k \{AXM^{ass}(cntl, L(o_1^k))\}$ and define an *Observation and Control Theory* $Th^{o,c}(o_1^k)$ of $\Sigma_k^{o,c}$ as $Th^{o,c}(o_1^k) = \{\mathcal{A} : \Sigma_k^{o,c} \vdash_L \mathcal{A}\}$. From this definition we deduce the following rule of inference to connect theories at different instants along a trajectory of observations and control actions for a finite machine.

NESTING OF THEORIES (Meta-Level Rule of Inference)

$A \in Th^o(o_1^k)$ implies $A \in Th^{o,c}(o_1^k)$, $A \in Th^{o,c}(o_1^k)$ implies $A \in Th^o(o_1^{k'})$, for any $k' > k$.

The sequence of theories satisfying the following condition

$$\cdots \subseteq Th^o(o_1^k) \subseteq Th^{c,o}(o_1^k) \subseteq Th^o(o_1^{k+1}) \subseteq \cdots$$

We see that this sequence of COCOLOGs combined with the meta-level requirements constitute a *closed loop feedback* logical control system as displayed in Figure 2

Next we show that each theory in a COCOLOG family as introduced in [2] can be extended to a theory that corresponds to a unique model up to isomorphism. Here we take Th_0 as an example and we shall first review the axiomatic theory Th_0.

2.8 Unique Model Property

Now by giving the machine axioms as before there can be infinitely many different models. For example, any finite or infinite machine $M' = (X^{M'}, U^{M'}, Y^{M'}, \Phi', \eta')$ satisfying $X^M \subseteq X^{M'}, U^M \subseteq U^{M'}, Y^M \subseteq Y^{M'}$ and such that Φ' and η' are compatible with Φ and η up to X^M, Y^M and U^M, can be a model of the given machine axioms. Hence the machine axioms alone cannot uniquely characterize

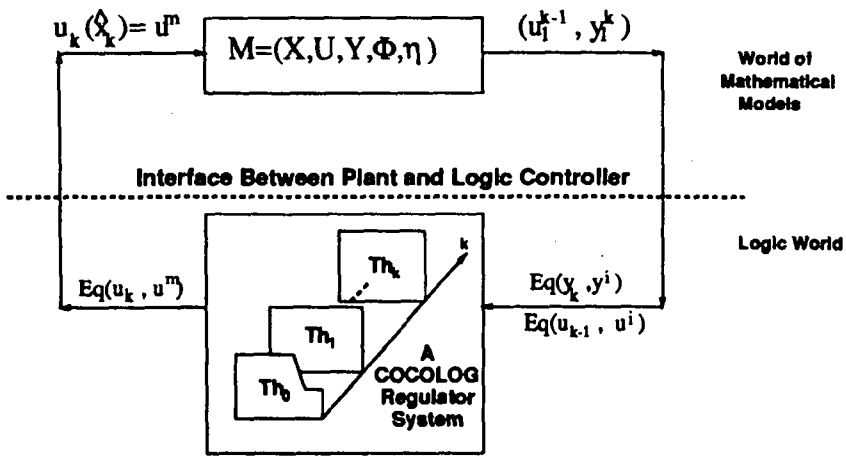


Figure 2: A Closed Loop Logic Control System

a given finite machine. In fact, one cannot determine a unique model by any given set of axioms. The most one can achieve by axiomatization is a set of equivalent models up to isomorphism. Hence the uniqueness will be used in this sense only.

The Size Axiom of $X^{\mathcal{M}}$

$$\begin{aligned} X_N^{\mathcal{M}}: \ & \neg Eq(x^1,x^2) \wedge \neg Eq(x^1,x^3) \wedge Eq(x^1,x^4) \wedge \dots \wedge \neg Eq(x^1,x^N) \\ & \wedge \neg Eq(x^2,x^3) \wedge \neg Eq(x^2,x^4) \wedge \cdots \wedge \neg Eq(x^2,x^N) \\ & \wedge \neg Eq(x^3,x^4) \wedge \cdots \wedge \neg Eq(x^3,x^N) \\ & \vdots \\ & \wedge \neg Eq(x^{N-1},x^N) \end{aligned}$$

$X_N^{\mathcal{M}}$ specify the fact that there are at least N distinct constant symbols in the state space $X^{\mathcal{M}}$ of the finite machine $\mathcal{M}$, i.e., $|X^{\mathcal{M}}| \geq N$.

Next we specify the fact there are at most N elements in the intended model by $\neg X_{N+1}^{\mathcal{M}}$.

$$\neg X_{N+1}^{\mathcal{M}}: \ \forall x \left(\bigvee_{i=1}^{N} Eq(x,x^i) \right).$$

By adding $X_N^{\mathcal{M}}$ and $\neg X_{N+1}^{\mathcal{M}}$ to the originally proposed machine axioms the only models one can get will be the models that have exactly N distinct states.

That is the set of N-state machines in which Φ and η are given as specified. Further, if we add restrictions on the size of $U^{\mathcal{M}}$ and $Y^{\mathcal{M}}$ we get a unique model for $\mathcal{M}$.

In the following we denote M and M' as finite machines and we also use them to denote the sets of elements in each machine as $M = X^M \bigcup U^M \bigcup Y^M$ and $M' = X^{M'} \bigcup U^{M'} \bigcup Y^{M'}$

Definition 2.1 (Homomorphism) If $\mathcal{M}$ and $\mathcal{M}'$ are two finite machines, then a map h from $\mathcal{M}$ to $\mathcal{M}'$ is called a *homomorphism* if

$$\begin{aligned} h(\Phi(\mathbf{x},\mathbf{u})) &= \Phi'(h(\mathbf{x}),h(\mathbf{u})) \\ h(\eta(\mathbf{x})) &= \eta'(h(\mathbf{x})) \\ h(+_{k(N)}(\mathbf{I},\mathbf{I}')) &= +'_{k(N)}(h(\mathbf{I}),h(\mathbf{I}')) \\ h(-_{k(N)}(\mathbf{I},\mathbf{I}')) &= -'_{k(N)}(h(\mathbf{I}),h(\mathbf{I}')) \end{aligned}$$

□

Next we show the unique model property of the theory Th_0 when together with the size axioms of X^M, U^M and Y^M. Let us first denote Axm^M as the machine axioms for the given finite machine M and with

$$\begin{aligned} \Sigma^0_M = \; & Axm^M \bigcup X^M_N \bigcup \neg X^M_{N+1} \\ & \bigcup U^M_m \bigcup \neg U^M_{m+1} \\ & \bigcup Y^M_p \bigcup \neg Y^M_{p+1} \end{aligned}$$

for the set of axioms for the finite machine M at the instant zero.

Theorem 2.1 (Unique Model Property) If Σ^0_M is defined as above, then Σ^0_M has a unique model up to isomorphism. □

Proof

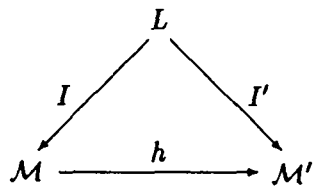
We begin by establishing the existence of a homomorphic mapping for any given pair of models of $\Sigma^0_{\mathcal{M}}$. This shows there is unique pre-interpretation for Σ^0_M, that is to say there is a homomorphism between all elements of any two models of the given theory excepting possibly the interpretations of the predicates, see [9].

Now consider any two models $\mathcal{M}$ and $\mathcal{M}'$ where $\mathcal{M} = (X^{\mathcal{M}}, Y^{\mathcal{M}}, U^{\mathcal{M}}, \Phi, \eta)$ and $\mathcal{M}' = (X^{\mathcal{M}'}, Y^{\mathcal{M}'}, U^{\mathcal{M}'}, \Phi', \eta')$. By the size axioms we have $|X^{\mathcal{M}}| = |X^{\mathcal{M}'}| = N$, $|Y^{\mathcal{M}}| = |Y^{\mathcal{M}'}| = p$ and $|U^{\mathcal{M}}| = |U^{\mathcal{M}'}| = m$. Then by the machine axioms we have $\Phi : X^{\mathcal{M}} \times U^{\mathcal{M}} \to X^{\mathcal{M}}$ and $\Phi' : X^{\mathcal{M}'} \times U^{\mathcal{M}'} \to X^{\mathcal{M}'}$; $\eta : X^{\mathcal{M}} \to Y^{\mathcal{M}}$ and $\eta' : X^{\mathcal{M}'} \times U^{\mathcal{M}'} \to X^{\mathcal{M}'}$. Now an one-to-one and onto mapping $h : \mathcal{M} \to \mathcal{M}'$ can be defined, where $\mathcal{M}$ here is also taken as the union of $X^{\mathcal{M}}$, $U^{\mathcal{M}}$ and $Y^{\mathcal{M}}$ and $\mathcal{M}'$ is also taken as the union of $X^{\mathcal{M}'}$, $U^{\mathcal{M}'}$ and $Y^{\mathcal{M}'}$.

Let L denote the set of symbols of logical theory for a finite machine, $I : L \to \mathcal{M}$ and $I' : L \to \mathcal{M}'$ be the interpretation functions correspond to the model $\mathcal{M}$ and $\mathcal{M}'$ respectively. Construct a mapping $h : \mathcal{M} \to \mathcal{M}'$ such that the following relation is satisfied:

$$h(m) = I'(I^{-1}(m)) \qquad \text{for any } m \in \mathcal{M}$$

The relations among the set of $L, \mathcal{M}, \mathcal{M}'$ and the mappings of I, I' and h are shown as follows:



We need to show that h is a bijective mapping. This property is guaranteed by the bijective mappings of I and I'.

First, onto can be shown by taking any $m' \in \mathcal{M}'$, then we have $I'^{-1}(m') = l$ for some $l \in L$ and $I(l) = m$ for some $m \in \mathcal{M}$. We can show that this m is the preimage of the m' under h.

$$\begin{aligned} h(m) &= I'(I^{-1}(m)) \\ &= I'(I^{-1}(I(I'^{-1}(m')))) \\ &= I'(I'^{-1}(m')) \\ &= m' \end{aligned}$$

Second, the one-to-one property can be shown by taking any $m_1, m_2 \in \mathcal{M}$, assume that

$$h(m_1) = h(m_2) \qquad \textit{but } m_1 \neq m_2$$

The following arguments will produce a contradiction as desired:

$$\begin{aligned} h(m_1) = h(m_2) \quad &\text{iff} \quad I'(I^{-1}(m_1)) = I'(I^{-1}(m_2)) \\ &\text{iff} \quad m_1 = m_2 \end{aligned}$$

This shows that the mapping h as constructed is a bijective mapping.

Now if we denote $h(m) = m'$ for any $m \in \mathcal{M}$ and $m' \in \mathcal{M}'$ and take any formula $Eq(\overline{\Phi}(x^i, u^i), x^j)$ from the language L. Interpretation I will map this formula to $\Phi(I(x^i), I(u^i)) = I(x^j)$ which is $\Phi(\mathbf{x}^i_m, \mathbf{u}^i_m) = \mathbf{x}^j_m$ and the interpretation I' will map the formula to $\Phi'(I'(x^i), I'(u^i)) = I'(x^j)$ which is $\Phi'(\mathbf{x}^i_{m'}, \mathbf{u}^i_{m'}) = \mathbf{x}^j_{m'}$. Now since $h(m) = I'(I^{-1}(m))$ we have the following relationship between the two models:

$$\begin{aligned} \Phi(\mathbf{x}^i_m, \mathbf{u}^i_m) = \mathbf{x}^j_m \quad &\text{iff} \quad \Phi'(\mathbf{x}^i_{m'}, \mathbf{u}^i_{m'}) = \mathbf{x}^j_{m'} \\ \eta(\mathbf{x}^i_m) = \mathbf{y}^i_m \quad &\text{iff} \quad \eta'(\mathbf{x}^i_{m'}) = \mathbf{y}^i_{m'} \\ +_{k(N)}(\mathbf{l}_m, \mathbf{l}'_m) = \mathbf{l}''_{m'} \quad &\text{iff} \quad +'_{k(N)}(\mathbf{l}_{m'}, \mathbf{l}'_{m'}) = \mathbf{l}''_{m'} \\ -_{k(N)}(\mathbf{l}_m, \mathbf{l}'_m) = \mathbf{l}''_{m'} \quad &\text{iff} \quad -'_{k(N)}(\mathbf{l}_{m'}, \mathbf{l}'_{m'}) = \mathbf{l}''_{m'} \end{aligned}$$

From this it follows immediately that the mapping h so defined is a homomorphism, i.e., h satisfies the following relationships:

$$\begin{aligned} h(\Phi(\mathbf{x}^i, \mathbf{u}^i)) &= \Phi'(h(\mathbf{x}^i), h(\mathbf{u}^i)) \\ h(\eta(\mathbf{x}^i)) &= \eta'(h(\mathbf{x}^i)) \\ h(+_{k(N)}(\mathbf{l}, \mathbf{l}')) &= +'_{k(N)}(h(\mathbf{l}), h(\mathbf{l}')) \\ h(-_{k(N)}(\mathbf{l}, \mathbf{l}')) &= -'_{k(N)}(h(\mathbf{l}), h(\mathbf{l}')) \end{aligned}$$

We conclude that $\Sigma^0_{\mathcal{M}}$ has a unique pre-interpretation up to isomorphism. Next,it is evident that the equality predicate has a unique interpretation on the elements of the pre-interpretation. And finally, an examination of the recursive nature of the reachability axioms $\mathbf{AXM}^{\mathbf{rbl}}(L)$ reveals that there is a unique relation on the elements of the (proven unique) pre-interpretation that will satisfy the reachability axioms. It follows that up to isomorhism $\Sigma^0_{\mathcal{M}}$ has a uniqe model,as required.

□

Definition 2.2 (Proper Formula) A formula $\mathcal{P}$ is a *proper formula* with respect to a set of formula Γ if $\mathcal{P}$ contain neither any predicate symbols nor function symbols which do not appear in any formulas in Γ □

Definition 2.3 (Complete Axiomatization) A set of formulas Γ is said to be *complete* if either $\mathcal{P}$ or $\neg\mathcal{P}$ is a logical consequence of Γ for any proper formula $\mathcal{P}$ with respect to Γ □

It is known result that if a set of axioms has an unique model then that set of axioms is complete. We state this in the following theorem.

Theorem 2.2 (Complete Axiomatization) Let Σ_M^0 be defined as above as a set of axioms of a given finite machine M, then Σ_M^0 is a complete axiomatization of M. □

Now from the theorem stated above we can oderive an important result on f as follows.

Proof

To prove that $\Sigma_{\mathcal{M}}^0$ is a complete axiomatization of $\mathcal{M}$, we need to show that for any formula $A \in L$ either $\Sigma_{\mathcal{M}}^0 \vdash A$ or $\Sigma_{\mathcal{M}}^0 \vdash \neg A$ is true. We know $\Sigma_{\mathcal{M}}^0$ is consitent since the existence of the models for $\Sigma_{\mathcal{M}}^0$. By Lindenbau's lemma, see [10] if $\Sigma_{\mathcal{M}}^0$ is consistent first order theory, then there is a consistent complete extension of $\Sigma_{\mathcal{M}}^0$. But since we know $\Sigma_{\mathcal{M}}^0$ has a unique model, see Theorem 2.1, this implies the complete extension of $\Sigma_{\mathcal{M}}^0$ is $\Sigma_{\mathcal{M}}^0$ and henc e $\Sigma_{\mathcal{M}}^0$ is complete since otherwise $\Sigma_{\mathcal{M}}^0$ cannot have unique model. □

Theorem 2.3 (Decidable Theoremhood) The axiomatic theory as generated by Σ_M^0 for any given finite machine M is decidable.

Proof

By the generalized completeness of first order logic in general and COCOLOG in particular, see the previous sections we know that for any formula $\mathcal{P}$, $\mathcal{P}$ is provable if $\mathcal{P}$ is a consequence of $\Sigma_{\mathcal{M}}^0$. Now for any formula $\mathcal{P}$ we start a search for all possible proofs for both $\mathcal{P}$ and $\neg\mathcal{P}$. One of these two searches will terminate since $\Sigma_{\mathcal{M}}^0$ is a complete axiomatization, i.e., either $\mathcal{P}$ or $\neg\mathcal{P}$ will be a consequence of $\Sigma_{\mathcal{M}}^0$. Thus we can conclude that the axiomatic theory generated by $\Sigma_{\mathcal{M}}^0$ is decidable.

□

These results can be generalized to any theory $Th(o_1^k)$ in a COCOLOG family.

3 Resolution Proofs of COCOLOG Theorems

The Resolution Principle introduced by Robinson see [14] has been widely used in mechanical theorem proving. In this section we discuss a mechanical theorem proving method for COCOLOG theorems in terms of a resolution based procedure. The extension of the resolution technique called paramodulation for logic with equality will be introduced since, in a COCOLOG theory, the equality predicate is one of the atomic predicates used for expressing the machine axioms.

Consider the machine axioms in a COOCLOG theory as described in the last section. They are given by the equality predicate $Eq(\cdot,\cdot)$. To prove a theorem in which the equality predicate is involved, a direct approach is to introduce extra axioms to describe the equality relations in the logical theory. An alternative to this, which is often more efficient, can be achieved by introducing extra inference rules.

We first examine how to describe an equality relation. We know that an equality is an equivalence relation and furthermore one can substitute an equal for an equal, i.e., identity substitution. For more discussion of this see [10, 5]. The truth for an equality predicate of a set of clause S can be axiomatized as follows denoted as $K_{=}(S)$.

Equality Axiom Schemata

$$Eq(x,x)$$
$$Eq(x,y) \rightarrow Eq(y,x)$$
$$Eq(x,y) \wedge Eq(y,z) \rightarrow Eq(x,z)$$
$$Eq(x,y) \rightarrow Eq(f(x),f(y)) \qquad \textit{for each function symbol } f \textit{ in } S$$
$$Eq(x,y) \rightarrow (P(x) \rightarrow P(y)) \qquad \textit{for each predicate symbol } P \textit{ in } S$$

For a resolution based theorem proving procedure, these extra number of axioms will certainly reduce the speed of a search or increase the size of the search space. Since the number of attempted unifications during the search of a resolution proof is often an exponential function of the number of the input axioms. The increase of the search space is generally an exponential function against the increase of the number of input axioms. The number of equality axioms is a linear or at most a polynomial function of the number of function and predicate symbols. To avoid such a dramatic increase in the size of the search space for resolution proofs, Robinson and Wos [13] proposed a generalized resolution principle called paramodulation using reference rules in addition to the

resolution principle. In the following we will examine this generalized resolution proof.

The key concepts for the formulation of the paramodulation technique are (1) equality interpretations, called E-interpretations, and (2) the notion of unsatisfiability in the class of equality models, called E-unsatisfiability. These are formally defined as follows in terms of Herbrand interpretations. Formal definitions of Herbrand universes and Herbrand interpretations can be found in any standard logic textbook, and for their use in mechanical theorem proving we refer the reader to see [5]. A Herbrand universe of a set of clauses S consists of the minimum set of symbols relevant to S: the elements of a Herbrand universe of a set of clauses S are the set of possible ground terms generated from those clauses using iterations of all function symbols that occur in S. A Herbrand interpretation is an interpretation based on a Herbrand universe: a Herbrand interpretation consists of a set of ground predicates, i.e., the predicates with terms being replaced by elements in the Herbrand universe. A ground predicate in a Herbrand interpretation is interpreted as true in that interpretation. Hence a Herbrand interpretation will contain either $\mathcal{P}$ or $\neg\mathcal{P}$ where $\mathcal{P}$ is any ground predicate. As we stated earlier, it has been shown that a set of clauses is unsatisfiable if and only if it is unsatisfiable under all Herbrand interpretations. Now let us define an E-interpretation.

Definition 3.1 An *E-interpretation* of a set of clauses S is a Herbrand interpretation I satisfies the following conditions.

1. $Eq(x,x) \in I$
2. $Eq(x,y) \in I$ *then* $Eq(y,x) \in I$
3. $Eq(x,y), Eq(y,z) \in I$ *then* $Eq(x,z) \in I$
4. $Eq(x,y) \in I, L[x] \in I$ *then* $L[y] \in I$

where x, y, z are elements in the Herbrand universe of S and L is an atomic predicate or a negation of an atomic predicate i.e., a *literal* in I. □

Definition 3.2 A set of clause S is called *E-satisfiable* if there is an E-interpretation that satisfies all the clauses in S. Otherwise S is called *E-unsatisfiable.* □

An E-interpretation is an interpretation that satisfies equality axioms, i.e. which is a model of an equality theory. Next we give a theorem that states E-interpretations indeed characterize the equality axioms.

Theorem 3.1 ([5]) Let S be a set of clauses and $K_{=}(S)$ be the equality axioms of S. Then S is E-unsatisfiable if and only if the set of $S \bigcup K_{=}(S)$ is unsatisfiable □

Furthermore, if S is a finite set then we have the following result.

Theorem 3.2 ([5]) A finite set S of clauses is E-unsatisfiable if and only if there is a finite set S' of ground instances of clauses in S such that S' is unsatisfiable. □

Next we introduce a new inference rule, paramodulation, in addition to the resolution principle, and then state the result that by using both resolution and paramodulation, we can deduce the empty clause □ from an E-unsatisfiable set of clauses. That is the combination of paramodulation and resolution is complete for E-unsatisfiable set of clauses.

Definition 3.3 (Paramodulation) Let C_1 and C_2 be two clauses with no variables in common. If $C_1 = L[t] \bigcup C_1'$ and $C_2 = Eq(r,s) \bigcup C_2'$, where $L[t]$ is a literal containing the term t and C_1' and C_2' are clauses, and if t and r have a most general unifier σ, then infer

$$L\sigma[s\sigma] \bigcup C_1'\sigma \bigcup C_2'\sigma$$

where $L\sigma[s\sigma]$ denotes the result obtained by replacing single occurrence of $t\sigma$ in $L\sigma$ by $s\sigma$. The above inferred clause is called a *binary paramodulant* of C_1 and C_2. We also say that we apply *paramodulation* from C_2 into C_1. □

Next consider the machine axioms in a COCOLOG theory. They are expressed in terms of equality predicate, state transition function and output function. Like the equality predicate these functions can also be expressed either by extra axioms or they can be embedded into a set of new inference rules hence increasing the efficiency of the proof procedure. The inference rules corresponding to these functions will be designed so that they will imply the empty clause □ from any set of clauses which are unsatisfiable in any of the models of the given finite machine M. We will call such models M-unsatisfiable.

4 FE-Resolution in a COCOLOG Theory

Taking our inspiration from the concepts of E-interpretation, E-satisfiability and paramodulation for equality predicates, we propose similar concepts for functions in a COCOLOG theory. The results derived in this section can be generalized to extend to other logical theories. The key idea here is to restrict the size of the Herbrand universe of a given set of clauses. Recall that a Herbrand universe of a set of clauses S is a minimum set of ground terms of S. The minimality is defined in the sense that for any interpretation I_s of S, there exists a Herbrand interpretation I_s^* of S from defined Herbrand universe such that a clause $C \in S$ is true in I_s then C is true in I_s^*. Therefore S is unsatisfiable if and only if S is false in all Herbrand interpretations. The point is that a Herbrand universe is designed so that one can construct Herbrand interpretations within which the validity of a formula can be verified. In fact what we want to characterize here is a concept of relative validity with respect to the given finite machine. To be more precise, we want to define a pseudo-Herbrand universe from which we can construct interpretations and verify the validity of a formula with respect to the finite machine. Relative validity means that a formula is interpreted as true in any model of the *given* finite machine. These are semantical constructions. The corresponding syntactical counterpart will be called FE-resolution. The idea is to add a function evaluation ability to the resolution principle and paramodulation. The syntax and the semantics will be connected by a completeness result for FE-resolution. FE-resolution can then be extended to the predicates in EA_{pr} and the set of evaluable atomic predicates at the predicate level (and therefore evaluable valid formulas with respect to the models of the given finite machine) can then be eliminated at the formula level.

Definition 4.1 ($\mathcal{M}$-Universe) Given a finite machine $\mathcal{M} = (X^{\mathcal{M}}, U^{\mathcal{M}}, Y^{\mathcal{M}}, \Phi, \eta)$ and a set of clauses S, an *$\mathcal{M}$-universe* is defined as the union of the following sets:

$$
\begin{aligned}
\mathcal{M}_0 &= \{a : a \text{ is a constant in } S\}, \\
\mathcal{M}_1 &= \{\Phi(a_x, a_u), \eta(a_x), +_{k(N)}(a_I, a_I'), -_{k(N)}(a_I, a_I') : \\
&\quad a_x \in \mathcal{M}_0 \bigcap X^{\mathcal{M}}\ a_u \in \mathcal{M}_0 \bigcap U^{\mathcal{M}}, \text{ and } a_I, a_I' \in \mathcal{M}_0 \bigcap I^{\mathcal{M}}_{k(N)}\} \\
&\vdots \\
\mathcal{M}_{k+1} &= \{\Phi(a_x, a_u), \eta(a_x), +_{k(N)}(a_I, a_I'), -_{k(N)}(a_I, a_I') :
\end{aligned}
$$

$$a_x \in \mathcal{M}_k \bigcap X^{\mathcal{M}} u \in \mathcal{M}_k \bigcap U^{\mathcal{M}}, \text{ and } a_I, a_I' \in \mathcal{M}_k \bigcap I^{\mathcal{M}}_{k(N)}\}$$

Thus an $\mathcal{M}$-universe is defined as

$$\mathcal{M} = \begin{cases} \bigcup_{k=0}^{\infty} \mathcal{M}_k & \text{if } \mathcal{M}_0 \neq \phi \\ X^{\mathcal{M}} \bigcup U^{\mathcal{M}} \bigcup Y^{\mathcal{M}} & \text{in case } \mathcal{M}_0 = \phi \end{cases}$$

Obviously $\mathcal{M} \subseteq X^{\mathcal{M}} \bigcup U^{\mathcal{M}} \bigcup Y^{\mathcal{M}}$ holds. □

Now we say a set A is an *atom set* (or a M *Base*) of S if A consists of only ground atoms occurring in S where terms are elements of an M-universe of S.

We define an interpretation on an M-universe of S as a pseudo-Herbrand interpretation where Herbrand universe being replaced by an M-universe. To be more specific, Let $A = \{A_1, A_2, \cdots, A_n, \cdots\}$ be the E atom set of S, then

$$I = \{m_1, m_2, \cdots, m_n, \cdots\}$$

is an interpretation if m_i is either A_i or $\neg A_i$. An E-interpretation over an M-universe will be defined as given in Definition 3.1, where the Herbrand Universe is replaced by the M-universe. Next we define an M-interpretation:

Definition 4.2 (M-Interpretation) Given a set of clauses S and a finite machine M, an *M-interpretation* is an E-interpretation which in addition satisfies the following conditions.

1. $Eq(\overline{\Phi}(a, b), a') \in I$ if $\Phi(a, b) = a'$
2. $Eq(\overline{\eta}(a), c) \in I$ if $\eta(a) = c$

For any $a, a' \in X^M$, $b \in U^M$ and $c \in Y^M$. □

In fact an M-interpretation is an pseudo-Herbrand interpretation where all machine axioms will be interpreted true and $Eq(\cdot, \cdot)$ is interpreted as the identity relation. A set of clauses S is said to be satisfiable in an interpretation I, denoted by $I \models_s S$, if and only if for any clause C in S there exists some ground clause C' of C such that C' is interpreted true in I. Truthfulness of first order formulas is then defined by use of the standard Tarski semantics (see e.g. [10]). Next we define M-valid and M-unsatisfiability.

Definition 4.3 (M-Relative Validity) *A* formula $\mathcal{A}$ is said to be *M-relative valid, or M-valid* if $\mathcal{A}$ is interpreted true in every M-interpretation, i.e. for each M-interpretation I_M we have $I_M \models \mathcal{A}$. □

Correspondingly we have:

Definition 4.4 (M-Relative Unsatisfiability) *A* formula $\mathcal{A}$ is said to be *M-relative unsatisfiable, or M-unsatisfiable* if $\mathcal{A}$ is interpreted false in every M-interpretation, i.e. for each M-interpretation I_M we have $I_M \not\models \mathcal{A}$. □

Obviously we have that $\mathcal{A}$ is M-relatively valid if and only if $\neg\mathcal{A}$ is M-relatively unsatisfiable. Next we show that an M-interpretation characterizes the class of models for the given finite machine M defined by the set of axioms Σ_M^0. But before we do so, we first give two Lemmas. It is a known result that if a set of clauses S is satisfiable in an interpretation I then S is also satisfiable in any Herbrand interpretation I^* corresponding to I. (see Lemma 4.1 in [5]). Now we extend this result from Herbrand interpretations to E-interpretations and M-interpretations respectively.

Lemma 4.1 Let $K_=(S)$ be the equality axioms for a set of clauses S, then any Herbrand interpretation I^* satisfies $K_=(S)$ implies I^* is also an E-interpretation. □

Proof. By the converse of Theorem 3.2, $I^* \models K_=(S)$ implies that for any axiom $A \in K_=(S)$, each ground instance A' of A is such that $I^* \models A'$. But since each axiom in $K_=(S)$ is assumed to be quantified universally it follows that any ground instance A' of any axiom A in $K_=(S)$ is satisfied by I^*. Therefore I^* meets all the conditions for an E-interpretation and so I^* is also an E-interpretation. □

Lemma 4.2 Let $\Sigma_M^0(S)$ be the machine axioms for a set of clauses S, then any E-interpretation I_E satisfying $\Sigma_M^0(S)$ is also an M-interpretation. □

Proof. The proof is similar to that of the previous Lemma. $I_E \models \Sigma_{\mathcal{M}}^0(S)$ implies that for any axiom $A \in \Sigma_{\mathcal{M}}^0(S)$ there exists a ground instance A' of A such that $I_E \models A'$. Since each axiom A in $\Sigma_{\mathcal{M}}^0(S)$ is universally quantified it follows the $I_E \models A$. This implies $I_E \models A'$ for any ground instance A' of A in $\Sigma_{\mathcal{M}}^0(S)$ and hence all the conditions for I_E to be an $\mathcal{M}$-interpretation are met. We conclude that I_E is an $\mathcal{M}$-interpretation. □

Theorem 4.1 Let S be a set of clauses, $\Sigma^0_M(S)$ and $K_=(S)$ as the set of machine axioms and equality axioms respectively. Then S is M-unsatisfiable if and only if $S \bigcup \Sigma^0_M \bigcup K_=(S)$ is unsatisfiable. □

Proof. Suppose S is $\mathcal{M}$-unsatisfiable but $S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$ is satisfiable. The latter implies that

$$I \models S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$$

This is equivalent to

$$I \models S \text{ and } I \models \Sigma^0_{\mathcal{M}}(S) \text{ and } I \models K_=(s)$$

This implies for any Herbrand interpretation I^* corresponding to I we have

$$I^* \models S \text{ and } I^* \models \Sigma^0_{\mathcal{M}}(S) \text{ and } I^* \models K_=(S)$$

By Lemma 4.1 we have

$$I_E \models S \text{ and } I_E \models \Sigma^0_{\mathcal{M}}(S)$$

By Lemma 4.2 we known that S is satisfied by an $\mathcal{M}$-interpretation, that is to say $I_{\mathcal{M}} \models S$, which contradicts the fact that S is $\mathcal{M}$-unsatisfiable and hence the result is proved.

The other direction of the proof is simple. Suppose $S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$ is unsatisfiable but S is $\mathcal{M}$-satisfiable. The latter statement implies there is an $\mathcal{M}$-interpretation $I_{\mathcal{M}} \models S$. Clearly we have $I_{\mathcal{M}} \models \Sigma^0_{\mathcal{M}}(S)$ and $I_{\mathcal{M}} \models K_=(S)$ since $I_{\mathcal{M}}$ is also an E-interpretation. Therefore we have $I_{\mathcal{M}} \models S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$ which contradicts to the assumption that $S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$ is unsatisfiable. This completes the proof. □

In the following we denote $f(x_1, \cdots, x_n)$ as a n-ary function with n variables $x_1, x_2, \cdots, x_n$,and $b \equiv f(a_1, \cdots, a_n)$ shall denote function f evaluated at constant $a_1, \cdots, a_n$ is b. Now we define rules for function evaluation based resolution, also called FE-resolution. Then we shall show that these rules combined with rules of standard resolution and paramodulation are complete. In particular, an empty clause will be generated from a set of M-unsatisfiable set of clause by applying these extended resolution. Function evaluation can be done either at the constant level or at the variable level, i.e., f can be partially evaluated at a_1, a_2, etc, to give the function $f(a_1, \cdots, a_{m-1}, x_m, \cdots, x_n)$. Thus FE-resolution should also be defined for constant and variable cases respectively.

Definition 4.5 (Constant FE-Resolution) Let C_1 be a clause which is in the form $C_1 = L[f(a_1, \cdots, a_n)] \vee C_1'$, where $L[f]$ is a literal containing the function symbol f in its ground form $f(a_1, \cdots, a_n)$, C_1' is a subclause of C_1, then infer

$$C = L[b] \vee C_1'$$

where $b \equiv f(a_1, \cdots, a_n)$. The inferred clause C is called a *constant FE-resolvent of* C_1. □

Applying constant FE-resolution will result in the generation of all equivalent clauses to those which contain ground terms but not ground function terms. Now constant FE-resolution can be extended to variable FE-resolution:

Definition 4.6 (Variable FE-Resolution) Let C_1 be a clause which is in the form

$$C_1 = L[f(a_1, \cdots, a_k, x_{k+1}, \cdots, x_n, y_1, \cdots, y_l)] \vee C_1'(x_{k+1}, \cdots, x_n, x_{n+1}, \cdots, x_{n+m})$$

where $L[f]$ is a literal containing the function symbol f which is in a semi-ground form, $f(a_1, \cdots, a_k, x_{k+1}, \cdots, x_n, y_1, \cdots, y_l)$, C_1' is a subclause of C_1 and C_1' has shared variables $x_{k+1}, \cdots, x_{n+m}$ with $L[f]$. If $b \equiv f(a_1, a_2, \cdots, a_n, c_1, \cdots, c_l)$ for any constants $a_{k+1}, a_{k+2}, \cdots, a_n, c_1, \cdots, c_l$, we may then infer

$$C = L\sigma[b] \vee C_1'\sigma$$

where $\sigma = \{x_{k+1}/a_{k+1}, x_{k+2}/a_{k+2}, \cdots, x_n/a_n, y_1/c_1, \cdots, y_l/c_l\}$ is a substitution. Therefore the above formula is equivalent to the following:

$$C = L\sigma[b] \vee C_1'(a_{k+1}, \cdots, a_n, x_{n+1}, \cdots, x_{n+m})$$

The inferred clause C is called a *variable FE-resolvent of* C_1. □

A variable FE-resolution will generate a ground instance of the function and then a semi-ground instance of the original clause will be logically implied by the original clause. Finally we state the completeness result in the following theorem.

Theorem 4.2 (Completeness of FE-Resolution) FE-resolution together with paramodulation and resolution are complete, i.e., an empty clause □ will be derived by using the standard resolution, paramodulation and FE-resolutions from a given set of clauses S, if S is M-unsatisfiable. □

Proof

By Theorem 4.1 we know that S is $\mathcal{M}$-unsatisfiable if and only if $S \bigcup K_{=}(S) \bigcup \Sigma^0_{\mathcal{M}}(S)$ is unsatisfiable. By Theorem 3.2 $S \bigcup K_{=}(S) \bigcup \Sigma^0_{\mathcal{M}}(S)$ is unsatisfiable if and only if $S \bigcup \Sigma^0_{\mathcal{M}}(S)$ is E-unsatisfiable. Furthermore, by the completeness of the paramodulation we have $S \bigcup \Sigma^0_{\mathcal{M}}(S) \vdash_{\sigma_{rp}} \square$ and therefore there exits a deduction of resolution and paramodulation of the empty $\square$ clause from $S \bigcup \Sigma^0_{\mathcal{M}}(S)$, denoted by $D[S \bigcup \Sigma^0_{\mathcal{M}}(S) \vdash_{\sigma_{rp}} \square]$.

Now we need to show $S \vdash_{\sigma_{fe}} \square$, that is, $\square$ is deducible from S by using the deduction rules from σ_{fe}. It is enough to show the existence of a deduction $D[S \vdash_{\sigma_{fe}} \square]$. It will be seen that this deduction can be obtained directly from the existing deduction, $D[S \bigcup \Sigma^0_{\mathcal{M}}(S) \vdash_{\sigma_{rp}} \square]$ and that the changes of the base clause set from $S \bigcup \Sigma^0_{\mathcal{M}}(S)$ to S will be compensated for by the additional syntactic inference rules of function evaluations.

According to our earlier definition, $\Sigma^0_{\mathcal{M}}(S)$ consists of two set of clauses which specify: (1) functions, i.e., in the form of $\Phi(\mathbf{x_i}, \mathbf{u_i}) = \mathbf{x_j}, \eta(\mathbf{x_i}) = \mathbf{y_i}, +_{k(N)}(\mathbf{I}, \mathbf{I'}) = \mathbf{I''}$ and $-_{k(N)}(\mathbf{I}, \mathbf{I'}) = \mathbf{I''}$ and (2) variable size clauses which specify the facts $x_1 \neq x_2, x_1 \neq x_3, \cdots$. We need only consider to convert the first set of clauses to FE-resolution since the set (2) is implied by the use of function evaluation and function definition.

Consider a clause $Eq(f(a,b),c)$ in $\Sigma^o_{\mathcal{M}}(S)$ (the same is true for unary-function $Eq(f(a), b)$, a state output function in $\Sigma^0_{\mathcal{M}}(S)$). If this clause, denoted C_1 is one of the parent clauses in the given deduction process, the other parent clause must be the form of $C_2 = \neg Eq(f(x,u),y) \wedge C_2'(x,u,y)$, where $C_2'(x,u,y)$ is a subclause of C_2. In this deduction the letters x, u, y will either take the variable -type values- which we denote x, u, y or they take constant values which in this proof are denoted a, b, c, i.e., we are dealing with variable or semi-variable FE-resolution as defined in Definitions 4.6 and 4.5. Taking the element **x** to have the generic variable letter form x or the generic constant letter form **a**, we get the first line of the following table where eight cases need to be considered:

Var	*Cases*							
x	a	a	x	a	x	x	a	x
u	b	b	b	x	b	y	x	y
y	c	y	c	c	y	c	y	z

In the following, we show this case by case:

Case 1. If $C_2 = \neg Eq(f(a,b),c) \vee C_2'(a,b,c)$ then the resolvent produced by the use of the resolution principle to this and the clause $Eq(f(a,b),c)$ in $\Sigma^o_{\mathcal{M}}(S)$ is $C = C_2'(a,b,c)$.

Consider the application of function evaluation to $f(a,b)$ in C_2, we shall have $\neg Eq(c,c) \vee C_2'(a,b,c)$ if $c \equiv f(a,b)$. This clause can then be further resolved with $Eq(x,x)$ to the final resolvent $C' = C_2'(a,b,c) = C$. The case $c \neq c' \equiv f(a,b)$ will not occur since function evaluation will be consistent with the axiom of $Eq(f(a,b),c)$.

Case 2. If $C_2 = \neg Eq(f(a,b),y) \vee C_2'(a,b,y)$, then the resolvent is $C = C_2'\sigma(a,b,y\sigma)$, where σ is a substitution defined as $\sigma = \{c/y\}$.

Consider the application of function evaluation to $f(a,b)$ in C_2, again we get $\neg Eq(c,y) \vee C_2'(a,b,y)$ if $c \equiv f(a,b)$. Furthermore, we may resolve this with $Eq(x,x)$, and we have the final resolvent $C' = C_2\sigma(a,b,y\sigma)$ where $\sigma = \{c/x, c/y\}$. Obviously $C' = C$.

Case 3. If $C_2 = \neg Eq(f(x,b),c) \vee C_2'(x,b,c)$ then the resolvent is $C = C_2'\sigma(x\sigma,b,c)$, with $\sigma = \{a/x\}$.

Consider the application of function evaluation to $c \equiv f(x,b)$. We shall find some a' such that $c \equiv f(a',b)$ and define $\sigma = \{a'/x\}$. Now this function evaluation will lead us to $\neg Eq(c,c) \vee C_2'\sigma(x\sigma,b,c)$. Again resolve this with $Eq(x,x)$ we get $C' = C_2'\sigma(x\sigma,b,c) = C$.

Case 4. $C_2 = \neg Eq(f(a,x),c) \vee C_2'(a,x,c)$ is treated in a similar manner as in Case 3.

Case 5. If $C_2 = \neg Eq(f(x,b),y) \vee C_2'(x,b,y)$ then the resolvent is $C = C_2'\sigma(x\sigma,b,y\sigma)$, with $\sigma = \{a/x, c/y\}$.

Consider the application of function evaluation to $y \equiv f(x,b)$ we shall find some a', c' such that $c' \equiv f(a',b)$ and define $\sigma = \{a'/x, c'/y\}$. Now this function evaluation will result $\neg Eq(c',c') \vee C_2'\sigma(x\sigma,b,y\sigma)$. Further resolve this with $Eq(x,x)$ and get $C' = C_2'\sigma(x\sigma,b,y\sigma) = C$.

Case 6. $C_2 = \neg Eq(f(x,y),c) \vee C_2'(x,y,c)$ is treated in a similar manner as in Case 5.

Case 7. $C_2 = \neg Eq(f(a,x),y) \vee C_2'(a,x,y)$ is treated in a similar manner as in Case 5.

Case 8. If $C_2 = \neg Eq(f(x,y),z) \bigvee C_2'(x,y,z)$ then the resolvent is $C = C_2'\sigma(x\sigma, y\sigma, z\sigma)$ with $\sigma = \{a/x, b/y, c/z\}$.

An application of function evaluation to $z \equiv f(x,y)$ may lead us to any a', b', c' such that $c' \equiv f(a',b')$, but not necessarily with $a' = a$ and $b' = b$ and $c' = c$.

This completes our case by case analysis and hence we can conclude that the existence of a deduction of the form $D[S \bigcup \Sigma^0_{\mathcal{M}}(S) \vdash_{\sigma_{rp}} \square]$ implies the existence of a deduction of the form $D[S \vdash_{\sigma_{fe}} \square]$ and hence we conclude that FE-resolution is complete.

□

In cases, except in Cases 1 and 2, trial and error steps are inevitable in order to get a substitution identical to the resolvent produced by the resolution and paramodulation.

5 Experiments

In the first page of the appendix we show a seven state machine its controlled state transitions. On the three succeeding pages we display the results of three automatic theorem proving experiments in a version of COCOLOG where the reachability predicate Rbl(x,y,l) denotes reachability of y from x in less than or equal to l steps. The basic software used was the GTP automatic theorem proving software of [12] extended with the FE-resolution facility by Q-X. Yu. The seven state machine experiment was devised by Y-J. Wei. The print out of the ATP excercises on pages 33,34, and 35 displays the successful resolution proofs of the predicates Rbl(x1,x7,3), Rbl(x2,x4,4), and Rbl(x,y,5), with x and y free, respectively. The total computing time in minutes was 0.06, 0.63 and 32.04, respectively and the corresponding number of resolutions was 289, 1366 and 59,017.

6 Conclusion

We have presented a function evaluation based resolution methodology, called FE-resolution, for the sets of conditional observer and controller logics referred to by the acronym (COCOLOG). The unique model property of COCOLOG is established first and then we derive the results of complete axiomatization and

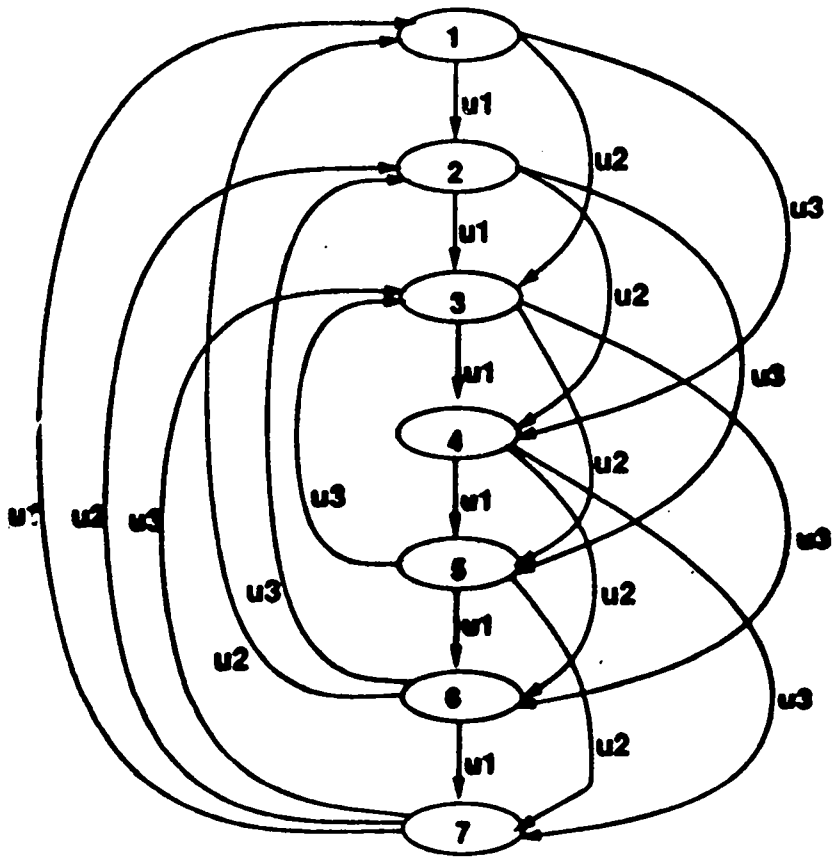
decidability properties of the COCOLOG theorems. Completeness of the FE-resolution technique is achieved through the relative truthfulness and validity of the COCOLOG models. Complexity reduction and experiments on theorem proving by use of FE-resolution are continuing at the present time.

7 Acknowledgements

Work partially supported by NSERC grant A1329 and INRIA Sophia-Antipolis, France. The authors would like to express their thanks to Q-X. Yu and Y-J. Wei for their assistance with this work.

Appendix

7–State Machine



```
Theorem: control21.thm +m +s +t -l +e +z?

 1: A Rbl(x,x,0)
 2: A Rbl(x,z,1) | ~Eq(phi7(x,y),z)
 3: A ~Rbl(z,u,y) | Rbl(v,u,x) | ~Eq(minus7(x,1),y) | ~Eq(phi7(v,w),z)
 4: A ~Rbl(z,u,x) | Rbl(z,u,y) | ~Eq(x,minus7(y,1))
 5: A Eq(x,x)
 6: A ~Eq(x,y) | Eq(y,x)
 7: A ~Eq(x,y) | ~Eq(y,z) | Eq(x,z)
 8: A ~Eq(x,y) | Eq(phi7(x,z),phi7(y,z))
 9: A ~Eq(x,y) | Eq(phi7(z,x),phi7(z,y))
10: A ~Eq(x,y) | Eq(Mu(x),Mu(y))
11: A ~Eq(x,y) | Eq(minus7(x,1),minus7(y,1))
12: A ~Eq(x,y) | Eq(minus7(1,x),minus7(1,y))
13: T ~Rbl(x1,x7,3)
14: (13a,4b)  [--] ~Rbl(x1,x7,x) | ~Eq(x,2)
15: (14b,5a)  [--] ~Rbl(x1,x7,2)
16: (15a,3b)  [--] ~Rbl(x,x7,y) | ~Eq(1,y) | ~Eq(phi7(x1,z),x)
17: (16a,2a)  [--] ~Eq(1,1) | ~Eq(phi7(x1,x),y) | ~Eq(phi7(y,z),x7)
18: (17c,5a)  [--] ~Eq(1,1) | ~Eq(phi7(x1,x),x4)
19: (18b,5a)  [--] ~Eq(1,1)
20: (19a,5a)  [--] []

AUM: 1061      SUN: 518       UNR: 48%
FAC: 0         RES: 289       MOD: 111       ABF: 4
TAU: 0         MER: 0         IDN: 27        TNT: 150
LIT: 3         LNT: 51        TIM:  0.03     LVL: 4

  1:  0.00       2:  0.00       3:  0.03       4:  0.03
```

```
Theorem: control21.thm +m +r +t -l +e +z3

  1: A Rbl(x,x,0)
  2: A Rbl(x,z,1) | ~Eq(phi7(x,y),z)
  3: A ~Rbl(z,u,y) | Rbl(v,u,x) | ~Eq(minus7(x,1),y) | ~Eq(phi7(v,w),z)
  4: A ~Rbl(z,u,x) | Rbl(z,u,y) | ~Eq(x,minus7(y,1))
  5: A Eq(x,x)
  6: A ~Eq(x,y) | Eq(y,x)
  7: A ~Eq(x,y) | ~Eq(y,z) | Eq(x,z)
  8: A ~Eq(x,y) | Eq(phi7(x,z),phi7(y,z))
  9: A ~Eq(x,y) | Eq(phi7(z,x),phi7(z,y))
 10: A ~Eq(x,y) | Eq(Mu(x),Mu(y))
 11: A ~Eq(x,y) | Eq(minus7(x,1),minus7(y,1))
 12: A ~Eq(x,y) | Eq(minus7(1,x),minus7(1,y))
 13: T ~Rbl(x2,x4,4)
 14: (13a,4b)  [--] ~Rbl(x2,x4,x) | ~Eq(x,3)
 15: (14b,5a)  [--] ~Rbl(x2,x4,3)
 16: (15a,4b)  [--] ~Rbl(x2,x4,x) | ~Eq(x,2)
 17: (16a,4b)  [--] ~Rbl(x2,x4,x) | ~Eq(y,2) | ~Eq(x,minus7(y,1))
 18: (17a,2a)  [--] ~Eq(x,2) | ~Eq(1,minus7(x,1)) | ~Eq(phi7(x2,y),x4)
 19: (18c,5a)  [--] ~Eq(x,2) | ~Eq(1,minus7(x,1))
 20: (19b,5a)  [--] ~Eq(2,2)
 21: (20a,5a)  [--] []

AGN: 6480      SUN: 3228      UNR: 49%
FAC: 0         RES: 1366      MOD: 524       ABF: 5
TAU: 0         MER: 0         IDN: 171       THT: 1364
LIT: 3         LHT: 326       TIM:  0.30     LVL: 5

  1:  0.00       2:  0.00       3:  0.03       4:  0.30       5:  0.30
```

```
Theorem: control2[illegible] +m +s +t -l +e +z3

 1: A Rbl(x,x,0)
 2: A Rbl(x,x,1) | ~Eq(phi7(x,y),z)
 3: A ~Rbl(z,u,y) | Rbl(v,u,x) | ~Eq(minus7(x,1),y) | ~Eq(phi7(v,w),z)
 4: A ~Rbl(z,u,x) | Rbl(z,u,y) | ~Eq(x,minus7(y,1))
 5: A Eq(x,x)
 6: A ~Eq(x,y) | Eq(y,x)
 7: A ~Eq(x,y) | ~Eq(y,z) | Eq(x,z)
 8: A ~Eq(x,y) | Eq(phi7(x,z),phi7(y,z))
 9: A ~Eq(x,y) | Eq(phi7(z,x),phi7(z,y))
10: A ~Eq(x,y) | Eq(Mu(x),Mu(y))
11: A ~Eq(x,y) | Eq(minus7(x,1),minus7(y,1))
12: A ~Eq(x,y) | Eq(minus7(1,x),minus7(1,y))
13: T ~Rbl(x,y,5)
14: (13a,4b)  [--] ~Rbl(x,y,z) | ~Eq(z,4)
15: (14b,5a)  [--] ~Rbl(x,y,4)
16: (15a,4b)  [--] ~Rbl(x,y,z) | ~Eq(z,3)
17: (16b,5a)  [--] ~Rbl(x,y,3)
18: (17a,4b)  [--] ~Rbl(x,y,z) | ~Eq(z,2)
19: (18a,4b)  [--] ~Rbl(x,y,z) | ~Eq(u,2) | ~Eq(z,minus7(u,1))
20: (19a,2a)  [--] ~Eq(x,2) | ~Eq(1,minus7(x,1)) | ~Eq(phi7(y,z),u)
21: (20c,5a)  [--] ~Eq(x,2) | ~Eq(1,minus7(x,1))
22: (21b,5a)  [--] ~Eq(2,2)
23: (22a,5a)  [--] []

AUN: 207412     SUN: 116430     UNR: 56%
FAC: 0          RES: 59017      MOD: 31058      ABF: 3
TAU: 0          MER: 9          IDN: 4521       THT: 41818
LIT: 3          LHT: 11073      TIM: 16.16      LVL: 7

  1:  0.00        2:  0.00        3:  0.01        4:  0.13        5:  0.98
  6: 14.76        7: 16.16
```

References

[1] Wolfgang Bibel. Short proofs of the pigeonhole formulas based on the connection method. *Journal of Automated Reasoning*, pages 287–297, 6 1990.

[2] P. E. Caines and S. Wang. Cocolog:A conditional observer and controller logic for finite machines. In *Proceedings of the 1990 Conference on Decision and Control*, Honolulu, Hawaii, December, 1990.

[3] Peter E. Caines and Suning Wang. Classical and Logic-Based Regulators for Partially Observed Automata: Dynamic Programming Formulation. In *Proceeding of the 1989 Conference on Information Sciences and Systems*, John Hopkins University, Baltimore, MA, March 1989.

[4] Peter E. Caines, Suning Wang, and Russell Greiner. Dynamical Logic Observers for Finite Automata. In *Proceeding of the 1988 Conference on Information Sciences and Systems*, pages 50–56, Princeton University, Princeton NJ, March 1988.

[5] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973.

[6] Robert Goldblatt. *Logics of Time and Computation*. CSLI/Stanford, Stanford, CA, 1987.

[7] H. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[8] R. A. Kowalski. *Logic for Problem Solving*. North-Holland, New York, 1979.

[9] John Wylie Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, Germany, 1984.

[10] Elliott Mendelson. *Introduction to Mathematical Logic*. Van Nostrand Reinhold Company, New York, N.Y. 10001, 1964.

[11] Karen L. Myers. *Universal Attachment: A Technique for Integrating Diverse Representation and Reasoning Methods*. PhD thesis, Stanford University, 1990.

[12] M. Newborn. *The Great Theorem Prover.* Newborn Software, P.O. Box 429, Victoria Station, Westmount, P.Q. Canada, 1989.

[13] G. A. Robinson and L. Wos. Paramodulation and theorem proving in first order theories with equality. *Machine Intelligence*, 4:95–99, 1969.

[14] J.A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

ISSN 0249-6399