REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE April 1997	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE
Scalable Data and Sensor Fusion Through Optimal Solutions of Multiple-Agent Systems

5. FUNDING NUMBERS
DAAH04-95-C-0011

6. AUTHOR(S)
Wolf Kohn, Jeffrey Remmel, John James, Anil Nerode

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Intermetrics, Inc. 23 Fourth Avenue, Burlington, MA 01803

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

ARO 34393.1-MA-SDI

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
We address the problem of finding an unbiased estimate of the battlefield state given that the data available is dynamic, noisy, and given in a multiplicity of representations. The approach proposed in the study is unique because it does not attempt to transform the data to a common representation. Rather we establish a framework which we call the Multiple Agent Hybrid Estimation Architecture (MAHEA) in which we allow heterogeneous data to flow between individual agents in the network to improve their individual estimates of the current plant state. For each MAHEA agent, we can construct a special optimization criterion for the battlefield estimation optimization problem which we call the Estimation Lagrangian. The type of Lagrangian that we need to construct is special for the plant estimation problem in that we want the Lagrangian to be 0 at each point where the agent has reached the desired estimate of the plant state. Such lagrangians are not suitable for most control problems. We also develop mechanisms to solve the problems of agent synchronization and how agents with different models can produce coherent (synchronous) and compatible (common view) estimates of the battlefield. Our solution to these problems uses the Noether invariance conditions in a novel way.

14. SUBJECT TERMS
Declarative Control
Agent Architectures
Sensor Fusion   Sensor Agents

15. NUMBER OF PAGES
35

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

19970515 171

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

DTIC QUALITY INSPECTED 4

Scalable Data and Sensor Fusion Through Optimal Solutions of Multiple-Agent Systems

FINAL TECHNICAL REPORT

Wolf Kohn
Jeffrey Remmel
John James
Anil Nerode

June 1996

U.S. Army Research Office

1 Introduction

The problem that we are addressing in this research is the following. Suppose we are given a suite of heterogeneous sensors and their controlling agents which are measuring the dynamics of a battlefield. For example, the agents may be providing radar samples, scout information, or satellite information. We assume that each sensor agent has a model of the battlefield dynamics relative to its own domain. Moreover, we assume that the sensor agents interact with each other in real time via a communications network. The fundamental problem is to find an unbiased estimate of the battlefield state given that the data available is dynamic, noisy, and given in a multiplicity of representations. The approach proposed in the study is unique because it does not attempt to transform the data to a common representation. Rather we establish a framework which we call the Multiple Agent Hybrid Estimation Architecture (MAHEA), see figure 1, in which we allow heterogeneous data to flow between individual agents in the network to improve their individual estimates of the current battlefield state.

The outline of this paper is as follows. We start by providing a brief description of a MAHEA agent, the basics of a MAHEA agent models of the battlefield and the procedure it uses to improve the its battlefield estimate over time. Basically each agent of MAHEA, formulates a relaxed variational optimization problem whose successful resolution produces an estimate of the battlefield. Each agent operates as a real-time theorem prover in the domain of relaxed variational theory [35]. In section 2, we shall give more details on the formulation of the relaxed variational optimization problem and how the agent solves this problem. In section 3, we provide a general description of how the agent theorem prover operates. We note that the operation of the agent described in sections 2 and 3 are essentially the same as the operation of an agent in our Multiple Agent Hybrid Control Architecture, see [24, 25], which can be used to control a variety of processes including automated manufacturing , multimedia networks, flexible gun tubes, and flight planning for missiles, and traffic management of highways.

The new results produced by our research on this project will be presented in sections 4 and 5. In section 4, we shall show how we can construct a special optimization criterion for the battlefield estimation optimization problem which we call the Estimation Lagrangian. The type of Lagrangian that we need to construct is special for the battlefield estimation problem in that we want the Lagrangian to be 0 at each point where the agent has reached the desired estimate of the battlefield. Such Lagrangians are not suitable for most control problems. Finally in section 5, we shall describe our mechanism to solve the problems of agent synchronization and how agents with different models can produce coherent (synchronous) and compatible (common view) estimates of the battlefield. Our solution to these problem uses the Noether invariance conditions in a novel way.
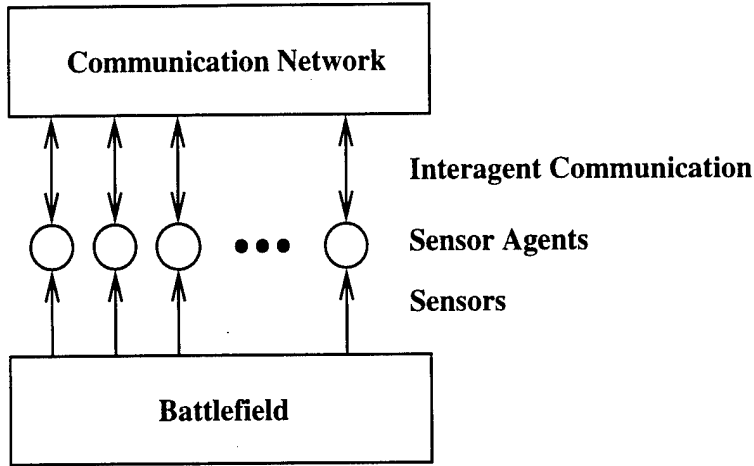
2

Framework

Figure 1: Agent Framework

2 The Multiple Agent Hybrid Estimation Architecture

In this section, we describe the main operational and functional characteristics an agent in a MAHEA network. As we mentioned in the introduction, our Multiple Agent Hybrid Estimation Architecture is implemented as a distributed system composed of agents and a communication network which we call the logic communication network. The architecture realizing this system operates as an on-line distributed theorem prover. At any update time, each active agent generates estimation actions as side effects of proving an existentially quantified subtheorem (lemma) which encodes the model of the battlefield as viewed by the agent. The conjunction of lemmas at each instant of time, encodes the desired behavior of the entire network . Each agent of MAHEA, consists of five modules: a Planner, a Dynamic Knowledge Base, a Deductive Inferencer, an Adapter and a Knowledge Decoder. We briefly overview the functionality of an agent in terms of its modules .

The basic architecture of an estimation agent is pictured in figure 2. The agent consists of 5 modules with the following functionality:

1. **Planner** The Planner constructs and repairs the agent state estimation optimization criteria which we refer to as the Estimation Lagrangian associated with the agent. In particular, the Planner generates a statement representing the desired model of the estimation system as an existentially quantified logic expression herein referred to as the Estimation Statement.

3

2. **Inferencer** The Inferencer determines whether there is a state estimate for the agent's relaxed variational state estimation problem which is a near optimal solution where the agent's Estimation Lagrangian is used as a cost function. If there is such a solution, the agent infers a near optimal estimation and sends data to the other agents. Otherwise it infers failure terms and a new state for the agent and reports the failure to the other agents. In particular, the Inferencer determines whether the Estimation Statement is a theorem in the theory currently active in the knowledge base. If the Estimation Statement logically follows from the current status of the Knowledge Base, the inferencer generates, as a side effect of proving this Estimation Statement to be true, the current state estimate of the Battlefield. If the Estimation Statement does not logically follow from the current status of the Knowledge Base, that is, the desired behavior is not realizable, the inferencer transmits the failed terms to the Adapter module for replacement or modification.

3. **Adapter** The Adapter repairs failure terms and constructs correction terms.

4. **Knowledge Base** The Knowledge Base stores and updates the agent's battlefield model and constraints. The Knowledge Base also stores the requirements of operations or processes within the scope of the agent's estimation problem. It also encodes system constraints, inter-agent protocols and constraints, sensory data, operational and logic principles and a set of primitive inference operations defined in the domain of equational terms.

5. **Knowledge Decoder** The Knowledge Decoder receives and translates the other agent's data.

To better understand, how these five modules function, we first need to discuss the basic elements of an agent's model and how it behaves. We will discuss this model in the next section and we will then follow with a more detailed discussion of the five modules of a MAHEA agent.

2.1 An MAHEA Agent's Model

In general, a hybrid system has a hybrid state, the simultaneous dynamical state of all plants and digital control devices. Properly construed, the hybrid states will form a differentiable manifold which we call the *carrier manifold* of the system. To incorporate the digital states as certain coordinates of points of the carrier manifold, we "continualize" the digital states. That is, we view the digital states as finite, real-valued, piecewise-constant functions of continuous time and then we take smooth approximations to them. This also allows us to consider logical and differential or variational constraints on the same footing, each restricting the points allowed on the carrier manifold. In fact, all logical or discontinuous features can be continualized without practical side-effects.
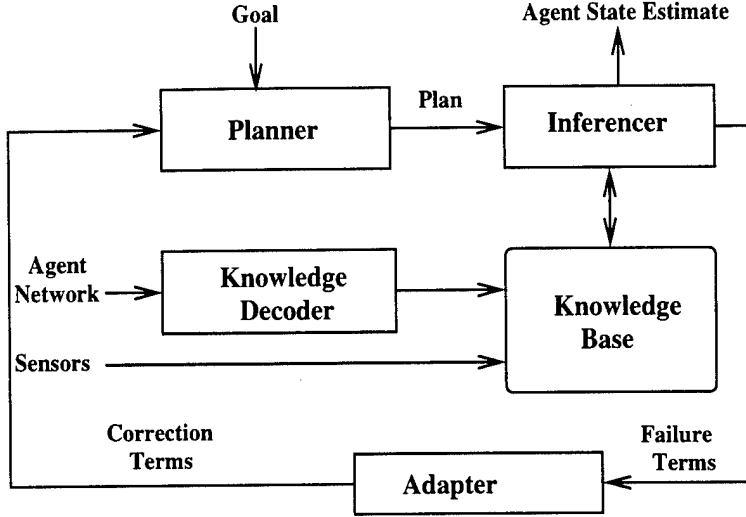
4

MAHEA Sensor Agent

Figure 2: MAHEA Agent Architecture

This is physically correct since for any semiconductor chip used in an analog device, the zeros and ones are really just abbreviations for sensor readings of the continuous state of the chip. Every constraint of the system, digital or continuous, is incorporated into the definition of what points are on the carrier manifold. Lagrange constraints are regarded as part of the definition of the manifold as well, being restrictions on what points are on the manifold.

More specifically, let A_i, $i = 1, \ldots, N(t)$ denote the agents active at the current time t. In our model, t takes values on the real line. At each time t, the status of each agent in the network is given by a point in a locally differentiable manifold M [23]. The Estimation Lagrangian L_i of an active agent A_i is given by a continuous function,

$$L_i : M \times T \rightarrow R^+ \tag{1}$$

where T is the real line (time space) and R^+ is the positive real line. A point p in the manifold M is represented by a data structure of the form:

$$p(id, proc(proc\_data), estimate(estimation\_data), in(synch\_data), mp(mult\_data)) \tag{2}$$

Here id is an identifier taking values in a finite set ID, $proc()$ is a relation characterizing battlefield processes status which depends on a list of parameters labeled $proc\_data$, whose parameters define the operational, load, and timing characteristics of the process involved. The relation $estimate$ captures attributes of the battlefield being represented which depends on a list of parameters labeled $estimation\_data$ whose parameters, among other things, characterize various constraints of the battlefield representation of an agent at a level of abstraction

5

compatible with the logic communication network. The relation $in()$ carries synchronization information of the logic communication network. This includes information such as priority level, connectivity and time constants. Finally, the relation $mp()$ carries multiplicity information, that is, it represents the level of network usability at this point. The associated parameter list, $mult\_data$, is composed of statistical parameters reflecting the logic network's load.

From an agent's point of view, the dynamics of the battlefield is characterized by certain trajectories on the manifold M. These trajectories represent the agent estimate of the state of the battlefield plus the flow of information through the network and its status. Specifically, we need to define two items:

(i) The Estimation Lagrangian functions :

$$\{L_i(p, t) : i \in I(t)\} \tag{3}$$

where $I(t)$ is the set of active agents at time t and

(ii) the actions or estimates issued by the agents.

We will see shortly that these actions are implemented as infinitesimal transformations defined in M. The general structure of an Estimation Lagrangian function in (3) for an active agent i at time t is given in (4) below:

$$L_i(p, t) = F_i(U_i, L, \alpha_i)(p, t) \tag{4}$$

where F_i is a smooth function, L is the vector of Estimation Lagrangian functions, U_i is the state estimation error function, and α_i is the command action issued by the i-th agent. We will devote the rest of this subsection to characterizing this model.

We start with a discussion of the main characteristics of the manifold M. In general a manifold M is a topological space (with topology Θ) composed of three items:

(a) A set of points of the form of (2).

(b) A countable family of open subsets of M, U_i such that

$$\bigcup_i U_i = M.$$

(c) A family of smooth homeomorphisms,$\{\phi_i : \phi_i : U_i \to V_i\}$, where for each j, V_j is an open set in R^k. The sets U_i are referred to in the literature as coordinate neighborhoods or charts. For each chart U_j the corresponding function ϕ_i is referred to as its coordinate chart.

The coordinate chart functions satisfy the following additional condition:

Given any charts U_i and U_j such that $U_i \cap U_j \neq \emptyset$, the function $\phi_i \circ \phi_j^{-1} : \phi_j(U_i \cap U_j) \to \phi_j(U_i \cap U_j)$ is smooth.

In the literature, one usually finds an additional property, which is the Hausdorff property in the definition of manifolds [24]. Since this property does not hold in our application we will not discuss it.

Now we proceed to customize the generic definition of the manifold to our application. We start with the topology Θ associated with M. We note that the points of M have a definite structure, see (2), whose structure is characterized by the values, or more precisely by intervals of values, of the parameters in the lists *proc_data*, *estimation_data*, *synch_data* and *mult_data*. The number of these parameters equals k. The knowledge about these parameters is incorporated into the model by defining a finite topology Ω on R^k [4].

The open sets in Ω are constructed from the clauses encoding what we know about the parameters. The topology Θ of M is defined in terms of Ω as follows: For each open set W in Ω such that $W \subseteq V_j \subseteq R^k$, we require that the set $\phi_j^{-1}(W)$ be in Θ. The sets constructed in this way form a basis for Θ so that a set $U \subseteq M$ is open if and only if for each $p \in U$, there is j and an open set $W \in \Omega$ such that $W \subseteq V_j$ and $p \in \phi_j^{-1}(W)$.

To characterize the actions commanded by a MAHEA agent we need to introduce the concept of derivations on M. Let F_p be the space of real valued smooth functions f defined in a neighborhood a point p in M. Let f and g be functions in F_p. A *derivation* v of F_p is a map

$$v : F_p \to F_p$$

that satisfies the following two properties:

$$v(f + g)(p) = (v(f) + v(g))(p) \qquad \text{(Linearity)} \qquad (5)$$

$$v(f \cdot g)(p) = (v(f) \cdot g + f \cdot v(g))(p) \qquad (\text{ Leibniz Rule}) \qquad (6)$$

Derivations define vector fields on M and a class of associated curves called integral curves [25]. Suppose that C is a smooth curve on M parameterized by $\psi : I \to M$ where I a subinterval of R. In local coordinates, $p = (p^1, ..., p^k)$, C is given by k smooth functions $\psi(t) = (\psi^1(t), \ldots, \psi^k(t))$ whose derivative with respect to t is denoted by $\dot{\psi}(t) = (\dot{\psi}^1(t), \ldots, \dot{\psi}^k(t))$. We introduce an equivalence relation on curves in M as the basis of the definition of tangent vectors at a point in M [13]. Two curves $\psi_1(t)$ and $\psi_2(t)$ passing through a point p are said to be equivalent at p (notation: $\psi_1(t) \sim \psi_2(t)$), if there exists $\tau_1, \tau_2 \in I$ such that

$$\psi_1(\tau_1) = \psi_2(\tau_2) = p \qquad (7)$$

$$\dot{\psi}_1(\tau_1) = \dot{\psi}_2(\tau_2). \qquad (8)$$

Clearly, $\sim$ defines an equivalence relation on the class of curves in M passing through p. Let $[\psi]$ be the equivalence class containing ψ. A *tangent vector* to $[\psi]$ is a derivation, $v|_p$, such that in local coordinates $(p^1, \ldots, p^k)$, it satisfies the condition that given any smooth function $f : M \to R$,

$$v|_p(f)(p) = \sum_{j=0}^{k} \psi^j(t) \frac{\partial f(p)}{\partial p^j} \qquad (9)$$

7

where $p = \psi(t)$. The set of tangent vectors associated with all the equivalence classes at p defines a vector space called the *tangent vector space* at p, denoted by TM_p. The set of tangent spaces associated with points in M can be "glued" together to form a manifold called the *tangent bundle* which is denoted by TM.

$$TM = \bigcup_{p \in m} TM_p$$

For our purposes, it is important to specify explicitly how this gluing is implemented. This will be explained below after we introduce the concept of a vector field and discuss its relevance in the model.

A *vector field* on M is an assignment of a derivation $v|_p$ to each point p of M which varies smoothly from point to point. That is, if $p = (p^1, ..., p^k)$ are local coordinates, then we can always write $v|_p$ in the form

$$v|_p = \sum_j \lambda^j(p) \frac{\partial}{\partial p^j} \tag{10}$$

Then v is a vector field if the coordinate functions λ_i are smooth.

Comparing (9) and (10) we see that if ψ is a parameterized curve in M whose tangent vector at any point coincides with the value of v at a point $p = \psi(t)$, then in the local coordinates $p = (\psi^1(t), \ldots, \psi^k(t))$, we must have

$$\dot{\psi}^j(t) = \lambda^j(p) \text{ for } j = 1, \ldots, k. \tag{11}$$

In our application, each command issued by the MAHEA agent is implemented as a vector field in M. Each agent constructs its command field as a combination of 'primitive' predefined vector fields. Since the chosen topology for M, Θ, is not metrizable, we cannot guarantee a unique solution to (11) in the classical sense for a given initial condition. However, they have solutions in a class of continuous trajectories in M called *relaxed curves* [29]. In this class, the solutions to (11) are unique. We discuss the basic characteristics of relaxed curves as they apply to our process control formulation and implementation in Section 3. Next, we describe some of their properties as they relate to our battlefield model and estimation process. For this objective, we need to introduce the concept of flows in M.

If v is a vector field, any parameterized curve passing through a point p in M is called an *integral curve associated with v*, if in local coordinates (5) holds. An integral curve associated with a field v, denoted by $\Psi(t, p)$ is termed the flow generated by v if it satisfies the following properties:

$$\Psi(t, \Psi(\tau, p)) = \Psi(t + \tau, p) \quad \text{(semigroup property)} \tag{12}$$
$$\Psi(0, p) = p \quad \text{(initial condition)}$$
$$\text{and}$$
$$\frac{d}{dt}\Psi(t, p) = v \mid_{\Psi(t,p)} \quad \text{(flow generation)}$$
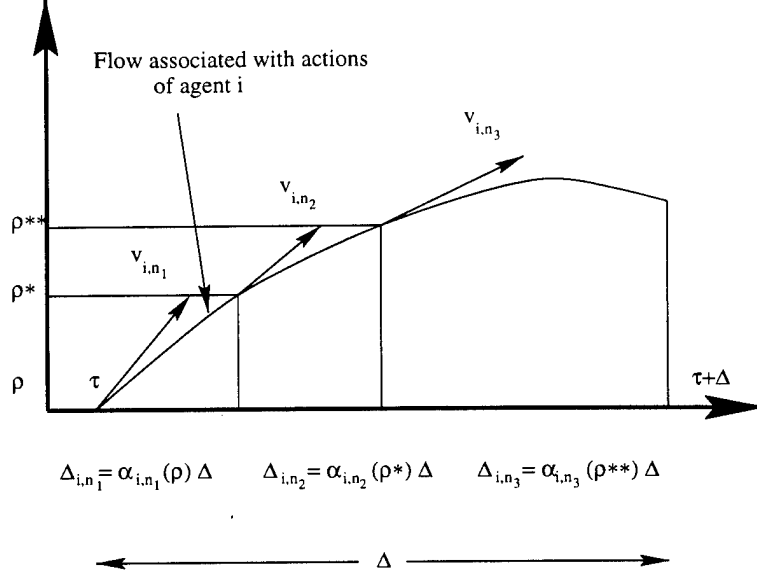
8

Figure 3: Conceptual illustration of agent action schedule

Now we are ready to customize these concepts for our model. Suppose that a MAHEA agent A_i is active. Let $\Delta > 0$ be the width of the current decision interval, $[t, t + \Delta)$. Let $U_i(p, t)$ be the state estimation error at the beginning of the interval. Agent A_i has a set of primitive actions:

$$\{v_{i,j} : j = 1, \ldots, n_i \text{ where } v_{i,j}|_p \in TM_p \text{ for each } p \in m\} \tag{13}$$

During the interval $[t, t + \Delta)$, agent A_i schedules one or more of these actions to produce a flow which will reduce the state estimation error. In particular, A_i determines the fraction $\alpha_{i,j}(p, t)$ of Δ that action $v_{i,j}$ must be executed as a function of the current estimation requests $S_{r,i}(t, p)$ and the vector of Estimation Lagrangians $L(p, t) = (L_1(p, t), \ldots, L_{N(t)})$ of the active agents in the MAHEA network. Figure 3 conceptually illustrates a schedule of actions involving three primitives. We will use this example as means for describing the derivation of our model. The general case is similar.

The flow Ψ_i associated with the schedule of Figure 3 can be computed from the flows associated with each of the actions:

$$\Psi_i(\tau, p) = \begin{cases} \Psi_{v_{i,n_1}}(\tau, p) \text{ if } t \leq \tau \leq t + \Delta_{i,n_1} \\ \Psi_{v_{i,n_2}}(\tau, \Psi_{v_{i,n_1}}(\tau, p)) \text{ if } t + \Delta_{i,n_1} \leq \tau \leq t + \Delta_{i,n_1} + \Delta_{i,n_2} \\ \Psi_{v_{i,n_3}}(\tau, \Psi_{v_{i,n_2}}(\tau, \Psi_{v_{i,n_1}}(\tau, p))) \\ \qquad \text{if } t + \Delta_{i,n_1} + \Delta_{i,n_2} \leq \tau \leq t + \Delta_{i,n_1} + \Delta_{i,n_2} + \Delta_{i,n_3} \end{cases} \tag{14}$$

where $\Delta = \Delta_{i,n_1} + \Delta_{i,n_2} + \Delta_{i,n_3}$ and $\alpha_{i,n_1} + \alpha_{i,n_2} + \alpha_{i,n_3} = 1$. We note that the flow Ψ_i given by (14) characterizes the evolution of the process as viewed

9

by agent A_i. The vector field $v_i|_p$ associated with the flow Ψ_i is obtained by differentiation and the third identity in (12). This vector field applied at p is proportional to

$$v_i|_p = [v_{i,n_1}, [v_{i,n_2}, v_{i,n_3}]] \tag{15}$$

where $[.,.]$ is the Lie bracket due to the parallelogram law, see [38]. The Lie bracket is defined as follows: Let v and w, be derivations on M and let $f : M \rightarrow R$ be any real valued smooth functions. The Lie bracket of v and w is the derivation defined by

$$[v, w](f) = v(w(f)) - w(v(f)),$$

see [9].

Thus the composite action $v_i|_p$ generated by the i-th agent to improve the state estimation is a composition of the form of (15). Moreover from a version of the Chattering lemma and duality [18], we can show that this action can be expressed as a linear combination of the primitive actions available to the agent as follows.

$$[v_{i,n_1}, [v_{i,n_2}, v_{i,n_3}]] = \sum_j \gamma_j^i(\alpha) v_{i,j} \tag{16}$$

$$\sum_j \gamma_j^i(\alpha) = 1$$

with the coefficients γ_j^i determined by the fraction of time that each primitive action $v_{i,j}$ is used by agent i.

The effect of the field defined by the composite action $v_i|_p$ on any smooth function (equivalent function class) is computed by expanding the right hand side of (14) in a Lie-Taylor series [9]. In particular, we can express the change in the state estimation error U_i due to the flow over the interval Δ in terms of $v_i|_p$. The evolution of the state estimation error U_i over the interval starting at point p is given by

$$U_i(t + \Delta, p'') = U_i(t, \Psi_i(t + \Delta, p)) \tag{17}$$

Expanding the right hand side of (17) in a Lie-Taylor series around (t, p), we obtain,

$$U_i(t + \Delta, p'') = \sum_j \frac{\left(v_i|_p(U_i(p, t))\right)^j \Delta^j}{j!}$$

where

$$\left(v_i|_p(\cdot)\right)^j = v_i|_p\left(\left(v_i|_p(\cdot)\right)^{-1}j\right) \tag{18}$$

and

$$\left(v_i|_p\right)^0(f) = f. \text{ for all } f$$

In general, the series in the right hand side of (18) will have countable many non-zero terms. In our case, since the topology of M is finite because it is

10

generated by finitely many logic clauses, this series will have only finitely many non-zero terms. Intuitively, this is so because in computing powers of derivations (i.e., limits of differences), we need only to distinguish among different neighboring points. In our formulation of the topology of M, this can only be imposed by the information in the clauses of the agent's Knowledge Base. Since each agent's knowledge base has only finitely many clauses, there is a term in the expansion of the series in which the power of the derivation goes to zero. This is important because it allows the series in the right hand side to be effectively generated by a locally finite automaton. We will expand on the construction of this automaton in the next section when we discuss the inference procedure carried out by each agent.

We note that given the set of primitive actions available to each agent, the composite action is determined by the vector of fraction functions α_j . We will see in the next section that this vector is inferred by each agent from the proof of existence of solutions of an optimization problem.

Now we can write the specific nature of the model formulated in expression (4). At time t and at point $p \in M$ the estimation error function of agent i is given by :

$$U_i(p,t) = U_i(p,t^-) + S_{r,i}(p,t) + \sum_k Q_{i,k} L_k(p,t^-) \qquad (19)$$

where t^- is the end point of the previous update interval, $S_{r,i}$ is the estimation request function to agent i, and $Q_{i,k}$ is a multiplier determining the required degree of accuracy and the urgency of the estimate of the Agent k that Agent i requires. This allocation is determined from the characteristics of the process both agents are estimating and from the process description encoded in the agent's knowledge base. The actual request from agent k to agent i is thus the term, $Q_{i,k} L_k(p,t^-)$. The information sent to agent i by agent k is the state estimation function $L_k(p,t^-)$ at the end of the previous interval. Finally the point $p \in M$ carries the current estimate of the process monitored by the agents appearing in (19). Agent k thus contribute to Agent i's new estimate only if $Q_{i,k} \neq 0$.

This concludes our description of the model. For space considerations, some details have been left out. In particular those related to the strategy for activation and deactivation of agents. These will be discussed in a future paper.

3 The Five Modules

In each agent of MAHEA, the Estimation Statement is the formulation of a relaxed variational optimization problem whose successful resolution produces an action schedule of the form of (15). Each agent operates as a real-time theorem prover in the domain of relaxed variational theory [35]. A customized version of this theory, enriched with elements of differential geometry, equational logic and automata theory provides a general representation for the dynamics, constraints, requirements and logic of the Estimation Agent network . We devote the rest of this section to the discussion of the main elements of this theory in

the context of the operational features of MAHEA. The architecture is composed of two items: The Control Agent, and the logic communication Network. These items are illustrated in Figures 5 and 6 respectively. We will discuss them in the remaining of this section.

3.1 Architectural Elements of a control agent:

We will discuss next the functionality of the five modules of a control agent. These are: the Knowledge Base, the Planner, the Inferencer, the Knowledge Decoder and the Adapter.

3.1.1. Knowledge Base: The Knowledge Base consists of a set of equational first order logic clauses with second order extensions. The syntax of clauses is similar to the ones in the Prolog language. Each clause is of the form

$$Head \leftarrow Body \qquad (20)$$

where $Head$ is a functional form, $p(x_1, ...x_n)$, taking values in the binary set $[true, false]$ with $x_1, x_2, \ldots, x_n$ variables or parameters in the domain M of the MAHEA network. The symbol $\leftarrow$ stands for logical implication. The variables appearing in the clause head are assumed to be universally quantified. The $Body$ of a clause is a conjunction of one or more logical terms,

$$e_1 \wedge e_2 \wedge \ldots \wedge e_n \qquad (21)$$

where $\wedge$ is the logical 'and'. Each term in (21) is a relational form. A relational form is one of the following: an equational form, an inequational form, a covering form, or a clause head. The logical value of each of these forms is either true or false. A relational form e_i is true precisely at the set of tuples of values S_i of the domain taken by the variables where the relational form is satisfied and is false for the complement of that set. Thus for $e_i = e_i(x_1, \ldots, x_n)$, S_i is the possibly empty subset of M^n ,

$$S_i = \{(x_1, \ldots, x_n) \in M^n : e_i(x_1, \ldots, x_n) = true\}$$

so that
$$e_i(x_1, \ldots, x_n) = false \text{ if } (x_1, \ldots, x_n) \in M^n/S_i.$$

The generic structure of a relational form is given in Table 1.

Form	Structure	Meaning
equational	$w(x_1, \ldots, x_n) = v(x_1, \ldots, x_n)$	equal
inequational	$w(x_1, \ldots, x_n) \neq v(x_1, \ldots, x_n)$	not equal
covering	$w(x_1, \ldots, x_n) < v(x_1, \ldots, x_n)$	partial order
clause head	$q(x_1, \ldots, x_n)$	recursion,chaining

Table 1. Structure of the Relational Form

12

Declartive Estimation Agent

Boundary Estimation Agent
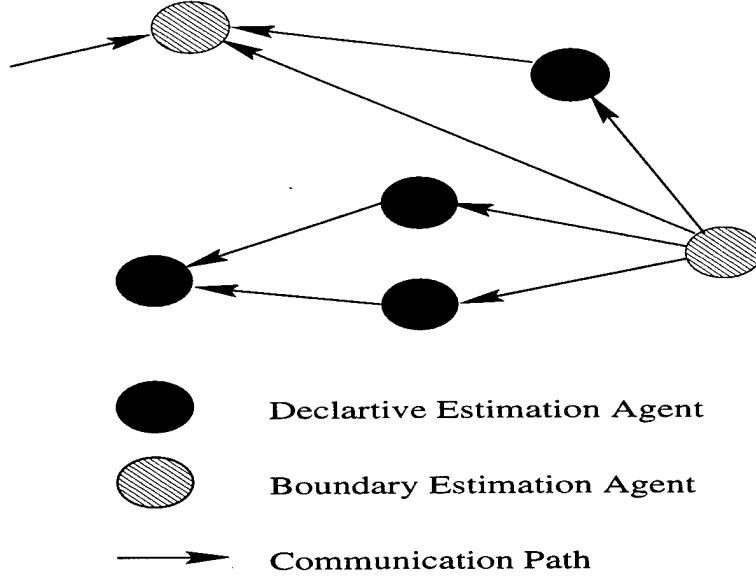
Communication Path

Figure 4: Network of Cooperating Control Agents

In Table 1, w and v are polynomic forms with respect to a finite set of operations whose definitional and property axioms are included in the Knowledge Base. A polynomic form v is an object of the form $v(x_1, \ldots, x_n) = \sum_{\omega \in \Omega} (v, \omega) \cdot \omega$ where Ω^* is the free monoid generated by the variable symbols $\{x_1, \ldots, x_n\}$ under juxtaposition. The term (v, ω) is called the coefficient of v at ω. The coefficients of a polynomic form v take values in the domain of definition of the clauses. The domain in which the variables in a clause head take values is the manifold M described in section 2. The logical interpretation of (20) and (21) is that the *Head* is true if the conjunction of the terms of *Body* are jointly true for instances of the variables in the clause head. M is contained in the Cartesian product :

$$M \subseteq G \times S \times X \times A \tag{22}$$

where G is the space of goals, S is the space of sensory data, X is the space of estimation states and A is the space of estimation actions. These were described in section 2. G, S, X, and A are manifolds themselves whose topological structure are defined by the specification clauses in the Knowledge Base (see figure 7). These clauses, which are application dependent, encode the requirements on the closed-loop behavior of the estimation model of the agent. In fact the closed loop behavior, which we will define later in this section in terms of a variational formulation, is characterized by continuous curves with values in M. This continuity condition is central because it is equivalent to requiring the system to look for actions that make the closed loop behavior satisfy the requirements of the battlefield model.

13

The denotational semantics of each clause in the knowledge base is one of the following:

1. a conservation principle,

2. an invariance principle, or

3. a constraint principle.

Conservation principles are one or more clauses about the balance of a particular process in the dynamics of the system or the computational resources. For instance, equation (19) encoded as a clause expresses the conservation of error for the Agent i's estimation of the battlefield.

$$consevation\_of\_error(p, t, [Q_{i,k}], S_{r,i}, [L_k], \Delta, U_i(t,p)) \rightarrow$$

$$U_i(t + 2\Delta, p'') = \sum_j \frac{(v_i|_p(U_i(t + \Delta, p')))\Delta^j}{j!} \ \wedge$$

/* encoding of equation (13) */

$$U_i(t + \Delta, p') = U_i(t,p) + S_{r,i}(t,p) + \sum_k Q_{i,k}L_k(t,p,\dot{p}) \ \wedge \qquad (23)$$

/* encoding of equation (15) */

$$process\_evolution(p, t, p'') \ \wedge \ /* \text{ encoding of equation (13) } */$$

$$consevation\_of\_error(p'', t + \Delta, [Q_{i,k}], S_{r,i}, [L_k], \Delta, U_i(t + 2\Delta, p''))$$

Conservation principles always involve recursion whose scope is not necessarily a single clause, as in the example above, but with chaining throughout several clauses.

Invariance principles are one or more clauses establishing constants of the evolution of agent's estimation error functions in a general sense. These principles include stationarity, which plays a pivotal role in the formulation of the theorems proved by the architecture, and geodesics. For example, in the state estimation of multimedia processes,invariant principles specify quality response requirements. That is, levels of performance as a function of traffic load that the system must satisfy. The importance of invariance principles lies in the reference they provide for the detection of unexpected events. For example, in the state estimation of a multimedia process, the update time after a request is serviced is constant, under normal operating conditions. An equational clause that states this invariance has a ground form that is constant; deviation from this value represents deviation from normality.

Constraint principles are clauses representing engineering limits to actuators or sensors and, most importantly, rules of engagement.

The clause database is organized in a nested hierarchical structure illustrated in Figure 5. The bottom of this hierarchy contains the equations that characterize the algebraic structure defining the terms of relational forms: an algebraic variety [40].
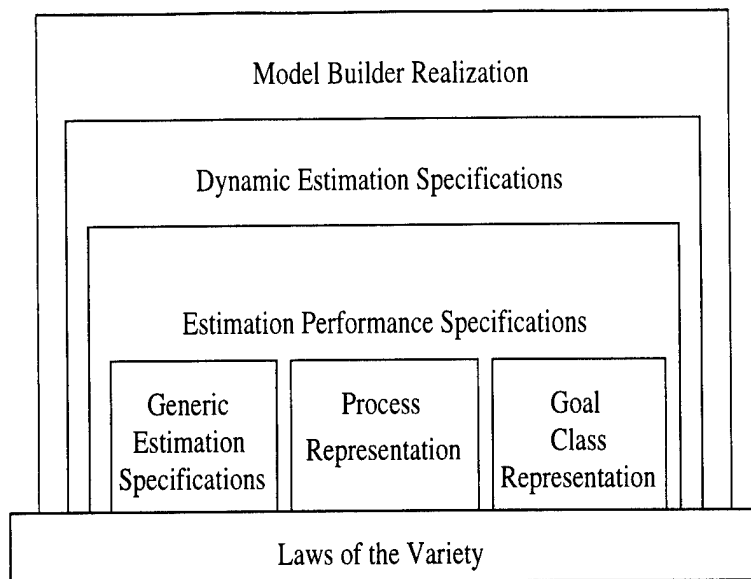
14

Figure 5: Knowledge Base Organization

At the next level of the hierarchy, three types of clauses are stored: Generic Estimation Specifications, Battlefield Representation and Goal Class Representation.

The Generic Estimation Specifications are clauses expressing general desired behavior of the system. They include statements about stability, complexity and robustness that are generic to the class of declarative rational controllers. These specifications are written by constructing clauses that combine laws of the kind which use the Horn clause format described earlier.

The Process Representation is given by clauses characterizing the dynamic behavior and structure of the battlefield, which includes sensors and actuators. These clauses are written as conservation principles for the dynamic behavior and as invariance principles for the structure. As for the Generic Estimation Specifications, they are constructed by combining a variety of laws in the equational Horn clause format.

The Goal Class Representation contains clauses characterizing sets of desirable operation points in the domain (points in the manifold M). For example, Goal Class clauses could specify the type and size of permitted state estimation errors. These clauses are expressed as soft constraints; that is, constraints that can be violated for finite intervals of time. They express the ultimate purpose of the controller but not its behavior over time.

The next level of the hierarchy involves the Estimation Performance Specifications. These are typically problem-dependent criteria and constraints. They are written in equational Horn clause format. They include generic constraints such as speed and time of response, and qualitative properties of state trajecto-

15

ries [35]. Dynamic Estimation Specifications are equational Horn clauses whose bodies are modified as a function of the sensor and goal commands.

Finally, Model Builder Realization clauses constitute a recipe for building a procedural model (an automaton) for generating variable instantiation (unification) and for theorem proving.

3.1.2. The Planner:

The function of the theorem Planner, which is domain-specific, is to generate, for each update interval, a symbolic statement of the desired behavior of the system, as viewed, say by the agent j, throughout the interval. The theorem statement that it generates has the following form:

Given a set of primitive actions there exists state estimation action schedule $v_i|_p$ of the form (16) and a fraction function differential $d\alpha(\cdot)$ (Figure 4) in the control interval $[t, t + \Delta)$ such that $d\alpha(\cdot)$ minimizes the functional

$$\int_t^{t+\Delta} L_i\big(\Psi_i(\tau, p), v_i|_p(G(_i(\tau, p)))\big) d\alpha(p, d\tau) \qquad (24)$$

subject to the following constraints:

$$g_i(S_i, \Psi_i(t + \Delta, p)) = G_i(t, X_i)$$
(local goal for the interval),

$$\sum_m Q_{i,m}(p, t) L_m(p, t) = V_i(p, t) \qquad (25)$$

(inter-agent constraint, see (19))

and

$$\int_t^{t+\Delta} d\alpha(p, d\tau) = 1$$

In (24), L_i is the *Estimation Lagrangian* of the system as viewed by Agent i for the current interval of control $[t, t + \Delta)$. This function, which maps the Cartesian product of the state and estimation action spaces into the real line with the topology defined by the clauses in the knowledge base, captures the dynamics, constraints and requirements of the system as viewed by agent i The *Local Estimation Lagrangian* function L_i is a continuous projection in the topology defined by the knowledge base (see [32]) in the coordinates of the i-th agent of the global Lagrangian function L that characterizes the system as a whole. In (25), p represents the state of the process under control as viewed by the agent and G_i is the parallel transport operator bringing the goal to the current interval. The operator G_i is constructed by lifting to the manifold the composite flow (see equation (14)). We note that the composite flow and the action schedule are determined once the fraction function is known and that this function is the result of the optimization (24), (25). In particular, the
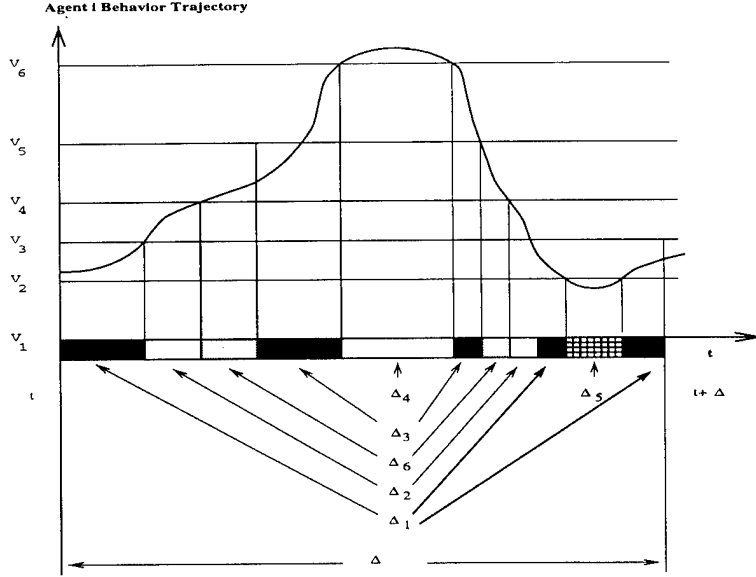
16

Figure 6: Illustration of optimization

action schedule is constructed as a linear combination of primitive actions (see equation (16)).

The term $d\alpha(\cdot)$ in (24) is a Radon probability measure [36] on the set of primitive estimation actions or derivations that the agent can execute for the interval $[t, t + \Delta)$. It measures, for the interval, the percentage of time to be spent in each of the primitive derivations. The central function of the control agent is to determine this mixture of actions for each control interval. This function is carried out by each agent by inferring from the current status of the knowledge base whether a solution of the optimization problem stated by the current theorem exists, and, if so, to generate corresponding actions and state updates. Figure 6 illustrates the relations between the primitive actions and the fraction of Δ they are active in the interval $[t, t + \Delta)$.

The expressions in (25) constitute the constraints imposed in the relaxed optimization problem solved by the agent. The first one is the local goal constraint expressing the general value of the state at the end of the current interval. The second represents the constraints imposed on the agent by the other agents in the network. Finally, the third one indicates that is a probability measure. Under relaxation and with the appropriate selection of the domain (see [20], the optimization problem stated in (24) and (25) is a convex optimization problem. This is important because it guarantees that if a solution exists, it is unique up to probability, and also, it guarantees the computational effectiveness of the inference method that the agent uses for proving the theorem.

The construction of the theorem statement given by (24) and (25) is the central task carried out in the Planner. It characterizes the desired behavior of the

17

process as viewed by the agent in the current interval so that its requirements are satisfied and the system "moves" towards its goal in an optimal manner.

3.1.3. Adapter:

The function under the integral in (24) includes a term, referred to as the "catch-all" potential, which is not associated with any clause in the Knowledge Base. Its function is to measure unmodeled dynamic events. This monitoring function is carried out by the Adapter which implements a generic commutator principle similar to the Lie bracket discussed in section 2, see (24). Under this principle, if the value of the catch-all potential is empty, the current theorem statement adequately models the status of the system. On the other hand, if the theorem fails, meaning that there is a mismatch between the current statement of the theorem and system status, the catch-all potential carries the equational terms of the theorem that caused the failure. These terms are negated and conjuncted together by the Inferencer according to the commutation principle (which is itself defined by equational clauses in the Knowledge Base) and stored in the Knowledge Base as an adaptation dynamic clause. The Adapter then generates a potential symbol, which is characterized by the adaptation clause and corresponding tuning constraints. This potential is added to criterion for the theorem characterizing the interval.

The new potential symbol and tuning constraints are sent to the Planner which generates a modified Estimation Lagrangian for the agent and goal constraint. The new theorem, thus constructed, represents adapted behavior of the system. This is the essence of reactive structural adaptation in the our model

At this point, we pause in our description to address the issue of robustness. To a large extent, the adapter mechanism of each controller agent provides the system with a generic and computationally effective means to recover from failures or unpredictable events. Theorem failures are symptoms of mismatches between what the agent thinks the system looks like and what it really looks like. The adaptation clause incorporates knowledge into the agent's Knowledge Base which represents a recovery strategy. The Inferencer, discussed next, effects this strategy as part of its normal operation.

3.1.4. Inferencer:

The Inferencer is an on-line equational theorem prover. The class of theorems it can prove are represented by statements of the form of (20) and (21), expressed by an existentially quantified conjunction of equational terms of the form:

$$\exists Z \left(W_1(Z,p) \; rel_i \; V_1(Z,p) \wedge \ldots \wedge W_n(Z,p) \; rel_i \; V_n(Z,p) \right) \tag{26}$$

where Z is a tuple of variables each taking values in the domain D, p is a list of parameters in D, and $\{W_i, V_i\}$ are polynomial terms in the semiring polynomial algebra:

$$\widetilde{D}\langle \Omega \rangle \tag{27}$$

18

with $\widetilde{D} = (D, \langle +, \cdot, 1, 0 \rangle)$ a semiring algebra with additive unit 0 and multiplicative unit 1. In (26), rel_i, $i = 1, \ldots, n$ are binary relations on the polynomial algebra. Each rel_i can be either an equality relation ($=$), inequality relation ($\neq$), or a partial order relation ($<$). In a given theorem, more than one partial order relation may appear. In each theorem, at least one of the terms is a partial order relation that defines a complete lattice on the algebra; that corresponds to the optimization problem. This lattice has a minimum element if the optimization problem has a minimum. Given a theorem statement of the form of (26) and a knowledge base of equational clauses, the inferencer determines whether the statement logically follows from the clauses in the Knowledge Base, and if so, as a side effect of the proof, generates a non-empty subset of tuples with entries in M giving values to Z. These entries determine the agent's actions. Thus, a side effect is instantiation of the agent's decision variables. In (27), Ω is a set of primitive unary operations, $\{v_i\}$, the infinitesimal primitive fields defined in section 2. Each v_i maps the semiring algebra, whose members are power series involving the composition of operators, on Z to itself:

$$v_i : \widetilde{D}\langle\langle Z \rangle\rangle \rightarrow \widetilde{D}\langle\langle Z \rangle\rangle \tag{28}$$

These operators are characterized by axioms in the Knowledge Base, and are process dependent. In formal logic, the implemented inference principle can be stated as follows: Let Σ be the set of clauses in the Knowledge Base. Let $\Rightarrow$ represent implication. Proving the theorem means to show that it logically follows from Σ, i.e.

$$\Sigma \Rightarrow theorem. \tag{29}$$

The proof is accomplished by sequences of applications of the following inference axioms:

(i) equality axioms

(ii) inequality axioms

(iii) partial order axioms

(iv) compatibility axioms

(v) convergence axioms

(vi) knowledge base axioms

(vii) limit axioms

The specifics of these inference axioms can be found in [12] where it is shown that each of the inference principles can be expressed as an operator on the Cartesian product:

$$\widetilde{D}\langle\langle W \rangle\rangle \times \widetilde{D}\langle\langle W \rangle\rangle \tag{30}$$

Each inference operator transforms a relational term into another relational term. The inferencer applies sequences of inference operators on the equational
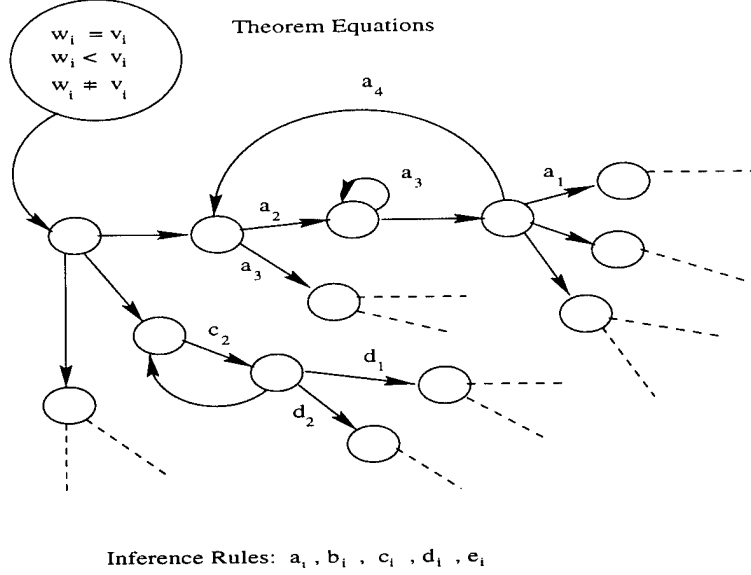
19

Figure 7: Conceptual Structure of the Proof Automaton

terms of the theorem until these terms are reduced to either a set of ground equations of the form of (31) or it determines that no such ground form exists.

$$Z_i = \alpha_i, \alpha_i \in D \tag{31}$$

The mechanism by which the inferencer carries out the procedure described above is by building a procedure for variable goal instantiation: a locally finite automaton. We refer to this automaton as the Proof Automaton. This important feature is unique to our approach. The proof procedure is customized to the particular theorem statement and Knowledge Base instance it is currently handling. The structure of the proof automaton generated by the inferencer is illustrated in Figure 7.

In Figure 7, the initial state represents the equations associated with the theorem. In general, each state corresponds to a derived equational form of the theorem through the application of a chain of inference operators to the initial state that is represented by the path,

Each edge in the automaton corresponds to one of the possible inferences. A state is terminal if its equational form is a tautology, or it corresponds to a canonical form whose solution form is stored in the Knowledge Base. In traversing the automaton state graph, values or expressions are assigned to the variables. In a terminal state, the equational terms are all ground states (see (31)). If the automaton contains at least one path starting in the initial state and ending in a terminal state, then the theorem is true with respect to the given Knowledge Base and the resulting variable instantiation is a valid one. If this is not the case, the theorem is false. The function of the complete
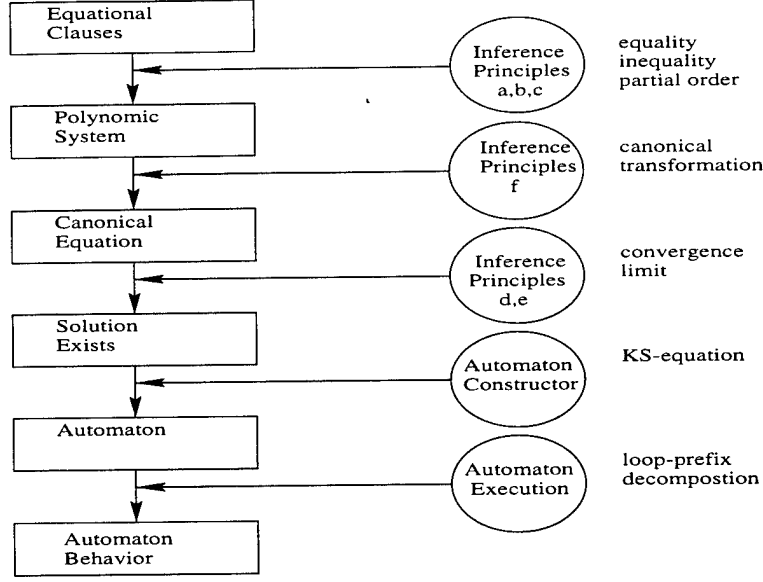
Figure 8: Summary of Inferencer Procedure

partial order term present in the conjunction of each theorem provable by the inferencer is to provide a guide for constructing the proof automaton. This is done by transforming the equational terms of the theorem into a canonical fixed point equation, called the Kleene-Schutzenberger Equation (KSE) [12], which constitutes a blueprint for the construction of the proof automaton. This fixed point coincides with the solution of the optimization problem formulated in (24) (25), when it has a solution. The general form of KSE is :

$$Z = E(p) \cdot Z + T(p) \tag{32}$$

In (32), E is a square matrix, with each entry a rational form constructed from the basis of inference operators described above, and T is a vector of equational forms from the Knowledge Base. Each non-empty entry, $E_{i,j}$, in E corresponds to the edge in the proof automaton connecting states i and j. The binary operator "·" between $E(p)$ and Z represents the "apply inference to" operator. Terminal states are determined by the non-empty terms of T. The p terms are custom parameter values in the inference operator terms in $E(\cdot)$.

A summary of the procedure executed by the inferencer is presented in Figure 8.

We note that the construction of the automaton is carried out from the canonical equation and not by a non-deterministic application of the inference rules. This approach reduces the computational complexity of the canonical equation (low polynomic) and is far better than applying the inference rules directly (exponential).

The automaton is simulated to generate instances of the state, action and

21

evaluation variables using an automaton decomposition procedure [37] which requires $nlog_2n$ time, where n # of states of the automaton. This "divide and conquer" procedure implements the recursive decomposition of the automaton into a cascade of parallel unitary (one initial and one terminal state) automata. Each of the resulting automata on this decomposition is executed independently of the others. The behavior of the resulting network of automata is identical with the behavior obtained from the original automaton, but with feasible time complexity.

The inferencer for each Estimation Agent fulfills two functions: to generate a proof for the system behavior theorem of each agent generated by the Planner (equations (24) and (25)) and to function as the central element in the Knowledge Decoder. We now describe its function for proving the behavior theorem. Later, we will overview its function as part of the Knowledge Decoder. To show how the inferencer is used to prove the Planner theorem, (24), (25), first, we show how this theorem is transformed into a pattern of the form of (26). Since (24), (25) formulates a convex optimization problem, a necessary and sufficient condition for optimality is provided by the following dynamic programming formulation:

$$V_i(Y,\tau) \quad = \quad inf_{\alpha_i} \int_\tau L_i(\Psi_i(\tau,Y), v_i|_p(G_i(\tau,p)))d\alpha_(p,d\tau) \qquad (33)$$

$$\frac{\partial V_i}{\partial \tau} \quad = \quad inf_{\alpha_i} H(Y, \frac{\partial V_i}{\partial \tau}, \alpha_i)$$
$$\text{where } Y(t) = p \text{ and } \tau \in [t, t+\Delta)$$

In (33), the function V_i, called the optimal cost-to-go function, characterizes minimality starting from any arbitrary point inside the current interval. The second equation is the corresponding Hamilton-Jacobi-Bellman equation for the problem stated in (24) and (25) where H is the Hamiltonian of the relaxed problem. This formulation provides the formal coupling between deductive theorem proving and optimal control theory. The inferencer allows the real-time optimal solution of the formal control problem resulting in intelligent distributed real-time control of the multiple-agent system. The central idea for inferring a solution to (33) is to expand the cost-to-go function $V(.,.)$ in a rational power series V in the algebra:

$$\widetilde{D}\langle\langle(Y,\tau)\rangle\rangle \qquad (34)$$

Replacing V for V_j in the second equation in (33), gives two items: a set of polynomic equations for the coefficients of V and a partial order expression for representing the optimality. Because of convexity and rationality of V, the number of equations to characterize the coefficients of V is finite. The resulting string of conjunctions of coefficient equations and the optimality partial order expression are in the form of (26).

In summary, for each agent, the inferencer operates according to the following procedure.

Step 1: Load current theorem (24), (25).

Step 2: Transform theorem to equational form (26) via (33).

Step 3: Execute proof according to figure 8.

If the theorem logically follows from the Knowledge Base (i.e., it is true), the inferencer procedure will terminate on step 3 with actions . If the theorem does not logically follow from the Knowledge Base, the Adapter is activated, and the theorem is modified by the theorem Planner according to the strategy outlined above. This mechanism is the essence of reactivity in the agent. Because of relaxation and convexity, this mechanism ensures that the estimatable set of the domain is strictly larger than the mechanism without this correction strategy.

3.1.5 Knowledge Decoder:

The function of the Knowledge Decoder is to translate knowledge data from the network into the agent's Knowledge Base by updating the inter- agent specification clauses. These clauses characterize the second constraint in (33). Specifically, they express the constraints imposed by the rest of the network on each agent. They also characterize the global-to-local transformations (see [22]). Finally, they provide the rules for building generalized multipliers for incorporating the inter-agent constraints into a complete unconstrained criterion, which is then used to build the cost-to-go function in the first expression in (33). A generalized multiplier is an operator that transforms a constrain into a potential term. This potential is then incorporated into the original Estimation Lagrangian of the agent which now accounts explicitly for the constraint.

The Knowledge Decoder has a built-in inferencer used to infer the structure of the multiplier and transformations by a procedure similar to the one described for (14). Specifically, the multiplier and transformations are expanded in a rational power series in the algebra defined in (34). Then the necessary conditions for duality are used to determine the conjunctions of equational forms and a partial order expression needed to construct a theorem of the form of (26) whose proof generates a multiplier for adjoining the constraint to the Estimation Lagrangian of the agent as another potential.

The conjunction of equational forms for each global-to-local transformation is constructed by applying the following invariant embedding principle:

> For each agent, the actions at given time t in the current interval, as computed according to (33), are the same actions computed at t when the formulation is expanded to include the previous, current, and next intervals.

By transitivity and convexity of the criterion, the principle can be analytically extended to the entire horizon. The invariant embedding equation has the same structure as the dynamic programming equation given in (33), but with the global criterion and global Hamiltonians instead of the corresponding local ones.
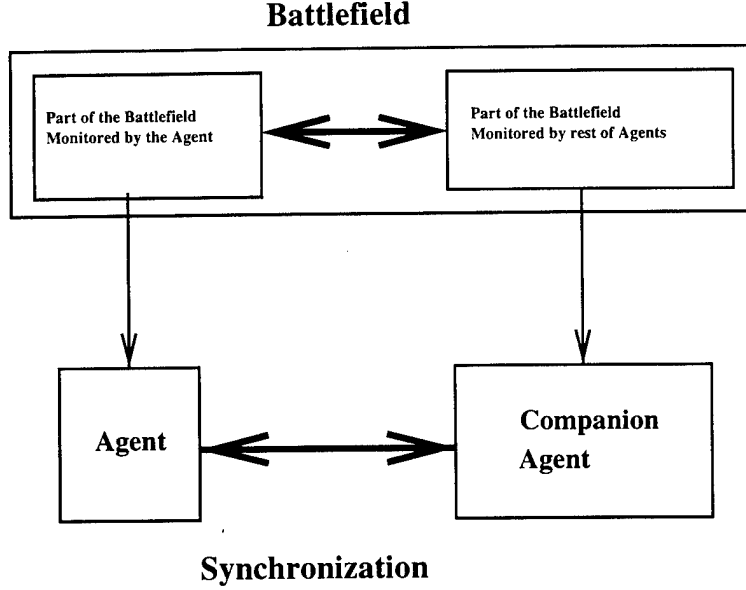
23

Battlefield

Synchronization

Figure 9: The Companion Agent

The local-to-global transformations are obtained by inverting the global-to-local transformations, obtained by expressing the invariant embedding equation as an equational theorem of the form of (26). These inverses exist because of convexity of the relaxed Lagrangian and the rationality of the power series.

It is important at this point to interpret the functionality of the Knowledge Decoder of each agent in terms of what it does. The multiplier described above has the effect of aggregating the rest of the system and the other agents into an equivalent companion system and companion agent, respectively, as viewed by the current agent. This is illustrated in Figure 9.

The aggregation model (Figure 9) describes how each agent perceives the rest of the network. This unique feature allows us to characterize the scalability of the architecture in a unique manner. Namely in order to determine computational complexity of an application, we have only to consider the agent with the highest complexity (i.e., the local agent with the most complex criterion) and its companion.

4 Constructing Estimation Lagrangians.

As described in the previous section, the Planner module of each agent A constructs an Estimation Lagrangian $L(t, x, \dot{x})$ such that the desired evolution of the system occurs when the system evolves along a curve $x = C(t)$ which mini-

mizes

$$\int_{t_0}^{t_0+\Delta} L(t, x, \dot{x})dt \tag{35}$$

where $x(t_0) = x_0$ and $x(t_0+\Delta) = x_1$. Here in local coordinates, $x = (x^1, \ldots, x^n)$ and $\dot{x} = (\dot{x}^1, \ldots, \dot{x}^n)$.

We would like to modify the agent's Estimation Lagrangian by adding a divergence free function to construct a Estimation Lagrangian with a special property that when we reach our desired estimation for the system that the resulting change in the estimation due to the flow is 0. That is, suppose that $S(t, x)$ is a smooth function (C^2 is enough). Then

$$\frac{dS}{dt} = \frac{\partial S}{\partial t} + \sum_{j=1}^{n} \frac{\partial S}{\partial x^j}\dot{x}^j. \tag{36}$$

Now suppose that we replace the agent's original Lagrangian by

$$\widetilde{L} = L - \frac{\partial S}{\partial t} - \sum_{j=1}^{n} \frac{\partial S}{\partial x^j}\dot{x}^j. \tag{37}$$

For any curve $C(t)$ with $C(t_0) = x_0$ and $C(t_0 + \Delta) = x_1$, let

$$J(C) = \int_{t_0}^{t_0+\Delta} L(t, C(t), \dot{C}(t))dt \tag{38}$$

$$\widetilde{J}(C) = \int_{t_0}^{t_0+\Delta} \widetilde{L}(t, C(t), \dot{C}(t))dt. \tag{39}$$

Then

$$J(C) - \widetilde{J}(C) = S(t + \Delta, x_1) - S(t, x_0). \tag{40}$$

Note that value of the left hand side of (40) does not depend on C so that that the curve C^* which minimizes $J(C)$ also minimizes $\widetilde{J}(C)$.

We want to construct a geodesic field for agent A

$$\dot{x}^j = \psi^j(t, x) \text{ for } j = 1, \ldots, n \tag{41}$$

on the manifold M such that

$$\widetilde{L}(t, x, \dot{x}) = 0 \text{ if } \dot{x}^j = \psi^j(t, x) \text{ for } j = 1, \ldots, n \tag{42}$$

$$\widetilde{L}(t, x, \dot{x}) > 0 \text{ if otherwise.} \tag{43}$$

Note that this a very strong condition which would ensure that evolving the system along a geodesic would result in no change to our estimate. See (18).

In this section we shall explore the conditions that such an S must satisfy and how we can construct the desired geodesic field.

To this end, we note that if there exists a curve $x = C(t)$ which minimizes

$$\int_{t_0}^{t_0+\Delta} L(t, x, \dot{x})dt \tag{44}$$

25

where $x(t_0) = x_0$ and $x(t_0 + \Delta) = x_1$, then L must satisfy the Euler-Lagrange equations. That is, let E_j be the operator

$$E_j = \frac{d}{dt}\Big(\frac{\partial}{\partial \dot{x}^j}\Big) - \frac{\partial}{\partial x^j}$$

for $j = 1, \ldots, n$. Then L must satisfy the equations

$$E_j(L) = 0 \text{ for } j = 1, \ldots, n. \tag{45}$$

This given, it will be useful for subsequent formulas to rewrite (45) in terms of the canonical momentum p^j and the Hamiltonian of the system H. That is, define

$$p^j = \frac{\partial L}{\partial \dot{x}_j}(t, x, \dot{x}) \text{ for } j = 1, \ldots, n. \tag{46}$$

We shall assume that

$$det\Big(\frac{\partial^2 L}{\partial \dot{x}^j \dot{x}^k}\Big) \neq 0 \tag{47}$$

for all $t, x, \dot{x}$. Given (47), we can use the Inverse Function theorem to solve for the $\dot{x}^j$'s in terms of x, t, and p:

$$\dot{x}^j = h^j(t, x, p) \ j = 1, \ldots, n. \tag{48}$$

We then define the Hamiltonian H of the system by

$$H(t, x, p) = -L(t, x, \dot{x}) + \sum_{j=1}^{n} p^j h^j(t, x, p). \tag{49}$$

It then is easily verified that

$$\frac{\partial H}{\partial p^j} \quad = \quad h^j = \dot{x}^j, \tag{50}$$

$$\frac{\partial H}{\partial x^j} \quad = \quad -\frac{\partial L}{\partial x^j}, \text{ and} \tag{51}$$

$$\frac{\partial H}{\partial p^j} \quad = \quad -\frac{\partial L}{\partial t}. \tag{52}$$

It follows that the Euler-Lagrange equations are equivalent to the following.

$$\frac{d}{dt}p^j + \frac{\partial H}{\partial x^j} = 0 \text{ for } j = 1, \ldots, n. \tag{53}$$

It follows from (42) and (43) that

$$\widetilde{L}(t, x^1, \ldots, x^n, \psi^1, \ldots, \psi^n) = min_{\dot{x}}\widetilde{L}(t, x, \dot{x}). \tag{54}$$

Thus the solution to (41) must satisfy

$$\begin{aligned} 0 \quad &= \quad \frac{\partial \widetilde{L}}{\partial \dot{x}^j} \\ &= \quad \frac{\partial L}{\partial \dot{x}^j} - \frac{\partial S}{\partial x^j} \\ &= \quad p^j - \frac{\partial S}{\partial x^j}. \end{aligned}$$

26

Thus when $\dot{x}^j = h^j(t, x, p) = \psi^j(t, x)$ for $j = 1, \ldots, n$, then

$$p_j(t, x, \dot{x}) = \frac{\partial S}{\partial x^j}(t, x)$$

so that p_j is a function of just t and x. Moreover it must be the case that

$$
\begin{aligned}
0 &= \widetilde{L}(t, x, \dot{x}) \\
&= L(t, x, \dot{x}) - \frac{\partial S}{\partial t} - \sum_{j=1}^{n} \frac{\partial S}{\partial x^j} \dot{x}^j \\
&= L(t, x, \dot{x}) - \frac{\partial S}{\partial t} - \sum_{j=1}^{n} p^j h^j
\end{aligned}
$$

and hence S satisfies

$$\frac{\partial S}{\partial t} + H(t, x, \frac{\partial S}{\partial x}) = 0. \tag{55}$$

We claim that if the (41), (42), and (43) hold and $S(t, x)$ is a function such that (55) holds and

$$p^j = \frac{\partial L}{\partial \dot{x}_j}(t, x \, \dot{x}) - \frac{\partial S}{\partial x^j}(t, x), \tag{56}$$

then the Wierstrass condition for the existence of a curve $C(t)$ which is a strong minimum for (44). It then follows that we can construct an ϵ-approximation to the geodesic fields via the techniques of [26].

To make our statement precise, we make the following definition, see [7].

Definition 4.1 *The Weierstrass E-function of the functional*

$$J[y] = \int_a^b F(x, y, \dot{y}) dx, \ y(a) = A, \ y(b) = B \tag{57}$$

the function of $3n + 1$ variables:

$$E(x, y, x, w) = F(x, y, w) - F(x, y, z) - \sum_{i=1}^{n}(w^i - z^i)\frac{\partial f}{\partial \dot{y}^i}. \tag{58}$$

Then the following result is proved in [7].

Theorem 4.2 *Let γ be a curve which is extremal for the functional $J[y]$ of (57) and let*

$$\dot{y}^i = \psi^i(x, y) \ for \ i = 1, \ldots, n \tag{59}$$

be a field for the functional $J[y]$ of (57). Suppose that at every point (x, y) of some (open) region containing γ and covered by the field (59), the condition

$$E(x, y, \psi, w) \geq 0 \tag{60}$$

is satisfied for every finite vector $w = (w^1, \ldots, w^n)$. Then $J[y]$ has a strong minimum for the extremal γ.

27

Now under our assumptions,

$$
\begin{aligned}
\dot{x}^j &= h^j(t,x,p) \\
&= h^j\!\left(t,x,\frac{\partial S}{\partial x^j}\right) \\
&= \psi^j(t,x) \text{ for } j = 1,\dots,n.
\end{aligned}
\tag{61}
$$

Now take $\frac{d}{dt}$ of both sides of (54), we get

$$
\frac{dp^j}{dt} = \frac{\partial^2 S}{\partial t \partial x^j} + \sum_{k=1}^{n} \frac{\partial^2 S}{\partial x^k \partial x^j} \dot{x}^k.
\tag{62}
$$

Then taking the partial derivative with respect to x^j of both sides of (55), we get

$$
\frac{\partial^2 S}{\partial t \partial x^j} + \frac{\partial H}{\partial x^j} + \sum_{k=1}^{n} \frac{\partial H}{\partial p^k} \frac{\partial^2 S}{\partial x^j \partial x^k} = 0.
\tag{63}
$$

However since $\dot{x}^j = \frac{\partial H}{\partial p^j}$ for $j = 1,\dots n$, we can rewrite (63) as

$$
\frac{\partial^2 S}{\partial t \partial x^j} + \frac{\partial H}{\partial x^j} + \sum_{k=1}^{n} \frac{\partial^2 S}{\partial x^j \partial x^k} \dot{x}^k = 0.
\tag{64}
$$

Combining (62) and (64), we get that

$$
\frac{dp^j}{dt} = -\frac{\partial H}{\partial x^j}
\tag{65}
$$

for $j = 1,\dots n$. Note that this implies the system can be characterized by the equations

$$
\dot{x}^j = \frac{\partial H}{\partial p^j} \quad j = 1,\dots,n
\tag{66}
$$

and

$$
p^j = \frac{\partial H}{\partial x^j} \quad j = 1,\dots,n.
\tag{67}
$$

Next observe that conditions (42) and (43) imply that

$$
\frac{\partial S}{\partial t} - \frac{\partial S}{\partial \dot{x}^j} - \sum_{j=1}^{n} \frac{\partial S}{\partial x^j} \dot{x}^j \geq 0
\tag{68}
$$

with equality holding only if

$$
\dot{x}^j = \psi^j(t,x).
$$

Then (55) can be rewritten as

$$
\frac{\partial S}{\partial t} - L(t,x,\psi) + \sum_{j=1}^{n} \frac{\partial S}{\partial x^j} \psi^j = 0.
\tag{69}
$$

28

Solving for $\frac{\partial S}{\partial t}$ in (69) and substituting into (68), we set that

$$L(t,x,\dot{x}) - L(t,x,\psi) + \sum_{j=1}^{n} \frac{\partial S}{\partial x^j}(\dot{x}^j - \psi^j) \geq 0 \tag{70}$$

Thus since $\frac{\partial S}{\partial x^j} = p^j = \frac{\partial L}{\partial x^j}$, we can derive that

$$L(t,x,\dot{x}) - L(t,x,\psi) + \sum_{j=1}^{n} \frac{\partial L}{\partial x^j}(\dot{x}^j - \psi^j) \geq 0 \tag{71}$$

Next define the Wierstrass function E by

$$E(t,x,\psi,\dot{x}) = L(t,x,\dot{x}) - L(t,x,\psi) + \sum_{j=1}^{n} \frac{\partial S}{\partial x^j}(\dot{x}^j - \psi^j). \tag{72}$$

Then (71) says that

$$E(t,x,\psi,\dot{x}) \geq 0. \tag{73}$$

Moreover by applying Mean Value Theorem twice, we get that

$$E(t,x,\psi,\dot{x}) = \frac{1}{2} \sum_{j,k} \frac{\partial^2 L}{\partial \dot{x}^j \dot{x}^k}(t,x,z)(\dot{x}^j - \psi^j)(\dot{x}^k - \psi^k) \tag{74}$$

for some $z = \theta x + (1-\theta)\psi$ with $0 < \theta < 1$. Thus if $E(t,x,\psi,\dot{x}) > 0$, it must be that

$$det\left(\frac{\partial^2 L}{\partial \dot{x}^j \dot{x}^k}\right) > 0.$$

5 Agent Synchronization

Agent synchronization is based on the following result.

Theorem 5.1 *For any agents A_0 and A_1, let L_0 and L_1 be the corresponding agent Estimation Lagrangian where for each i, L_i is a function of the state x_i, $\dot{x}_i$, and the agent clock time t. We can explicitly construct state and clock transformation functions Ψ_x^{01} and Ψ_t^{01} from the Noether invariance relations such that for any given time interval I and $\epsilon > 0$,*

$$\int_I [L_0(x_0,\dot{x}_0,t_0) - L_1(\Psi_x^{01}(x_0,t_0), \dot{\Psi}_x^{01}(x_0,t_0), \Psi_t^{01}(x_0,t_0))]dt_0 \leq \epsilon. \tag{75}$$

The significance of this result is that one can explicitly construct transformation function to coherently fuse sensor data. That is, agents A_0 and A_1 may have different models of the battlefield. Thus for agents A_0 and A_1 to be able to communicate with each other, we need to construct transformations functions which allow agent A_0 to interpret agent A_1's state estimation. The theorem says that such transformation functions not only exist but can be computed for

the Noether invariance relations. Moreover if agent A_0 realizes that his state is incompatible with the state information given by agent A_1, agent A_0 goes to the adaptation loop to correct the incompatibility.

To illustrate Theorem 5.1, suppose that there is a global state estimation function for the battlefield $L(t, x, \dot{x})$ and we have two agents, Agent 1 with its Estimation Lagrangian $L_1(t_1, x_1, \dot{x}_1)$ and Agent 2 with its state estimation function $L_2(t_2, x_2, \dot{x}_2)$. Moreover assume that we have state and clock transition functions for Agent i, $i = 1, 2$, given by

$$x_i^j = x_i^j(t, x, w) \text{ for } j = 1, \ldots, n \qquad (76)$$

$$t_i = t_i(t, x, w) \qquad (77)$$

where $w = (w^1, \ldots, w^n)$ is a set of parameters for the transformation. That is, we assume

$$L_i(x_i(t, x, w), x_i(t, x, w), \dot{x}_i(t, x, w)) = L(t, x, \dot{x}) \qquad (78)$$

for $i = 1, 2$.

Next we shall state the Noether Invariance relations referred to in Theorem 5.1. We first need some notation. First we define two classes of infinitesimal transformations to each Agent i.

$$\frac{\partial x_i^j}{\partial w^k}|_{w=0} = \theta_{i,k}^j(t, x) \qquad (79)$$

and

$$\frac{\partial t_i}{\partial w^k}|_{w=0} = \tau_{i,k} \qquad (80)$$

Next let $E_{i,j}, = 1, \ldots, n$, be the Euler-Lagrange operators for L_i. That is, let

$$E_{i,j}(L_i) = \frac{d}{dt_i}\left(\frac{\partial L_i}{\partial \dot{x}_i^j}\right) - \frac{\partial L_i}{\partial x_i^j} \qquad (81)$$

This given, the Noether Invariance Relations are given by

$$\sum_{j=1}^{n} E_{i,j}(L_i)(\theta_{i,k}^j - \dot{x}_i^j \tau_{i,k}) = \frac{d}{dt}\left(L_i \tau_{i,k} + \sum_{j=1}^{n_i} \frac{\partial L_i^j}{\partial \dot{x}_i^j}(\theta_{i,k}^j - \dot{x}_i^j \tau_{i,k})\right) \qquad (82)$$

for $k = 1, \ldots, n$

The idea is that if the state estimation Lagrangian L_i has an extremal, i.e. if there exists a curve $x_i(t) = C_i(t)$ which minimizes

$$\int_{t_0}^{t_0+\Delta} L(t_i, x_i, \dot{x}_i) dt_i \qquad (83)$$

where $x(t_0) = x_0$ and $x(t_0 + \Delta) = x_1$, then L must satisfy the Euler-Lagrange equations $E_{i,j}(L_i) = 0$ for $j = 1, \ldots, n_i$. In that case the left hand side of the equations in (82) are 0 and hence there are constants $c_{i,k}$ such that

$$L_i \tau_{i,k} + \sum_{j=1}^{n_i} \frac{\partial L_i^j}{\partial \dot{x}_i^j}(\theta_{i,k}^j - \dot{x}_i^j \tau_{i,k}) = c_{i,k} \qquad (84)$$

30

for $k = 1, \ldots, n$ There are two important aspects about (84). First note that given the constants $c_{i,k}$, we can solve for the infinitesimals $\tau_{i,k}$ and $\theta^j_{i,k}$ and integrate to recover the desired transformation x^j_i and t_i. We will illustrate this type of calculation with a simple example below. Second, note that we can monitor the failure of synchronization of the state estimation Lagrangians by simply observing that the left hand side of (84) is not a constant. If the left hand side of (84) is not a constant, then we know that the current agent Lagrangian is not compatible with the system Lagrangian so that Agent i would use the Adapter to reconstruct his Estimation Lagrangian.

Example 5.1 *A simplified range model of radar returns has the following system Lagrangian.*

$$L = t^2 \left(\frac{\dot{x}^2}{2} - \frac{x^6}{6} \right). \tag{85}$$

Now suppose that Agent 1 has infinitesimal transformations

$$t_1 = t + \tau(x,t)w \tag{86}$$

$$x_1 = x + \theta(x,t)w \tag{87}$$

Then from (82), we have

$$(\dot{x}^2 - \frac{x^6}{3})\tau_1 - tx^5 + t\dot{x}\left(\frac{\partial \theta}{\partial t} + \frac{\partial \theta}{\partial x}\dot{x}\right) \tag{88}$$

$$-\dot{x}\frac{\partial \tau}{\partial t} - \dot{x}^2\frac{\partial \tau}{\partial x} + t(\dot{x}^2 - \frac{x^6}{6})\left(\frac{\partial \tau}{\partial t} + \frac{\partial \tau}{\partial t}\dot{x}\right) = 0.$$

Collecting terms, we obtain the following relations.

$$\dot{x}^0 \quad : \quad \frac{x}{3}\tau + t\theta + t\frac{x}{6}\frac{\partial \tau}{\partial t} = 0 \tag{89}$$

$$\dot{x}^1 \quad : \quad \frac{\partial \theta}{\partial t} - \frac{x^6}{6}\frac{\partial \tau}{\partial x} = 0 \tag{90}$$

$$\dot{x}^2 \quad : \quad t\tau + t\frac{\partial \tau}{\partial x} - \frac{t^2}{2}\frac{\partial \tau}{\partial t} = 0 \tag{91}$$

$$\dot{x}^3 \quad : \quad \frac{\partial \tau}{\partial x} = 0 \tag{92}$$

Note that by (92), $\tau = \tau(t)$ is just a function of t. Also (92) and (90) imply that $\frac{\partial \theta}{\partial t} = 0$ so that $\theta = \theta(x)$ is just a function of x. This means that (89) and (91) are just ordinary differential equations which one can easily solve via standard power series methods to show that

$$\tau \quad = \quad t \tag{93}$$

$$\theta \quad = \quad -\frac{1}{2}x. \tag{94}$$

Hence

$$t_1 \quad = \quad t + tw \tag{95}$$

$$x_1 \quad = \quad x - \frac{1}{2}xw. \tag{96}$$

31

6 Conclusions

We have shown that our MAHEA architecture is an effective mechanism to solve various problems of estimating a process in which the data available is dynamic, noisy, and given in a multiplicity of representations. A MAHEA agent network for estimating battlefield state is an efficient mechanism for state estimation which is extensible, robust, scalable, allows cross-checking, and supports heterogeneous information.

Deployment of an agent based system is very simple. As soon as a new source of information is available, a new estimation agent is spawned whose Knowledge Base is a model of the battlefield covered by that source (extensibility). Moreover no common representation of the data is required so that the system supports heterogeneous information sources. Thus our architecture allows us to incorporate existing models and estimation techniques. The built-in invariance condition tests for the validity of the data (cross-checking).

We have shown that the existing theory of hybrid systems and relaxed variation optimization can be adapted to the state estimation problem for the battlefield by constructing a Lagrangian which becomes 0 at points which correspond to consistent estimates of the battlefield and is positive at points which are not consistent. This property ensures that when an agent reaches consistent estimates of the battlefield, the evolution of the system produced by the flow of the corresponding geodesic field is adapted to the current information of the battlefield as viewed by each agent.

Another key result of our investigation is a theorem which we call the Theveninn Theorem which states in a network with many agents, an individual agent A can view the rest of the sensor agents as a single aggregated agent $C(A)$ called A's companion agent. Since an agent sees the rest of the estimate agent network as a single equivalent estimation agent, the architecture maintains linear complexity even as more agents are added. Details of these results will appear in a future report

Finally the robustness of the sensor agent's estimates follows from the fact that each MAHEA agent is computed by a finite state machine which is Lyapunov stable because the mapping induced by an agent is a contraction mapping.

We note that the techniques introduced in this paper for synchronization and consistency of state estimates can be adapted to solve synchronization problems for our Multiple Agent Hybrid Control Architecture. Again details will be given a future report

References

[1] Antsaklis, P., Kohn, W., Nerode, A, and Sastry, S. eds., *Hybrid Systems II*, Lecture Notes in Computer Science vol. 999, Springer-Verlag, (1995).

[2] Crossley, J.N., Remmel, J.B., Shore, R.A. and Sweedler, M.E., *Logical Methods* Birkhauser, (1993).

[3] Dodhiawala, R.T., V. Jagoenathan and L.S. Baum, "Erasmus System Design: Performance Issues" Proceedings of Workshop on Blackboard Systems Implementation Issues, AAAI, Seattle, WA., July 1987.

[4] Garcia, H.E. and A. Ray "Nonlinear Reinforcement Schemes for Learning Automata" Proceedings of the 29th IEEE CDC Conference, Vol. 4, pp 2204- 2207, Honolulu, HA, Dec. 5-7, 1990.

[5] Ge, X., Kohn, W., Nerode, A. and Remmel, J.B., "Algorithms for Chattering Approximations to Relaxed Optimal Control. MSI Tech. Report 95-1, Cornell University. (1995)

[6] Ge, X., Kohn, W., Nerode, A. and Remmel, J.B., "Feedback Derivations: Near Optimal Controls for Hybrid Systems", to appear in Hybrid Systems III, Springer Lecture Notes in Computer Science.

[7] Gelfand, I.M. and Fomin, S.V., *Calculus of Variations*, Prentice Hall, 1963.

[8] Grossman, R.L., Nerode, A., Ravn, A. and Rischel, H. eds., *Hybrid Systems*, Lecture Notes in Computer Science 736, Springer-Verlag, (1993).

[9] Kohn, W., "A Declarative Theory for Rational Controllers" Proceedings of the 27th IEEE CDC, Vol. 1, pp 131-136, Dec. 7-9, 1988, Austin, TX.

[10] Kohn, W., "Application of Declarative Hierarchical Methodology for the Flight Telerobotic Servicer" Boeing Document G-6630-061, Final Report of NASA- Ames research service request 2072, Job Order T1988, Jan. 15, 1988.

[11] Kohn, W., "Rational Algebras; a Constructive Approach" IR&D BE-499, Technical Document D-905-10107-2, July 7, 1989.

[12] Kohn, W., "The Rational Tree Machine: Technical Description & Mathematical Foundations" IR&D BE-499, Technical Document D-905-10107-1, July 7, 1989.

[13] Kohn, W., "Declarative Hierarchical Controllers" Proceedings of the Workshop on Software Tools for Distributed Intelligent Control Systems, pp 141-163, Pacifica, CA, July 17-19, 1990.

[14] Kohn, W., "Declarative Multiplexed Rational Controllers" Proceedings of the 5th IEEE International Symposium on Intelligent Control, pp 794-803, Philadelphia, PA, Sept. 5, 1990.

[15] Kohn, W., "Declarative Control Architecture" CACM Aug 1991,Vol34, No8.

[16] Kohn, W., "Advanced Architectures and Methods for Knowledge-Based Planning and Declarative Control" IR&D BCS-021, ISMIS'91, Oct. 1991.

33

[17] Kohn, W. and Murphy, A., "Multiple Agent Reactive Shop Floor Control" ISMIS'91, Oct. 1991.

[18] Kohn W., "Multiple Agent Inference in Equational Domains Via Infinitesimal Operators" Proc. Application Specific Symbolic Techniques in High Performance Computin g Environment". The Fields Institute, Oct 17-20 1993.

[19] Kohn W., "Multiple Agent Hybrid Control" Proc of thhe NASA-ARO Workshop on formal Models for Intelligent Control, MIT,sept 30- Oct2, 1193. Will appear as a paer in IEEE Ac.

[20] Kohn, W. and Nerode, A., "Multiple Agent Declarative Control Architecture" Proc. of the workshop on Hybrid Systems, Lygby, Denmark, Oct 19-21, 1992.

[21] Kohn, W. and Nerode, A., "Foundations of Hybrid Systems" In Hybrid Systems, Nerode, A, R, Grossman Eds. Springer Verlag series In Computer Science #726,

New York, 1993.

[22] Kohn W., and Nerode, A., "Multiple-Agent Hybrid Systems" Proc. IEEE CDC 1992, vol 4, pp 2956, 2972.

[23] Kohn, W. and Nerode, A., "An Autonomous Systems Control Theory: An Overview" Proc. IEEE CACSD'92, March 17-19, Napa, Ca.,pp 200- 220.

[24] Kohn W,, and Nerode A. "Models For Hybrid Systems: Automata, Topologies, Controllability, Observability" In [8], (1993) 317-356.

[25] Kohn W. and Nerode, A., "Multiple Agent Hybrid Control Architecture" In *Logical Methods* (J. Crossley, J. B. Remmel, R. Shore, M. Sweedler, eds.), Birkhauser, (1993) 593-623.

[26] Kohn, W., Nerode, A. and Remmel, J.B., " Hybrid Systems as Finsler Manifolds: Finite State Control as Approximation to Connections", In [1], (1995).

[27] Kohn, W., Nerode, A. and Remmel, J.B., "Continualization: A Hybrid Systems Control Technique for Computing", to appear in the proceedings of CESA'96.

[28] Kohn, W., Nerode, A. and Remmel, J.B., "Feedback Derivations: Near Optimal Controls for Hybrid Systems", to appear in CESA'96.

[29] Kohn, W. and T. Skillman, "Hierarchical Control Systems for Autonomous Space Robots" Proceedings of AIAA Conference in Guidance, Navigation and Control, Vol. 1, pp 382-390, Minneapolis, MN, Aug. 15-18, 1988.

[30] Kowalski, R., "Logic for Problem Solving" North Holland, NY, 1979.

34

[31] Kuich, W. and Salomaa, A., "Semirings, Automata, Languages" Springer Verlag, NY., 1985.

[32] Lloyd, J.W. "Foundations of Logic Programming" second extended edition, Springer Verlag, NY, 1987.

[33] Liu, J.W.S., "Real-Time Responsiveness in Distributed Operating Systems and Databases" proceedings of the Workshop on Software Tools for Distributed Intelligent Control Systems, Pacifica, CA., July 17-19, 1990, pp 185-192.

[34] Nii, P.H., "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures" the AI Magazine, Vol. 7, No. 2, Summer 1986, pp 38-53.

[35] Padawitz, P., "Computing in Horn Clause Theories" Springer Verlag, NY, 1988.

[36] Robinson, J.A., "Logic: Form and Function" North Holland, NY, 1979.

[37] Skillman, T. and Kohn, W., et.al., "Class of Hierarchical Controllers and their Blackboard Implementations" Journal of Guidance Control & Dynamics, Vol. 13, N1, pp 176-182, Jan.-Feb., 1990.

[38] Warner, F.W., *Foundations of Differential Manifolds and Lie Groups*, Scott-Foresman, Glenview, Ill.

[39] Warga, K., "Optimal Control of Differential and Functional Equations" Academic Press, NY., 1977.

[40] Young, L.C., "Optimal Control Theory" Chelsea Publishing Co., NY, 1980.