1

Abstraction-Based Command and Control with Patch Models

V. G. Rao, S. Goldfarb and R. D'Andrea

Cornell University

Abstract: In this chapter, we provide an overview of an abstraction-based command and control framework for complexity management called Patch Models . We review the mathematical structure of these models and the processes required to realize them through controlled motion of multiple vehicles, and verify them with respect to observations. We also discuss decision-making procedures and the use of the models in hierarchical command and control schemes. Open problems and directions for future work are highlighted throughout.

Keywords: Patch models, abstraction, command and control, hierarchical structure, verification, representation.

1.1 Introduction

In this chapter, we consider the problem of complexity management in modern warfare, particularly as it concerns cooperative control (with humans in the loop) of assets such as autonomous vehicles, employed against an adversary. Our approach to this broad problem is based on systematic abstraction of battlespace information into models that can be tailored to be relevant to the local situation. Decision-makers using such models, through appropriate software and connected through appropriate communication mechanisms, can provide, we believe, an information infrastructure that can manage large-scale complexity.

In military terms, the objective is to achieve a system-wide condition of "shared situation awareness" (SSA) (Endsley 1995; Nofi 2000) that allows greater responsiveness to events on the battlefield. The specific instantiation of this approach described here – the method of patch models – is adapted to represent fast-changing spatio-temporal information, concerning mobile friendly and enemy assets. The approach, however, can be generally applied in broader contexts, and we indicate elsewhere in this chapter the guiding principles that should be employed in the development of such abstraction-based systems.

Our objective is to provide an expository overview of work that has previously been described in technical detail elsewhere (Rao and D'Andrea 2006; Rao et al. 2006), and to discuss several important emerging problems for which we offer promising attacks.

For motivation, consider the following prototypical modern battlefield narrative. An enemy threat is detected by a large-area surveillance aircraft, and tracked coarsely. Shortly, a lower-flying autonomous surveillance aircraft, with greater resolution, arrives to loiter on the scene and track the target more closely. Meanwhile an autonomous attack aircraft is diverted from a different mission. Once the attack aircraft arrives on the scene, there is a short period of joint surveillance, followed by an attack. The mission is completed when the surveillance aircraft takes pictures of the effect of the strike, an operation usually referred to as "battle damage assessment," or BDA. The entire sequence of events takes twenty minutes.

Consider the problem of realizing the capabilities required to make the above scenario a routine one. Viewed in isolation, it may appear that the capabilities already exist. Certainly, situations very similar to this have already occurred. Imagine, however, this level of responsiveness and success for almost all threats detected anywhere in a large battlespace, and imagine further, that the detection rate is high, due to highly capable surveillance systems. The nature of the problem then, becomes clear, since we must now consider the infrastructure of capabilities that would cause such a scenario to play out in nearly *every* instance where it was necessary, against the backdrop of a large number of similar scenarios already in progress. It is this infrastructure that concerns us in this chapter.

Such a hypothetical future command and control system would be a highly complex entity. It has been noted (Simon 1996) that such complex systems typically comprise nearly-decomposable components. Our hypothetical system must be factored, or partitioned, into component capabilities that, one hopes, constitute a nearly exhaustive and almost mutually exclusive set. The work in this book was driven by one relatively reasonable partitioning of the overall problem into four such capabilities: distributed control and computation, management of adversarial interactions, management of uncertain evolution, and complexity management. This chapter concerns the fourth aspect, "complexity management."

Complexity management, in our view, is a problem dimension that is orthogonal to all other dimensions. In every dimension, complexity arises from at least two sources. The first is the effect of scaling up functions and subsystems. What, for instance, happens to a resource allocation system when it deals with hundreds or thousands of resources instead of dozens? This effect is largely a computational-complexity effect, and one seeks efficient, preferably distributed methods for solving problems known to be intrinsically hard (in the sense of NP completeness). The work of Carla Gomes[1] and her colleagues on recognizing and exploiting the phenomena of phase transitions and backdoors in NP-complete problems is an appropriate approach for this aspect of complexity.

[1] http://www.cs.cornell.edu/gomes/

The second source is the increased heterogeneity, richer ontology and increased communication complexity (in a semantic sense) that accompanies scaling to broader contexts of activity. Can a scheduler developed for aircraft in the scenario above be adapted to the presence of other battlespace elements such as spacecraft and missiles? Can a new kind of video camera used on a new variant of a surveillance system seamlessly feed its output into the battlespace management software used on a JSTARS aircraft? Such issues are the focus of "complexity management" as understood in this chapter.

Our approach to such problems is based on four guiding principles. The first is that the *representation* of information necessary to make decisions is more fundamental than the decision-making processes themselves. Second, the representations must be amenable to sharing across a variety of decision-processes (such as resource allocation and mission profile selection). Third, the representations must lend themselves to consistent, shared use by both human and artificial decision-makers. Fourth, the representations must be applicable to a variety of decision-making contexts, ranging from ground-level high-speed tactical contexts to slower-changing strategic contexts much further upstream in the decision-making hierarchy.

These four principles have been applied in the development of the methodology of patch models described here. In Section 1.2, we provide an overview of the modeling framework. In Section 1.3 we consider procedures for realization and verification of the models, with particular attention to the latter. In Section 1.4, we consider human and artificial decision-making with patch models. In Section 1.5, we consider the issues involved in the collaborative use of such models by a hierarchy of decision-makers. In Section 1.6, we present our conclusions. Throughout, we indicate open problems where they appear in the course of the development.

1.2 Overview of Patch Models

We begin by considering how we might represent a battlefield situation involving several friendly, neutral and enemy moving entities from the point of view of an individual decision-maker. This decision maker might be either a human performing a well-defined function (such as allocation of aircraft to targets or air-traffic control), or an ambiguous function such as reviewing status information and evaluating various what-if scenarios to develop or select a course of action. In the battlefield of the future, the decision-maker might well be a software program (with appropriate override mechanisms for supervising humans). Both human and artificial decision-makers may either be onboard the mobile resources they control, or at remote locations. All these decision-makers, performing different functions in different contexts require representations of the battlespace suited to their needs, and patch models serve this purpose. We provide here a simplified description of these models, which accommodate information that can be linked to primitive *spatio-temporal* information. A more detailed technical exposition with a detailed simulation case study can be found in Rao and D'Andrea (2006).

The information content of a battlespace is derived from a (possibly very large) set of independent noisy variables that must be measured, such as vehicle positions. Call this information *primitive* information. We restrict ourselves in this work to spatio-temporal primitive information. We suppose that there is a set $\mathcal{A}$ of primitive agents (such as vehicles) $\{a_1, \ldots, a_{\mathbf{q}}\}$, each of which is described by a state vector. Each state vector must contain

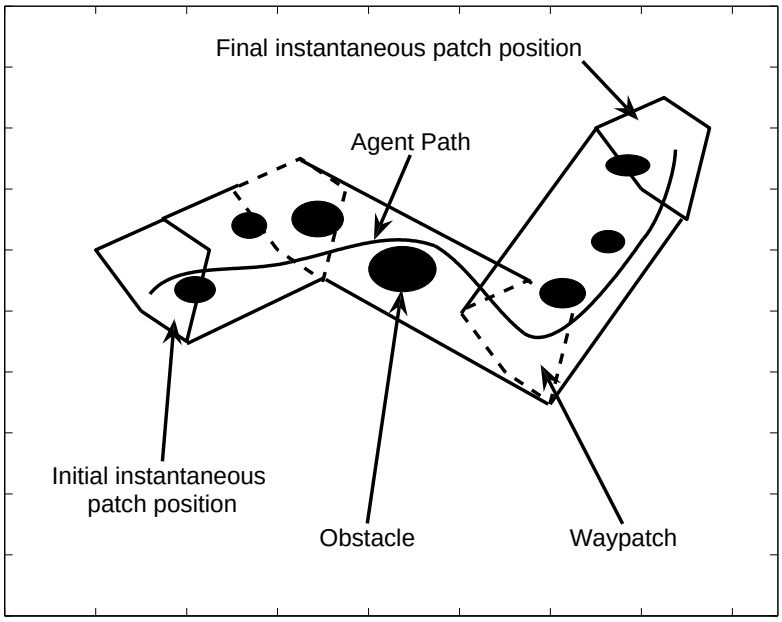

Figure 1.1 Primitive spatio-temporal information (an agent path) and its representation with a patch

spatio-temporal elements such as vehicle positions and velocities that allows the information to be organized spatio-temporally. Each state vector may also include different, agent-specific non-spatio-temporal elements such as camera-state or weapon-state. We may also include design-dependent abstract variables such as "damage state." We stack these individual agent state vectors and write the detailed domain dynamics as follows:

$$\begin{aligned} \dot{\bar{x}} &= f(\bar{x}, \bar{u}, \bar{w}), \\ \bar{y} &= h(\bar{x}, \bar{v}), \\ \bar{u} &= g(\bar{y}), \\ p_i(\bar{x}, \bar{u}) &\geq 0, i = 1, 2, \ldots, \end{aligned} \tag{1.1}$$

where $\bar{x}$, $\bar{u}$ and $\bar{y}$ are the global state, control and observation vectors respectively, and the p_i represent constraints in a standard form. The noise processes $\bar{w}$ and $\bar{v}$ are noise processes capturing the effects of imperfect surveillance capabilities and vehicle control capabilities. Clearly, individual decision-makers cannot observe $\bar{y}$ completely or influence all elements of $\bar{u}$ equally. These decentralization conditions must be modeled through constraints on the allowable forms of the control and observation functions, f and g which we will elaborate upon in later sections.

The primitive information is clearly vast in scope as well as quantity, and its evolution is described by complex dynamics. Much of this information is irrelevant to most decision-making roles. We cannot, however, merely omit states, as is done in model-order reduction

in classical control, since in some cases high *accuracy* and *precision* may be irrelevant for a decision, but coarse information may still be required. The first priority in modeling is therefore to select relevant information and to aggregate it into more abstract building blocks with relevance-dependent precision. We achieve this by introducing a construct called a patch. A patch is a moving convex figure, (here we restrict ourselves to polygons), as illustrated in Fig. 1.1, that moves along a piece-wise linear path in two-dimensional space, defined as follows:

$$P = \{R, T_1, \ldots T_{\mathbf{j}}\}, \tag{1.2}$$

$$R = \begin{pmatrix} x_1 & y_1 \\ \ldots & \ldots \\ x_{\mathbf{n}} & y_{\mathbf{n}} \end{pmatrix}, \tag{1.3}$$

$$T_j = \begin{bmatrix} x_j & y_j & \lambda_j & t_j \end{bmatrix}. \tag{1.4}$$

In the above, R is a root polygon and the T_i, $i \leq \mathbf{i}$, (called *waypatches*) represent affine transformations, each comprising a new translated position, a scaling factor and a time. This information structure is a patch. As shown in Fig. 1.1, visually a patch is a series of piecewise linear bands in space (or a tube in space-time). The spatial "band" is referred to as the *trace of the patch*, $P([t_0, t_f])$, while the instantaneous position of the transformed root polygon is denoted $P(t)$. Note that the term "patch" refers to the entire trajectory of the root polygon, and is thus an integral rather than a differential representation primitive.

A patch is meant to represent one or more agents over an interval of time of interest in a manner that aggregates individual agent information, and simplifies their dynamic behavior. For example, a sufficiently large triangle that is assumed to move at constant velocity in a straight line between start and end points requires just 14 independent pieces of primitive information (a root polygon and two waypatches, velocity being implicit), but can subsume an arbitrary number of agents with complex dynamics at an appropriate scale. The cost of using a patch as a representational primitive in place of primitive data is that there is loss of precision in both observation and control. For a given decision-maker, some primitive information will require representation with very precise, tight patches that move in ways that very closely conform to the actual motion of a small number of agents, while other parts of the primitive information set may be lumped together to form a very coarse and slowly varying representation that is sufficient for the objective at hand. We therefore construct *patch models* out of individual patches as follows. We define a set of patches, $M = \{P_1, \ldots, P_{\mathbf{i}}\}$ to be a patch model, and associate the individual patches with agents (assumed to be the loci of all primitive information) with a *representation function*:

$$h : \mathcal{A} \rightarrow M \equiv \{P_1, \ldots, P_{\mathbf{i}}\}. \tag{1.5}$$

Next, consider information relevant to decision-making that is *not* based on primitive measured data. This can be of two sorts. First, a decision-maker may anticipate more than one predicted evolution of the situation. Second, a decision-maker may also maintain a collection of possible courses of action, even if we anticipate only one evolving scenario. We must therefore maintain multiple *possible world* models. These models are all patch models defined exactly as above, with a specific model designated as the current working hypothesis model at any given time, and used for control. Such a collection of models is called a *view*:

$$V = \{M_1, \ldots, M_{\mathbf{k}}\} \tag{1.6}$$

A view is an information structure that represents both actual and possible scenarios and is designed to support the sort of "what-if" reasoning that is required in more complex decision-making. Finally, we may conceptualize an entire command and control system, comprising several decision makers, as a system of interconnected views, $\{V_1, V_2, \ldots, V_n\}$ for all n decision makers in the hierarchy. Both humans and artificial decision-makers observe and influence the battlefield through computer-maintained views comprising patch models. The topology of interconnection of this set of views describes the architecture of the command and control system.

Patch models, as described, are an enabling mechanism that can be used in a great variety of ways, once appropriate software has been designed to computationally represent and manipulate them. We have developed a simulation environment for distributed command and control of simulated point vehicles called *Patchworks*. A screen shot of this software is shown in Fig. 1.2. The software constructs visual renderings of a patch model, allowing humans to reason about and manipulate the models visually. Artificial decision-makers must directly work with the data structures comprising the model. A thorough overview of the features and capabilities of this software, illustrated through a detailed combat mission simulation, can be found elsewhere (Rao and D'Andrea 2006).

Since the purpose of this chapter is to understand fundamental themes and important open problems, we restrict ourselves to three particularly illuminating lines of investigation in the next three sections: realization and verification, decision-making, and hierarchical control . These three areas contain a rich set of open problems that require further investigation, and the purpose of these sections is to motivate these problems through results that have already been obtained, indicate how they might be formally posed, and suggest promising approaches.

1.3 Realization and Verification

Given a patch model formally set up as described in the previous section, we require mechanisms for controlling agent actions through the models and verifying the models through observations of primitive variables. These are the *realization* and *verification* problems respectively.

Realization is the problem of finding a path for an agent that maintains the truth of the the patch representing it in the model maintained by the controlling decision-maker. Call this patch the *ego patch*. Our current architecture supports only single ego patches for simplicity, but multiple ego patches can be supported at the cost of greater software complexity. The elements of the realization problem are illustrated in Fig. 1.1. The path indicated for the vehicle must avoid small obstacles while staying within the instantaneous position of the patch. A weaker form of control requires the vehicle to stay with the trace of the patch, and within the instantaneous patches only at end times, thus allowing in-mission delays. These problems are referred to as differential and integral patch realization problems respectively, and practical methods for solving them are discussed in detail in Rao et al. (2006). Realization problems are standard robotics path-planning problems with time-varying state constraints, but the special structure of the problem leads to many useful heuristics for computational efficiency. We do not discuss the basic problem of realization further in this overview, since the techniques are relatively well-known. We discuss briefly the problem of *hierarchical*

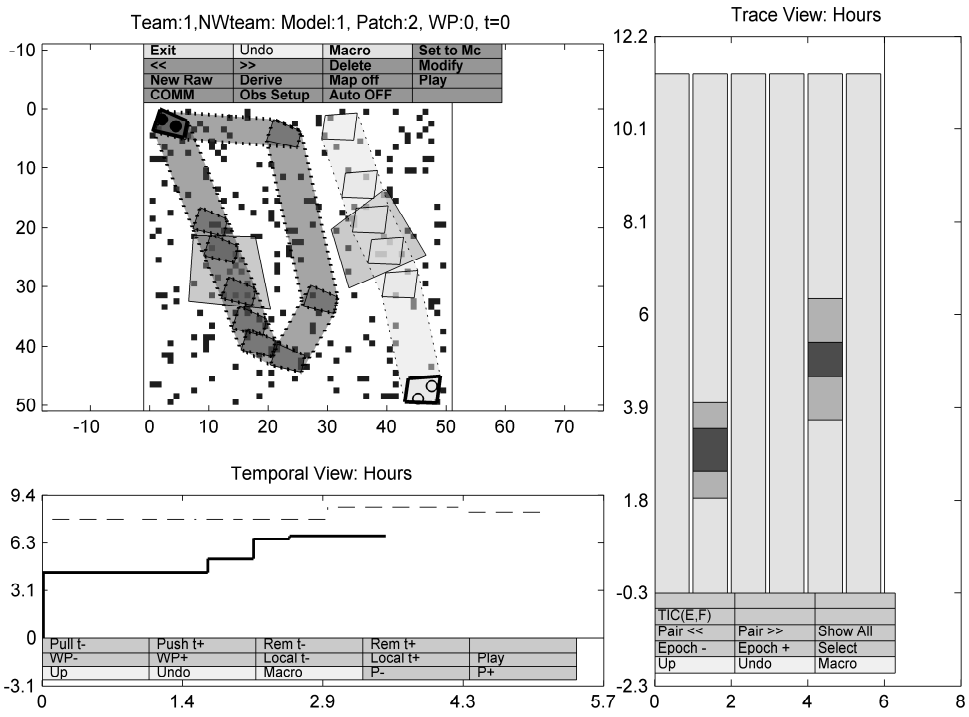

Figure 1.2 *Patchworks:* a simulation environment based on patch models. The top left pane shows a spatial view of four patches; the bottom left pane shows the velocities of the two moving patches. The right pane shows the qualitative state vector and is described in Section 1.5

realization, which is significantly different and an open problem.

Consider the situation where an agent is controlled by a hierarchy of supervisors through a reporting chain, rather than through a single ego patch. This corresponds to a hierarchy of patch models, where each model has a *unique* parent model. Supervisors achieve control by editing patches in their models, with changes being cascaded down the hierarchy (as constraints on the corresponding patches in lower-level models). More intuitively, we can consider patches in lower-level models to be virtual agents themselves, being controlled by higher-level models. The hierarchy of patches representing the agents controlled by the ego patch of the lowest-level model must satisfy a consistency condition defined as follows. Let $M_0, M_1, M_2, \ldots, M_k$ be a hierarchy of k models, with M_0 being the ground-level model and $P_e \in M_0$ being the ego patch. Let $\mathcal{A}_e = h_0^{-1}(P_e)$ be the set of agents controlled by P_e. Then, for all t:

$$h_0(\mathcal{A}_e) \subset h_1(\mathcal{A}_e) \subset \ldots \subset h_k(\mathcal{A}_e), \tag{1.7}$$

where the sequence of inclusions correspond to geometric inclusion in space-time. Note that the condition involves proper inclusion, since if a downstream patch is equal to an upstream patch and one assumes that supervising models may be changed continuously by a decision-maker, one would require a physically impossible zero-lag connection between the two models to maintain model truth at all levels.

It might appear that the hierarchical version of the problem is no more than a series of progressively constrained path planning problems. There is a crucial difference however: the "virtual agents" or intermediate patches being non-physical entities allows them to move in more unconstrained ways than physical agents. In particular, they can move "through" obstacles below a particular size, and through other patches, under appropriate conditions. This makes their motion much less constrained than motion in the robotics paradigm, where rigid-body obstacle avoidance is central. At this stage, solving this type of under-constrained path planning problem this remains an interesting open problem. Techniques drawn from fluid dynamics or optics, rather than robotics, and based on metaphors such as friction, viscosity and refraction, applied to patches, may be the appropriate ones.

Now consider the inverse problem of patch verification. Recall that decentralization assumptions must be modeled as constraints on the observations available to a given decision-maker. We require, however, a more general notion than observability as it is typically understood in control theory. Observability is a useful conceptual model for a static set of information-availability patterns – a given state variable is either available or not available to a decision-maker based on a single "observability" characteristic of that state within the system. When a state may be observed through a set of non-sequential, variably delayed, sparsely scattered (in time) and noisy measurements, a more appropriate notion is that of the "information pattern" introduced by Witsenhausen (Witsenhausen 1971) and used extensively in the non-linear stochastic control literature. Within this framework, we do not attempt to work through a separate representation of observation capabilities (such as the observation matrix, C, used in continuous LTI theory). We work, instead, directly with measurement sets. Introducing this approach is beyond the scope of this article, so we restrict ourselves to an example that illustrates the underlying principles and resultant problem formulations.

Let, at time t, a set $Y_{ij}(t)$ of observations of agent i be available to decision-maker j. Assume that j knows that the maximum velocity of i is v_i. Let $P(t)$ be the patch representing

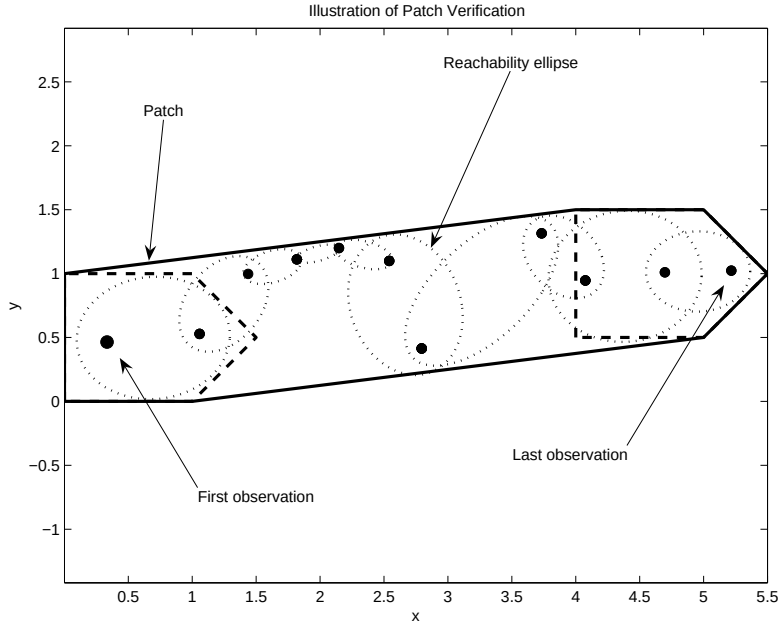

Figure 1.3 Patch Verification with noisy, intermittent, non-sequential observations

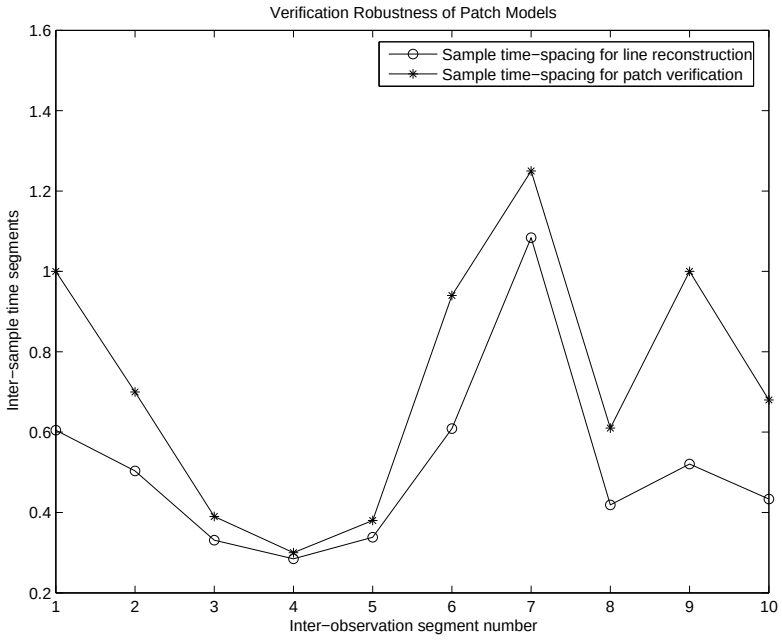

Figure 1.4 Interaction of patch geometry and sampling intervals

i in the patch model maintained by j. We suppose Y to be of the general form:

$$Y(t) = \begin{bmatrix} t_1 & x_1 & y_1 \\ t_2 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ t_k & x_k & y_k \\ \vdots & \vdots & \vdots \\ t_n & x_n & y_n \end{bmatrix}, \tag{1.8}$$

where $t_k < t$ and the x and y variables are two-dimensional location coordinates. Let $t_{\min} = \min\{t_1, \ldots, t_n\}$ and $t_{\max} = \max\{t_1, \ldots, t_n\}$. Given this data, we wish to verify that

$$(x(t), y(t)) \in P(t) \text{ for all } t \in (t_{\min}, t_{\max}). \tag{1.9}$$

We may also be interested in the weaker form of verification given by the conditions:

$$\begin{aligned} (x(t), y(t)) &\in P([t_{\min}, t]) \text{ for all } t \in (t_{\min}, t_{\max}). \\ (x(t_{\min}), y(t_{\min})) &\in P(t_{\min}) \\ (x(t_{\max}), y(t_{\max})) &\in P(t_{\max}) \end{aligned} \tag{1.10}$$

The former condition asserts that the agent stayed within the instantaneous patch location at all times, while the latter asserts that the agent stayed within the trace of the instantaneous moving patch in the interior of its trajectory, and within the instantaneous patches at the first and last observations. The latter set of conditions accommodates in-transit delays and speed-ups so long as they are compensated for. We will focus on the latter since it is geometrically more intuitive. The former case merely leads to faster sampling requirements.

Figure 1.3 shows a set of 11 observations corresponding to a moving agent represented by the pentagonal patch. To verify conditions 1.10 we require that the reachable regions between observations be contained with the trace of the patch (the long band between the start and end pentagon positions). Some straightforward geometry reveals that the reachable set for agent i between consecutive observations is an elliptical region with foci at the observation locations, (x_k, y_k) and (x_{k+1}, y_{k+1}), and semi-major axis given by $v_i(t_{k+1} - t_k)/2$ (for noisy observations, the reachable set is the union of a set of ellipses whose foci are at the boundaries of the uncertainty regions). If, as shown, all 10 of the ellipses obtained between the observations lies within $P([t_{\min}, t_{\max}])$, one can conclude that the first of the conditions in 1.10 is satisfied, with the other two conditions being trivial to check.

Now note a feature of Fig. 1.3: all 10 ellipses are tangent to the patch trace boundary. The trajectory and sampling points were constructed specially to achieve this condition, which can be interpreted usefully by asking the following question. Given a patch and a set of spatial sampling locations on the trajectory (the last two columns of Eqn. 1.8), what sets of corresponding observation times lead to verification and non-verification respectively? More generally, given that we know the maximum agent velocity (v_i), how closely can we reconstruct its trajectory using a finite set of samples? One extreme is easy to identify. If the sample locations and time are related by:

$$||(x_{k+1}, y_{k+1}) - (x_k, y_k)||_2 = v_i(t_{k+1} - t_k) \tag{1.11}$$

it is clear that only the straight-line paths between observation points could have been taken by the agent. The reachability ellipses in this case collapse to a poly-line. The set of observation times corresponding to this condition is then, in a sense, a maximal set of samples for the time period, since it allows full recovery of the path, with further sampling being unnecessary if no noise is present. The condition illustrated in Fig. 1.3 is the other extreme. Given the spatial observation locations, the sequence of ellipses that are *just* contained within the patch trace correspond to the minimal set of samples for the time period (or the slowest average sampling). In other words, were the samples any further apart in time, the patch would not be verifiable. Another useful way to understand this extreme is to note that if a patch does not tangentially touch all reachability sets, it is either less precise than it could be or more precise than the data warrants.

Fig. 1.4 illustrates this relation between the minimal and maximal time-separations between observations for the set of spatial sampling locations and patch geometry in Fig. 1.3. Where the two curves are highly separated, patch verification is much easier than full trajectory reconstruction. Where the two curves are close or touch, patch verification requires sampling of the same order as is required for trajectory reconstruction.

We have assumed in this discussion that sampling locations are provided, and considered the implications of various associated time vectors. The interpretation here is that given the same spatial geometry of the patch and path, the validity of the patch with respect to slower moving vehicles (longer inter-observation times) is easier to verify than that of faster vehicles moving along the same path. For faster vehicles, one must either observe at more points along the path, in order to shrink the reachability ellipses, or expand the patches themselves. Many interesting lines of investigation emerge from the phenomenology of patch verification. For instance, one could pose optimization problems to select the fewest possible sampling locations for a given patch and path, or alternately, attempt to find the smallest patch (in the sense of a metric such as area, aspect ratio, or number of waypatches) that can be verified with a set of observations considered *a posteriori*. In a real-time setting, we may also pose the problem of where and when to request an observation from an available surveillance resource in order to verify an *a priori* model that extends further in time than available data (i.e. is predictive with respect to observed entities, or imperative with respect to a controlled resource). At a more theoretical level, an interesting question is: what is the relationship between the sampling phenomenology discussed here, and sampling theorems of various sorts.

Next, we continue the development of these ideas by asking, given a sound representational framework of patch models and adequate realization and verification mechanisms, how can we actually use these models for control?

1.4 Human and Artificial Decision-Making

Human decision-makers can apply powerful visual cognition mechanisms to analyze situations, form intentions and issue imperatives through visual manipulation of patch models via GUIs such as in Fig. 1.2. It is reasonable to assume that patch models represented through appropriate software usefully, if reductively, mirror the corresponding spatio-temporal mental models (Endsley 2000), or "situation awareness" that provides the basis for human decision-making. For humans therefore, patch models can be a low-friction mechanism for quickly

capturing spatio-temporal beliefs, objectives, ideas and intentions with respect to controlled or observed entities. Given sufficiently powerful realization and verification mechanisms, these intentions can also be directly translated into action. The important open problems concerning the human aspect of patch-model based decision making are beyond the scope of this overview, since they involve issues in cognitive psychology.

The construction of artificial decision-makers capable of working with patch models, whether they are labeled AI systems or software agents, presents an interesting challenge. Since such systems do not have the powerful visual faculties that humans do, they must work directly with patch models in the form of data structures. Automated geometric or diagrammatic reasoning of the complexity that patch models support, unfortunately, is much too difficult for state-of-the-art AI techniques to handle (Sacks and Doyle (Sacks and Doyle 1992) provide a useful discussion of the issues). AI-based techniques are largely limited to automated reasoning and planning (Traverso et al. 2004) within the confines of first-order predicate logic, over finite ranges of discretized variables. Automated reasoning about models (such as patch models) constructed with geometric primitives requires reduction of these models to logic-based declarative models.

The tools we use for achieving this reduction are a pair of axiomatic systems for representing time and space respectively, and developed in the AI community. The temporal interval calculus (TIC) (Allen 1983, 1991) represents time through a time-segment primitive, $\tau = [t_1, t_2]$, rather than as a point-set, and analyzes temporal relations through a set of 13 primitive *qualitative* dyadic relations between two time intervals, $c(\tau_1, \tau_2)$, capturing conditions ranging from "before" through "during" to "after." A related system for representing primitive *regions* in space (simply-connected convex sets in the simplest case), the region connection calculus (RCC) (Randell et al. 1992) has also been developed, and comprises 8 primitive dyadic relations, $C(A, B)$, ranging from "disconnected from" through "partially overlapping" to "proper part" (the fewer primitive relations result from the absence of a past/future asymmetry in space).

Using TIC and RCC, one can construct from a patch model M a qualitative state vector, $\bar{X}(t)$, whose elements are the pairwise instantaneous region-connection states, $C(P_i(t), P_j(t))$, of all the instantaneous patches in a model, M. Each element of this vector evolves in a piece-wise constant manner. Each such piece-wise constant segment is called an epoch, and the interaction of epochs can be modeled with TIC. The right pane in 1.2 shows the qualitative state for a finite future window in time. Each vertical bar is an element of the state, and each uniformly-shaded portion of each bar is an epoch.

The qualitative state $\bar{X}$ over a window of time captures the condition of a patch model (with loss of quantitative geometric information) and is sufficient to support very flexible kinds of planning and scheduling. Here we consider a general reasoning problem: that of generating behavior rules for instantaneous patch motion that lead to a desired qualitative state. Since humans manipulate patch models to achieve control over agents primarily by editing the spatio-temporal geometry of patches, automation of this capability is fundamental to the construction of artificial decision makers.

The behavior design problem for a set of patches is defined as follows. Let M be a patch model, containing instantaneous patches, $P(t_0), \ldots, P_n(t_0)$ at t_0. For simplicity, replace the continuous time variable t with the discrete time index k and assume that the geometric state of the model is governed by:

$$\bar{x}_i\,(k+1) \equiv \bar{x}_i\,(k) + \bar{v}_i\,(k) \tag{1.12}$$

where $\bar{x}_i(k) = (x_i(k), y_i(k))$ is the mean of the vertex coordinates of $P_i(k)$ and $\bar{v}_i$, the velocity, is the control variable. For this system of evolving patches, we specify goals via an equivalence class of qualitative goal states, $\mathcal{X}^* \equiv \{\bar{X}_1^*, \bar{X}_2^*, \ldots, \bar{X}_m^*\}$. This allows us to formally state the behavior rule design problem:

Behavior Rule Design Problem: Determine a set of behavior rules $g_1, g_2, \ldots, g_n$ for the n agents, such that $\bar{v}_i(k) = g_i\left(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n, \bar{X}, k\right)$ and

$$\lim_{k \to k^*} \bar{X}_i(k) = \bar{X}_i^* \tag{1.13}$$

for some $\bar{X}_i^* \in \mathcal{X}^*$ and some $k^* < \infty$.

We note four features of this formulation. First, we note that as defined above, the goal set $\mathcal{X}$ may allow geometric configurations that are not intended. Second, we note that the reason an equivalence class of states is required, rather than a single goal state $\bar{X}^*$, is that in general we may not want to distinguish among different ways of achieving the same geometry (for example, if a configuration of a ring of n polygons is required, we may be agnostic with respect to the $(n-1)!$ different ways of realizing it). Third, we note that even though the *goals* are formulated in terms of $\bar{X}$, the behavior *rules* are allowed to depend on quantitative (position/velocity) variables. Forcing the rules to be functions of only the qualitative state ($v_i = g_i(\bar{X})$) in general leads to a very impoverished design space. Finally, note that the formulation only concerns the achievement of an *instantaneous* state represented through RCC. Automated manipulation along the temporal dimension alone is also possible (and somewhat simpler), and an example of this, involving rescheduling two patches (adjusting waypatch times) to restore a violated TIC constraint, is in Rao and D'Andrea (2006). Simultaneously achieving non-trivial spatial and temporal configurations, however, is a much harder problem. The behavior rule design problem is, in the general case, equivalent to theorem proving with joint RCC and TIC descriptions. Reasoning with RCC and TIC in general is known to be undecidable and reasoning with decidable subsets can be NP-complete Cohn and Hazarika (2001). Developing tractable reasoning mechanisms with a subset useful for decision-support with patch models is therefore an important open problem.

Here we illustrate the general features of the problem with an example. Details of this form of behavior design through both specially-constructed rules and general-purpose computational mechanisms such as simulated annealing is discussed in Goldfarb et al. (2006)

1.4.1 Example: The Surround Behavior

The intent of the behavior "surround" is to achieve a condition where all terminal instantaneous patches completely surround a static target patch P_T. The desired geometric configuration is an annular configuration of shapes along the perimeter of P_T, as shown in Figure 1.5. We attempt to model this intent as follows. For the set of instantaneous patches $P_1(k), P_2(k), \ldots P_n(k)$, we require that every $P_i, i \in [1, n], P_i \neq P_T$, overlap the target patch and two other patches (all non-disconnected relations being PO, or partially overlapping, in RCC terms). The equivalence class of goal states thus defined includes what is intended: every configuration that satisfies the geometric intent. A little thought will show, however, that the goal requirement lets in other unintended configurations as well. This is the case in general, and cannot be avoided, since it is an effect of the reductive nature of the representation with respect to the patch model.

In Basic Surround, each polygon has *at least* two neighbors and is in relation PO with target polygon P_T. In Refined Surround, each polygon has *exactly* two neighbors and is in relation PO with target polygon P_T. We omit the formal definition of the goal state equivalence class and the associated conditions on the qualitative state vector X for brevity.

We solved the problem using flocking rules designed specifically for this problem (details are described in Goldfarb et al. (2006)). The successful configuration shown in Fig. 1.5 is a an element $\bar{X}^*$ of the solution set $\mathcal{X}^*$, and this configuration was frequently achieved with appropriate tuning. An alternative undesirable configuration, admitted by the definition of the goal state for Basic Surround, is also shown in the Figure 1.5, and can also appear in simulation.

1.5 Hierarchical Control

Consider now the problem of multiple hierarchically-organized decision-makers attempting to cooperatively observe and/or control a set of vehicles through patch models. The set of vehicles is partitioned into a set of teams, each of which is controlled (or in the case of enemy surveillance, tracked) by the ego patch $P_e(t)$ of exactly one patch model M corresponding to one decision maker, the *ground level* decision-maker. Assume, for the sake of simplicity that all decision-makers have similar patch models (i.e. they partition the agent set in the same way, such that there is one to one correspondence between the patches representing a given pair of agents for any pair of models. In the notation of 1.2, $h_m(q_i) = h_m(q_j) \Leftrightarrow h_n(q_i) = h_n(q_j)$ for any pair of models, M_m and M_n and any pair of agents q_i and q_j).

For a particular team, comprising, say agents $\{q_1, q_2\}$, let this decision-maker be G. Now let G maintain a reporting relation with respect to a parent decision-maker, designated $Par(G)$, defined as follows. For agents $\{q_1, q_2\}$, G periodically informs $Par(G)$ of their location by sharing its current ego patch, $P(t)$. For all agents *except* $\{q_1, q_2\}$, $Par(G)$ provides G with updated positions by sharing the relevant patches.

If patches are being updated in real time, to reflect unpredictable movements, the patch representing $\{q_1, q_2\}$ in $Par(G)$ must contain P_e. In fact, if $Par(G)$ is to maintain a robust model of the locations of $\{q_1, q_2\}$, the inclusion must be *strict*, with a tolerance dependent on the frequency of updates and the maximum velocities of the vehicles. If this strict inclusion is maintained with sufficiently large tolerances throughout the reporting hierarchy, we can go even further and note that the rate at which $Par(G)$ reports to *its* parent, $Par(Par(G))$, with updates on the positions of $\{q_1, q_2\}$, is lower than the frequency with which it *receives* such reports. This is not surprising, since given strict inclusions, one may expect that some updates in lower-level patches will not violate the strict inclusion condition, and therefore require no perturbation of the parent model at all, leading to that particular update propagating no further.

Before we proceed further, it will be useful to introduce an example that we will use as a running example. The situation is depicted in Figure 1.6 as follows. Sixteen vehicles, divided into eight teams of two each, are being controlled by a hierarchy of 15 decision makers. There are eight ground level decision makers, four one level above, two at the third level and one top-level decision maker. In Figure 1.7, the eight views arranged in the outer rim of the square correspond to ground views, while the central view is that of the top-level decision-maker. In this case, the strict inclusion conditions were satisfied by constructing the initial

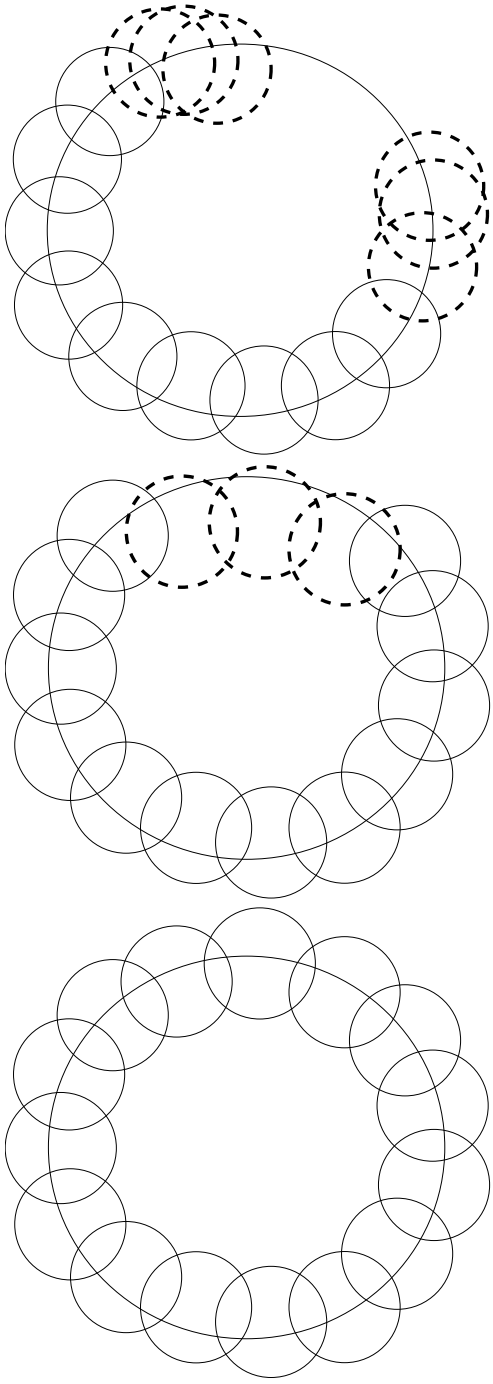

Figure 1.5 Comparison of solution states for Basic Surround (top), Refined Surround (middle), and the intended configuration (bottom). Both Basic and Refined Surround constraints require that each agent be in relation PO with the target. Basic Surround constraints also require that each agent be in relation PO with $at\ least$ two other agents, and Refined Surround constraints require that each agent be in relation PO with $exactly$ two other agents.

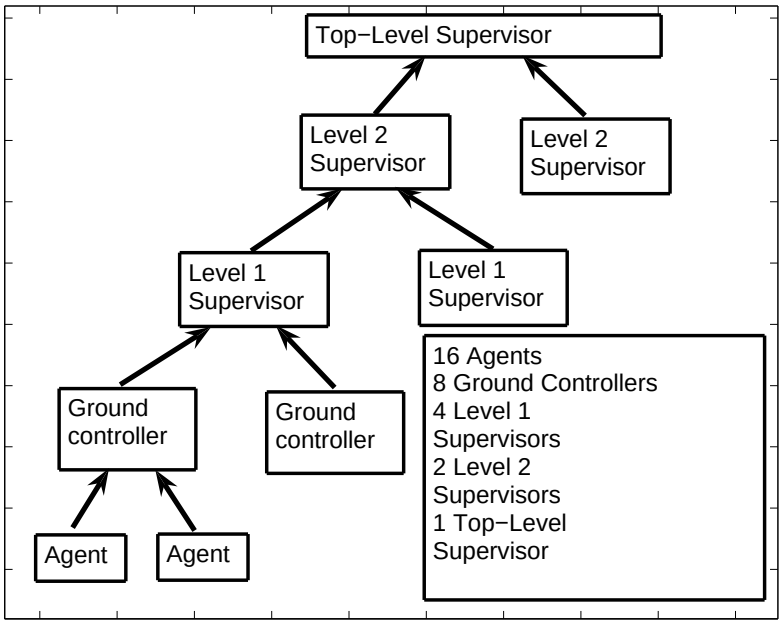

Figure 1.6 Hierarchy of decision-makers in example of 16-vehicle binary reporting tree

instantaneous circular patches as follows. The radius of the patch representing team j in the model at node i is given by:

$$r(i,j) = r_0 \lambda^{K(i,j)},$$

$$K = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 0 & 0 & -2 & -2 & -2 & -2 \\ 0 & 0 & 2 & 2 & -2 & -2 & -2 & -2 \\ -2 & -2 & -2 & -2 & 2 & 2 & 0 & 0 \\ -2 & -2 & -2 & -2 & 0 & 0 & 2 & 2 \\ 3 & 1 & -1 & -1 & -3 & -3 & -3 & -3 \\ 1 & 3 & -1 & -1 & -3 & -3 & -3 & -3 \\ -1 & -1 & 3 & 1 & -3 & -3 & -3 & -3 \\ -1 & -1 & 1 & 3 & -3 & -3 & -3 & -3 \\ -3 & -3 & -3 & -3 & 3 & 1 & -1 & -1 \\ -3 & -3 & -3 & -3 & 1 & 3 & -1 & -1 \\ -3 & -3 & -3 & -3 & -1 & -1 & 3 & 1 \\ -3 & -3 & -3 & -3 & -1 & -1 & 1 & 3 \end{bmatrix} \tag{1.14}$$

For a strict hierarchy of k levels, m ground-level agents and exactly p reports per decision-maker (p physical agents for ground-level decision makers), and a total of N decision makers,

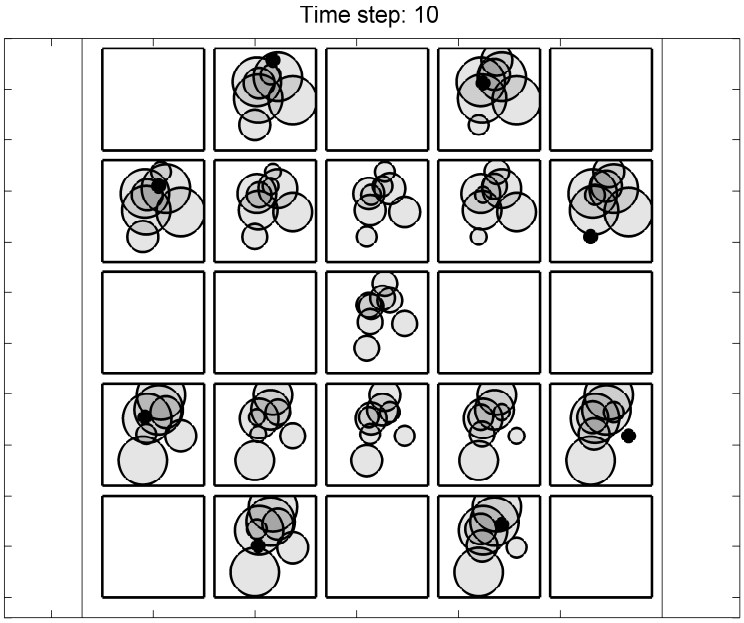

Figure 1.7 Views of all 15 decision makers. The central view is that of the top-level supervisor; the ones above and below it level 2 views; the four placed diagonally around the center are level 1 views, and the 8 on the rim are ground views. Dark patches in the outer rim are ego patches in ground-level models

the following relations must hold:

$$p^{k+1} = m,$$
$$k = \log_p m - 1, \tag{1.15}$$
$$N = \sum_{i=0}^{k} p^i. \tag{1.16}$$

In general, we may conjecture that the number of levels in the hierarchy will be in a logarithmic relationship with the number of agents being commanded, with the base of the logarithm being of the order of the average number of reports per decision-maker. If the degree distribution in the reporting hierarchy is other than a narrow normal distribution, however (such as a scale-free or small-world distribution), it is not clear what forms these relationships can assume.

For our example, the nominal command for each team is to stay in its initial position, but we imagine that they are all moving to evade enemy pursuers, necessitating updates in models propagating bottom-up and then top-down all through the hierarchy. We imagine, further, that all patches are circular at any given instant, and imagine that patches are updated whenever a new observation comes in, by adding a waypatch designed to maintain verifiability as discussed in Section 1.3. The circles representing a given pair of agents therefore must increase in radius upstream through the chain of parent decision makers, and then *continue* to increase down from the top of the hierarchy towards *other* ground level decision makers. The precision of representation therefore, decreases monotonically in the network as a function of degrees of separation from the ground-level reporting node.

Now imagine that all sensor signals are lost at time $t = 0$, and consider what must happen next in order to maintain the truth of all models for $t > 0$. This is a simple way to probe the effect of increasingly fast domain dynamics and external events. Clearly, since patch models are a conservative, worst-case representation, the ground level instantaneous patches at the last observation instant must be grown at a rate of at least $\dot{r}(t) = v_{\max}$, where r is the radius of a particular ego patch, and $v_{\max}$ is the maximum vehicle/agent velocity. Patches that are further away in the reporting web must also grow at a similar rate to maintain the inclusion relation. We now analyze the evolution of the 15 models in this example over time from an information-content perspective.

1.5.1 Information Content and Situation Awareness

How do we quantify the information content of a patch model? This question must be addressed before we approach such interesting matters as the relative situation awareness at various levels in the hierarchy. Consider the case of point agents, with spatio-temporal location being the only state of interest. In this case, a decision-maker may be said to possess complete information about an agent if the patch representing a given agent collapses to a one-dimensional curve coincident with the agent's trajectory in space-time. At the other extreme, if an agent is represented by a patch that is larger than the overall global area (or universal set) of interest (in this case the square "battlefield" represented in each of the non-empty views in Fig. 1.7, to which all activity is restricted), the patch contains no useful

information at all, since the information that the agent is in the battlefield at all times may be considered *a priori* and tautological. In between, so long as a patch does not cover the universal set, the patch contains *some* information. This notion of information content is clearly restricted to spatio-temporal location information and assumes the existence of a suitable bounded universal set to which all agents are known to be restricted. In all realistic cases, such a set will exist (though it might be as large as the entire surface of the earth).

Relating this notion to a formal information measure such as Shannon entropy is nontrivial, and remains an open problem. A measure called *residual area* is, however, very useful within the context of patch models, and we analyze the example from the previous subsection in terms of this measure. We define the residual area as:

$$r_A(P(t)) = 1 - ||\Sigma \cap (\Sigma - P(t))||/||\Sigma||. \tag{1.17}$$

Note that the measure is an instantaneous one. The measure is the normalized area of the portion of the set difference that is in the universal set, Σ (the norm notation indicating area). Note that strictly speaking, Σ is not a universal set in the usual sense, since parts of a patch can be outside it. While this issue can be fixed with additional notation, we will restrict ourselves to the present notation for clarity, since it does not cause confusion for the limited discussion presented here.

Now, let us analyze the information content in the various models in the example over time. First, consider the total information at a given instant in time. Figure 1.8 shows the residual areas for each of the 8 patches for each of the 15 decision makers at time $t = 90$, or approximately two-fifths of the way through the 240 time-step simulation. Decision-makers 8-15 are ground-level, 4-7 are level 1, 2-3 are level 2 and 1 is the top-level decision maker. Note the three lines of peaks. From left to right, these correspond to the content of the ground-level, level 1 and level 2 supervisors. The information content of the top-level supervisor corresponds to the line of partial peaks along the line for decision-maker 1. The left most line of peaks, forming a diagonal from decision-makers 8-15 and patches 1-8 shows that ground level decision makers have high information on their ego patches and almost no information elsewhere. The information content spreads out as one proceeds up the hierarchy. Figure 1.9 shows the same information in contour form, this time for 4 time slices at times $t = 10, 90, 160$ and 210. Note that information degrades first in the top decision-maker's view, and then progressively down the hierarchy, with the ground-level decision-makers being the last to lose all information. Note also that if one views the contour maps as a set of vertical slices of width 1, 2, 4 and 8, the amount of concentration of content along a diagonal in each slice indicates the degree of localization of information.

These two views of the evolution might suggest that as sensing and communication start to lag behind domain evolution, supervisors possess decreasing amounts of information. Consider an alternate view of the example. We sum the residual areas in each model and normalize, to get at the *total* information content per model. We then average the information content across all models at a given level in the hierarchy. The result is a measure of total information content at a given level. Figure 1.10 plots these quantities over time. Note the interesting behavior of the four curves: their *order*, defined as the order of ordinates at a given abscissa (time) point is completely reversed during the mission. This suggests that during epochs when sensing and reporting can keep up with the domain dynamics, supervisors do in fact possess more *total* information and are thus enabled to make better broad decisions. Once information degrades below a particular point, however, reports know more than supervisors.

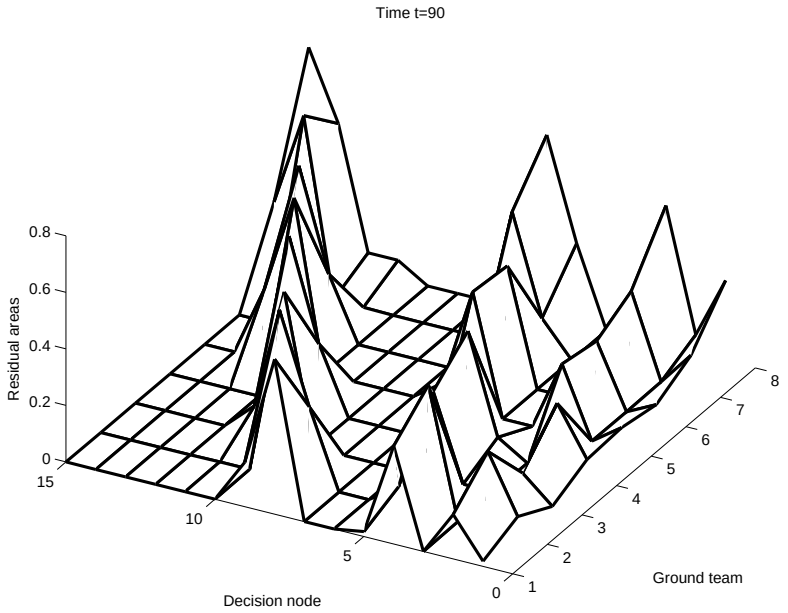

Figure 1.8 Information content in hierarchy, measured through residual areas

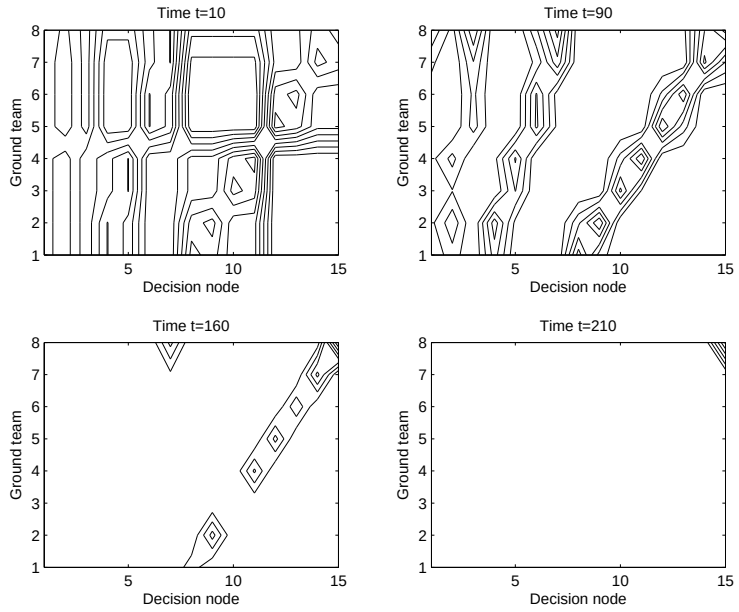

Figure 1.9 Contours of information content, measured by residual area, at 4 time points

If we interpret informed higher-level decision making as strategic, and lower-level decision-making as tactical, we can conclude that as things speed up, informed tactical responsiveness is retained longer than informed strategic responsiveness. Interpreted another way, this suggests that increasing sensing and reporting frequency by leveraging improved computing and communication increases the horizon of informed decision-making and responsiveness at *all* levels. If we assume that being better informed leads to decision making that is both *faster* and better, then this simulation is a demonstration of the validity of one of the key hypotheses of network-centric warfare (2001): that better networks enable better shared situation awareness (SSA), leading to faster responsiveness. Of course, this demonstration is an empirical one, and restricted, moreover, to the context of a very specific construction involving patch models. We can hope, however, that more generalized information models of C2 systems analyzed in a similar fashion might yield more general and powerful insights.

We conclude our dissection of this example with one final view of the data. Consider again the aggregate information evolution, but this time viewing each patch individually at a representative node at each level, yielding $4 \times 8 = 32$ curves. Figure 1.11 shows this evolution for the residual areas. The key point to note is the increasing asymmetry of the evolution of the information across patches as one goes down the hierarchy. This neatly captures an intuitive idea: that local information must degrade much more slowly than information that is distant in the reporting network. The value of having a supervisory role is that even though the total information may eventually be less than that of lower-level models, the *symmetry* is maintained through time. This balanced degradation may have some value in decision-making even beyond the point where the total information content starts to lag behind. Investigating this kind of "obsolete big picture" information and its uses is an open problem.

1.6 Discussion and Conclusions

The scenario we used to motivate this work in the introduction to the chapter, a version of the concept of operation referred to as *cooperative strike*, is in several ways, as old as warfare itself. An enemy threat is detected, a course of action selected, resources allocated and a mission executed. Though modern military terminology would describe this sequence of events as a "kill chain" (comprising *find, fix, track, target, engage* and *assess* stages), a very similar description might be applied to a bronze age battle involving chariots and archers.

The similarity is superficial primarily for one reason: the sheer scale and speed at which the decision-making is enabled and completed has no counterpart in pre-modern warfare. The technical enablers are reliable high-bandwidth communications in the battlefield, significant computing power at every decision node in the system and sophisticated software capabilities that are generally bucketed under the term "autonomy." Employed against an adversary without similar technology, the asymmetry in capabilities can be so vast that the resulting battlefield situations are unprecedented, even though one might say that the only change has been a speed up of the building block processes of battle. Information technology asymmetrically employed in the battlefield can lead to a discontinuous change in the nature of combat, even though the underlying changes in decision-making speed may be smooth (Albert et al. 2001). This observation has led to a new military doctrine called "Information Superiority" and a corresponding prescriptive concept of warfare called "network-centric warfare" (NCW)

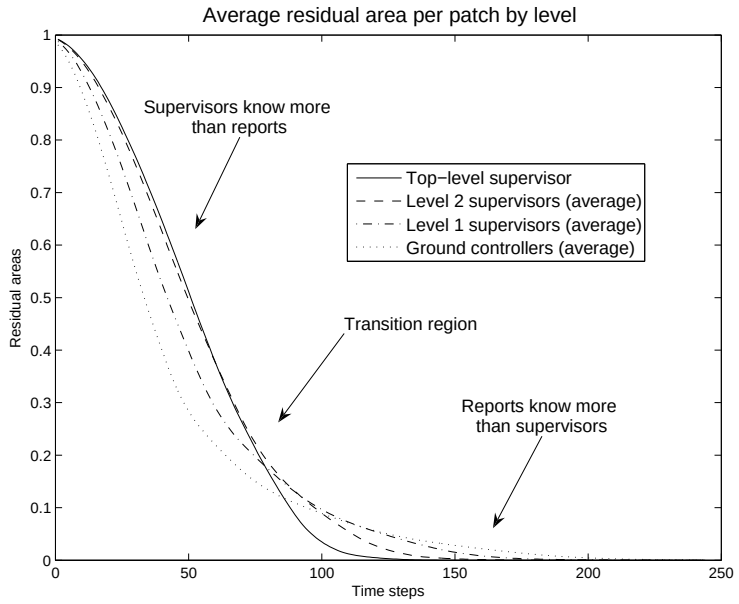

Figure 1.10 Relative information content at supervisory and ground level decision nodes over time. Note that initially, supervisors have more information, but it degrades faster than at lower levels.

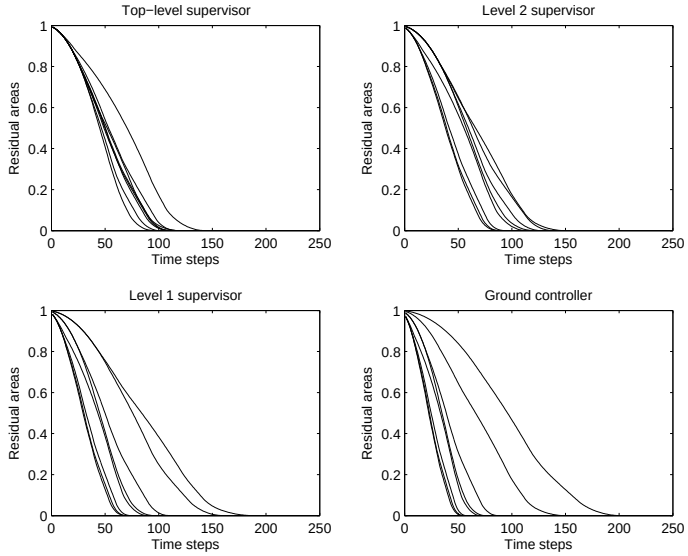

Figure 1.11 Asymmetric degradation of information at various levels. Note that lower level decision makers have greater asymmetry, leading to localization of knowledge, even though they retain useful knowledge longer.

(2001; Albert et al. 2000; Cebrowski 1999) . NCW as it is understood (and partially applied) today largely concerns the impact of communication and computing power on the battlefield. The impact of autonomy, however, is not yet fully understood, since autonomy technology is a software capability that still lags significantly behind the hardware capabilities. Autonomy, arguably the central element of this work, is a more delayed aspect of the impact of information technology, since it refers to capabilities manifested in advanced software, rather than improved hardware.

The typical problems of modern warfare tend to defy compact description. The work in this volume, for instance, was motivated by the unwieldy phrase, “cooperative control of distributed autonomous vehicles in adversarial environments” (often with the qualifying phrase, “with humans in the loop,” appended). The description is not redundant, however, and each element is integral to the intent of the description. The complexity of the description is simply a function of the all-encompassing nature of the source – information technology – of the most pressing modern research challenges (and opportunities). Since the evolution of information technology is impacting every aspect of military operations, the underlying research imperative is to reinvent the nature of warfare.

Patch models represent one approach – a representation-driven one – for managing the impact of information technology on combat technology. The sharable (among humans and artificial decision makers) and interoperable nature (among decision processes) of the models

enable clean, modular architectures, while their capacity for supporting contextual relevance-based abstractions make them suitable for managing information overload. While patch models themselves are a simplified framework, largely for dealing with simulated agents displaying a restricted subset of the behaviors that occur in battle, we expect that any larger-scale implementation effort will benefit from the principles that drove (and were validated by) this work.

Bibliography

2001 *Network-Centric Warfare: DoD report to Congress*. Department of Defense.

Albert DS, Garstka JJ and Stein FP 2000 *Network Centric Warfare: Developing and Leveraging Information Superiority (2nd Ed.)*. CCRP.

Albert DS, Garstka JJ, Hayes RE and Signori D 2001 *Understanding Information Age Warfare*. CCRP.

Allen JF 1983 Maintaining knowledge about temporal intervals. *Communications of the ACM* **26**(11), 832–843.

Allen JF 1991 Planning as temporal reasoning *Proc. Second International Conference on the Principles of Knowledge Representation and Reasoning*, Cambridge, MA.

Cebrowski AK 1999 Network-centric warfare: An emerging military response to the information age *Command and Control Research and Technology Symposium*.

Cohn AG and Hazarika SM 2001 Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae* **43**, 2–32.

Endsley MR 1995 Towards a theory of situation awareness in dynamic systems. *Human Factors* **37**(1), 32–64.

Endsley MR 2000 Situation models: an avenue to the modeling of mental models *Proc. 14th Triennial Congress of the Intl. Ergonomics Assoc.*

Goldfarb S, Rao VG and D'Andrea R 2006 Agent-based modeling with polygon primitives for aerospace applications *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Keystone, CO.

Nofi AA 2000 Defining and measuring shared situation awareness. Technical Report CRM D0002895.A1, Center for Naval Analysis, Alexandria, VA.

Randell D, Cui Z and Cohn AG 1992 A spatial logic based on regions and connection *Proc. 3rd Intl. Conf. on Knowledge Representation and Reasoning*.

Rao VG and D'Andrea R 2006 Patch models and their applications to multi-vehicle command and control *Proc. American Control Conference*, pp. 4958–4963, Minneapolis, MN.

Rao VG, Wongpiromsarn T, Ho T, Chung K and D'Andrea R 2006 Encapsulated motion planning for abstraction-based control of multivehicle systems *American Control Conference*, pp. 2995–3000, Minneapolis, MN.

Sacks EP and Doyle J 1992 Prolegomena to any future qualitative physics. *Computational Intelligence* **8**(2), 187–209.

Simon H 1996 *The Sciences of the Artificial (Third Edition)*. MIT Press, Cambridge, MA.

Traverso P, Ghallab M and Nau D 2004 *Automated Planning : Theory and Practice*. Morgan Kauffman.

Witsenhausen HS 1971 Separation of estimation and control for discrete time systems. *Proceedings of the IEEE* **59**(11), 1557–1565.