Toward Distributed Declarative Control of Networked Cyber-Physical Systems[⋆]

Mark-Oliver Stehr, Minyoung Kim, and Carolyn Talcott

SRI International
`stehr,mkim,clt@csl.sri.com`

Abstract. Networked Cyber-Physical Systems (NCPS) present many challenges that are not suitably addressed by existing distributed computing paradigms. They must be reactive and maintain an overall situation awareness that emerges from partial distributed knowledge. They must achieve system goals through local, asynchronous actions, using (distributed) control loops through which the environment provides essential feedback. Typical NCPS are open, dynamic, and heterogeneous in many dimensions, and often need to be rapidly instantiated and deployed for a given mission. To address these challenges, we pursue a declarative approach to provide an abstraction from the high complexity of NCPS and avoid error-prone and time-consuming low-level programming. A longer-term goal is to develop a distributed computational and logical foundation that supports a wide spectrum of system operation between autonomy and cooperation to adapt to resource constraints, in particular to limitations of computational, energy, and networking resources. Here, we present first steps toward a logical framework for NCPS that combines distributed reasoning and asynchronous control in space and time. The logical framework is based on partially ordered knowledge sharing, a distributed computing paradigm for loosely coupled systems that does not require continuous network connectivity. We illustrate our approach with a simulation prototype of our logical framework in the context of networked mobile robot teams that operate in an abstract instrumented cyber-physical space with sensors.

1 Introduction

A growing number and variety of devices can sense and affect their environment. Some are fairly simple, such as radio-frequency identification (RFID) access control, some quite sophisticated, such as mobile robots with localization and sensing capabilities. This opens up an opportunity for a new generation of *Networked Cyber-Physical Systems* (NCPS). Such systems can provide complex, situation-aware, and often critical services in applications such as distributed

[⋆] Support from National Science Foundation Grant 0932397 (A Logical Framework for Self-Optimizing Networked Cyber-Physical Systems) and Office of Naval Research Grant N00014-10-1-0365 (Principles and Foundations for Fractionated Networked Cyber-Physcial Systems) is gratefully acknowledged. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF or ONR.

surveillance, crisis response, medical systems, self-assembling structures or systems, networked space/satellite missions, or distributed critical infrastructure monitoring and control. General principles and tools are urgently needed for building robust, effective NCPS using individual cyber-physical devices as building blocks. In this paper, we present a declarative approach to NCPS based on a logical framework that supports distributed reasoning and can interact with the physical world asynchronously through observations and goal-oriented control.

Cyber-physical systems are often deployed in challenging environments with a wide spectrum of networking characteristics. They should be able to take advantage of opportunities for communication as they arise and must be robust in spite of delays and disruptions due to, for example, mobility, failures, or nodes entering and leaving the system. Topology changes (network partitioning in the extreme case) can happen continuously, so that even the common assumption of globally stable periods, which is crucial for many distributed algorithms, is not realistic. Hence, the logical framework is developed on top of a loosely coupled distributed computing model based on the *partially ordered knowledge-sharing* paradigm that has been used in our earlier work on disruption-tolerant networking [16]. Knowledge (as opposed to, say, a packet) is a semantically meaningful unit of information that can be stored, processed, aggregated, and communicated to other nodes. A unique feature of the knowledge-sharing model is that it is parameterized by an application-dependent partial order on knowledge that is available to all nodes and provides an abstraction of the knowledge semantics. In this model, the network topology can change continuously, communication can be unreliable, and no bounds are assumed on communication delays. As a consequence, any form of communication between nodes is acceptable. Similar to data mules for sensor networks [5] or message ferrying in delay- and disruption-tolerant networks [7], each node can cache knowledge for extended periods of time and hence can exploit the (possibly mobile) network dynamics to share knowledge without relying on an end-to-end path at any point in time.

The declarative view of NCPS enables us to recast information collection, control, and decision problems as logical problems that are primarily centered around the duality of two kinds of knowledge: facts and goals. Facts can represent sensor readings at specific locations and other information that is derived by possibly distributed computation. Goals can represent queries for information or requests to the system or individual components to perform certain actions. Although there are cases where a goal can be directly satisfied by a single action, it is more often the case that distributed actions are needed, sometimes in a coordinated manner that requires cooperation across multiple nodes. In our logical framework, both facts and goals will be treated on an equal footing together with corresponding reasoning rules. The distributed, dynamic, and interactive nature of the underlying systems is rarely considered in logical frameworks, which are traditionally designed as closed systems. Our framework is flexible enough to take into account the heterogeneous resources and capabilities at each node. Each node cooperates with its neighbors and interacts with its environment by sensing and affecting, driven by facts and goals. Overall system goals are refined to goals achievable by an individual node or device. Reasoning takes place locally at each node in the network as well as in cooperation (and competition) with other nodes. Seamless integration of cooperation and autonomy ensures that there is no need to rely on the existence or connectivity of other nodes, so that the local operation can always proceed, although possibly in a less optimal fashion. Solutions can be shared and composed oportunistically without being subject to any rigid or hierarchical flow constraints.

Recent applications of cognitive and more specifically declarative techniques in communication and networking, [4] being a noteworthy example, have attracted a lot of interest, but a declarative treatment of NCPS still remains a challenge. To study this problem we use *self-organizing mobile robots* as an example capturing many of the challenges of NCPS [5]. This example is inspired by previous work at SRI, in particular the Centibots project [14] that has developed a team-based hierarchical planning approach to accomplish a mission such as collaborative mapping, and the Commbots project [8], where the mission objective is to improve network connectivity by distributed control of robotic routers.

After presenting a simplified version of our distributed logical framework in Section 2, we illustrate in Section 3 the key ideas and a prototypical implementation by means of an abstract simulation of a networked mobile robot team operating in an instrumented cyber-physical space. Some related work will be summarized in Section 4.

2 Distributed Logic and Cyber-Inference

Declarative control aims at providing the user with a logical view of the cyber-physical system so that user objectives can be conveyed to the system, which then will make its best effort to realize those objectives. Such objectives are given in the context of the current system state (which is only approximately and partially observable). Furthermore, the user objectives can be part of a larger set of objectives (e.g., including system policies and objectives from other users), which we simply refer to as the system goal. Declarative control is the process of continuous adaptation of the system to transition to a state that satisfies the system goal, which in turn can continue to change based on feedback from the environment or because of new user requests.

The purpose of logic in this context is many-fold. First of all, it provides a language to express and communicate system goals. Dually, it allows expressing and communicating facts about the current system state. In both cases, communication includes communication with the users but also communication among the components of the system. At the level of an individual cyber-physical component, the logic provides a declarative interface for goal-oriented control and feedback through observations that are represented as logical facts. Finally, it provides a framework for inference and computation, which allows facts and goals to interact with each other and form new facts or goals.

Aiming at a solution to declarative control that covers the entire spectrum between cooperation and autonomy and makes opportunistic use of networking resources, it is clear that the logic needs to be inherently distributed. The opportunity to exchange knowledge with other nodes should lead to cooperation, and the absence of such opportunities should lead to more autonomous behavior. In the following we present a simple distributed inference system that accomplishes this goal. We use Horn clause logic to illustrate our approach, which we expect to generalize to more expressive logics.

For the following abstract logical treatment, we assume a networked cyber-physical system S with a finite set of cyber-nodes N. We assume a time-dependent network and environment model, in which two cyber-nodes have the capability to communicate (uni- or bidirectionally) whenever the network conditions admit it and where each cyber-node can have (not necessarily the same) sensors and actuators. The sensors can generate observations at arbitrary time points. The actuators are driven by goals, which they can either attempt to satisfy immediately or in a continuous asynchronous process with (partial) feedback provided through observations. Instead of imposing restrictive conditions on S we generally allow S to operate under arbitrary conditions. In the following, we use x and y to range over cyber-nodes and t to range over the time domain, for which we use natural numbers in this paper. For the following, we also assume a fixed signature Σ and a fixed finite theory Ω over Σ in Horn clause logic that is shared by all nodes of the cyber-physical system. We assume that Σ contains built-in constants for natural numbers and names of cyber-nodes. Additional *built-in functions*, and *built-in predicates* can be included in Σ. To account for the temporal and distributed character of cyber-physical systems, we assume that the signature Σ contains a distinguished set of predicates (distinct from built-ins) that we refer to as *cyber-predicates*, i.e., predicates that define the interface of the logic with the outside world (i.e., cyber-physical devices and users).

Using P and Q to range over atoms, we furthermore assume that all clauses in Ω are uniquely labeled and definite, i.e., of the form $l : P_1, \ldots, P_n \Rightarrow Q$ with a unique label l, where Q is not the application of a built-in predicate. For our proof system, we assume that $\Omega = \Omega_{\mathrm{f}} \cup \Omega_{\mathrm{b}}$, where Ω_{f} and Ω_{b} are sets of clauses that we refer to as *forward and backward clauses*, respectively.

The set of *facts* is simply defined as the set of all ground atoms. Hence, all predicates are allowed to occur in facts. The set of *goal predicates* is any set of predicates that includes at least the built-in predicates and all predicates that appear in the conclusion of a backward clause, i.e., a clause from Ω_{b}. Certain cyber-predicates can be included in this set if they are intended to form goals. For the purpose of this paper, the set of *goals* is simply the set of (not necessarily ground) atoms that are applications of goal predicates. A unit of *knowledge* can be either a fact or a goal. To simplify the presentation in this paper we assume that facts and goals can always be distinguished, even if they have the same underlying atom. Occasional conversions between a fact and a goal with the same atom should be clear from the context and will be left implicit. From now on, we will use K to range over units of knowledge and F and G to range over facts and goals, respectively.

A fact or goal that is the application of a built-in predicate is called a *built-in fact* or *built-in goal*, respectively. Given a built-in goal G, we say that $\sigma(G)$ is a *solution* of G iff there is a substitution σ such that $\sigma(G)$ is ground (hence can be viewed as a fact) and satisfied. All cyber-predicates p_c are explicitly time dependent and form atoms $p_c(t, \ldots)$, where t is a natural number denoting its timestamp, i.e., its time of creation at the creating node. A *cyber-fact* or *cyber-goal* is any fact or goal, respectively, of this form. Note that this does not preclude cyber-facts and cyber-goals from having additional temporal attributes or constraints that can be specified in the remaining arguments, e.g., the time of obervation in the case of a cyber-fact, or a deadline in the case of a cyber-goal.

To utilize the partially ordered knowledge-sharing model, we assume that the set of all knowledge units is equipped with a strict partial order $<$, called the *subsumption order*,[1] and a strict partial order $\prec$, called the *replacement order*, that extends $<$. The intended meaning of $K \prec K'$ is that K' replaces K, because K is semantically subsumed by K' (in this case we have $K < K'$), or because K is obsolete relative to K', e.g., out of date.

The logical state of a cyber-node is of the form $\Gamma \vdash \Delta\ @\ t, x$, where x is the unique name of the node, t is a natural number representing its local time, Γ is a finite set of facts, and Δ is a finite set of goals. It is worthwhile to point out that the interpretation of $\Gamma \vdash \Delta\ @\ t, x$ is nonstandard, and quite different from sequent calculus even without t, x. Intuitively, Γ is a set of facts that is continuously growing through observations or by inference, while Δ is a set of goals that the system is trying to satisfy without necessarily aiming to satisfy all of them. To reflect the reality of cyber-physical systems, inference is a continuous typically nonterminating process, and goals and facts can change at any time due to environment and user interaction. Since reasoning is a distributed process in space and time and the world can change during this process, an approach quite different from traditional work in formal specification, logic programming, or automated deduction is needed. We will also see that a goal does not have to be solved to be useful, but as a cyber-goal can still have an impact on the environment, which may be observable through cyber-facts.

A configuration of a cyber-physical system S is a set of local states $\Gamma \vdash \Delta\ @\ t, x$, one for each cyber-node x of S. Given a configuration c containing $\Gamma \vdash \Delta\ @\ t, x$, we write $\mathcal{F}_x(c)$ and $\mathcal{G}_x(c)$ to denote Γ and Δ, respectively. The proof system in Figure 1 defines a labeled transition relation $\rightarrow$ on configurations of the cyber-physical system S in the following sense: For configurations c and c', we have $c \rightarrow_r c'$ iff there exists an instance r of a proof rule such that c contains the premises of r, and c' is obtained by an update of c with the conclusion, i.e., by replacing $\Gamma \vdash \Delta\ @\ t, x$ by the conclusion $\Gamma' \vdash \Delta'\ @\ t', x$.

For readability, we have omitted an implicit side condition $t < t'$ in all proof rules ($t_x < t'_x$ in the communication rules), meaning that time increases monotonically at least by one unit in each step. Furthermore, we view each fact or goal as a singleton set and use the comma operator to denote set union. If the

[1] In the general model, the subsumption order is a quasi-order, but in this paper we identify knowledge units that are related by the induced equivalence relation.

comma operator is used in the premise of a proof rule, we always assume that it denotes the union of disjoint sets, i.e., Γ, K implies $K \notin \Gamma$ if it occurs in a premise. For a set $\mathcal{K}$ of knowledge units, we write $K \prec \mathcal{K}$ if there exist $K' \in \mathcal{K}$ such that $K \prec K'$, respectively. In the context of a proof rule that has a premise $\Gamma \vdash \Delta \ @ \ t, x$ we say that K is *fresh* (at x) if neither $K \in \Gamma, \Delta$ nor $K \prec \Gamma, \Delta$. In the condition of proof rules we use σ to range over all (not necessarily ground) substitutions that satisfy the condition of the proof rule. Some conditions in the reasoning rules will further restrict σ to most general substitutions.

To give some intuitive explanation of the proof rules in Figure 1, the control rule represents the addition of a new user-level objective to the set of system goals. The observation rule captures the generation of information from the environment. This can happen spontaneously or can be triggered by a goal that a cyber-device attempts to satisfy. The communication rules allow cyber-nodes to exchange facts or goals by means of asynchronous communication. The nature of communication, i.e., whether it is uni-/bidirectional, unicast, multicast, or broadcast remains unspecified. The replacement rules are used to overwrite subsumed and obsolete facts and goals. In this paper, we assume only a loose form of logical time synchronization that satisfies the minimal monotonicity requirement formulated in the communication rules. Of course, this does not preclude implementations with time synchronization that takes place even when no knowledge is exchanged. The forward and backward rules implement forward and backward reasoning. The first forward rule applies an instance of a Horn clause from the underlying theory if all conditions are available as facts, generating a new fact $\sigma(Q)$ corresponding to the conclusion in this process. The second forward rule covers the case where the available facts are not sufficient to apply the clause so that a new subgoal $\sigma(P_j)$ needs to be generated for a missing fact. The backward rules are analogous to the two forward rules, but apply a Horn clause in a goal-directed way, by first unifying the conclusion with an existing goal. Again, if the present facts are not sufficient to cover all conditions of a clause, a new subgoal is generated. In both forward and backward rules, the selection of facts, goals, and rules is entirely nondeterministic, and many strategies are sensible as long as they satisfy some local weak fairness requirements. Finally, the sleep rule allows the system to wait and hence slow down the reasoning for an arbitrary amount of time, e.g., to save energy, wait for new knowledge, or use the resources for other purposes.

The proof system is specifically designed to enable a distributed implementation using randomization techniques that can be used to satisfy the local fairness conditions with probability one. Facts and goals are represented as knowledge in the partially ordered knowledge-sharing model, where knowledge can be opportunistically exchanged across multiple hops and cached in the network until it is locally replaced by knowledge that is higher in the partial order. No global consistency is assumed or needed in this approach. There are several places in the proof system, where nondeterminism at a given node x can be implemented by a randomized choice. First, there is the selection of clauses in the four reasoning rules. Second, there is the selection of facts and/or goals to which the clause is

$$\frac{\Gamma \vdash \Delta \ @\ t, x}{\Gamma \vdash \Delta, G \ @\ t', x} \quad \text{if } G = p_c(t, \ldots) \text{ is a cyber-goal} \qquad \text{(Control)}$$

$$\frac{\Gamma \vdash \Delta \ @\ t, x}{\Gamma, F \vdash \Delta \ @\ t', x} \quad \text{if } F = p_c(t, \ldots) \text{ is a cyber-fact} \qquad \text{(Observation)}$$

$$\frac{\Gamma, F \vdash \Delta \ @\ t, x}{\Gamma \vdash \Delta \ @\ t', x} \quad \text{if } F \prec \Gamma, \Delta \qquad \text{(Replacement1)}$$

$$\frac{\Gamma \vdash \Delta, G \ @\ t, x}{\Gamma \vdash \Delta \ @\ t', x} \quad \text{if } G \prec \Gamma, \Delta \qquad \text{(Replacement2)}$$

$$\frac{\Gamma_x \vdash \Delta_x \ @\ t_x, x \quad \Gamma_y, F \vdash \Delta_y \ @\ t_y, y}{\Gamma_x, F \vdash \Delta_x \ @\ t'_x, x} \qquad \text{(Communication1)}$$

if $x \neq y$, $t'_x \geq t_y$, and F is fresh at x.

$$\frac{\Gamma_x \vdash \Delta_x \ @\ t_x, x \quad \Gamma_y \vdash \Delta_y, G \ @\ t_y, y}{\Gamma_x \vdash \Delta_x, G \ @\ t'_x, x} \qquad \text{(Communication2)}$$

if $x \neq y$, $t'_x \geq t_y$, and G is fresh at x

$$\frac{\Gamma \vdash \Delta, G \ @\ t, x}{\Gamma, \sigma(G) \vdash \Delta, G \ @\ t', x} \qquad \text{(Built-in)}$$

if G is a built-in goal with a solution $\sigma(G)$ such that $\sigma(G)$ is fresh.

$$\frac{\Gamma, \sigma(P_1), \ldots, \sigma(P_n) \vdash \Delta \ @\ t, x}{\Gamma, \sigma(P_1), \ldots, \sigma(P_n), \sigma(Q) \vdash \Delta \ @\ t', x} \qquad \text{(Forward1)}$$

if $l : P_1, \ldots, P_n \Rightarrow Q$ is a clause from Ω_{f},
and $\sigma(Q)$ is a fresh fact.

$$\frac{\Gamma, \sigma(P_1), \ldots, \sigma(P_{j-1}) \vdash \Delta \ @\ t, x}{\Gamma, \sigma(P_1), \ldots, \sigma(P_{j-1}) \vdash \Delta, \sigma(P_j) \ @\ t', x} \qquad \text{(Forward2)}$$

if $l : P_1, \ldots, P_n \Rightarrow Q$ is a clause from Ω_{f},
σ is a most general substitution, $\sigma(P_j)$ is a goal, and $\sigma(P_j)$ is fresh.

$$\frac{\Gamma, \sigma(P_1), \ldots, \sigma(P_n) \vdash \Delta, G' \ @\ t, x}{\Gamma, \sigma(P_1), \ldots, \sigma(P_n), \sigma(Q) \vdash \Delta, G' \ @\ t', x} \qquad \text{(Backward1)}$$

if $l : P_1, \ldots, P_n \Rightarrow Q$ is a clause from Ω_{b},
$\sigma(Q) = \sigma(G')$ is a fresh fact.

$$\frac{\Gamma, \sigma(P_1), \ldots, \sigma(P_{j-1}) \vdash \Delta, G' \ @\ t, x}{\Gamma, \sigma(P_1), \ldots, \sigma(P_{j-1}) \vdash \Delta, G', \sigma(P_j) \ @\ t', x} \qquad \text{(Backward2)}$$

if $l : P_1, \ldots, P_n \Rightarrow Q$ is a clause from Ω_{b},
$\sigma(P_j)$ is a fresh goal, σ is a most general substitution such that $\sigma(Q) = \sigma(G')$.

$$\frac{\Gamma \vdash \Delta \ @\ t, x}{\Gamma \vdash \Delta \ @\ t', x} \qquad \text{(Sleep)}$$

Fig. 1. Proof Rules of our Distributed Logical Framework for NCPS

applied. Third, there is the choice of the substitution and hence solution in the rule for built-ins. And finally, there is the sleep rule, which can be implemented by random waiting using a suitable distribution to make the reasoning process adaptive to resource constraints and network conditions.

3 Sample Application

To test our ideas we will focus on a specific application that we call *self-organizing mobile robots* as a special case of controllable networks that captures many interesting aspects of NCPS [5]. Consider a self-organizing network of mobile robots deployed in a building, e.g., to achieve situation awareness during an emergency. This is a challenging test case for various reasons. The network is highly dynamic, and temporary disconnections or failures are part of the normal operation and need to be compensated for by real-world actions. Parameters such as a robot's position can be controlled only indirectly via actions, and costs of changes (e.g., energy consumption) cannot be neglected.

As a concrete sample mission, we chose a primary goal such as delivery of the collected information (e.g., images) from a particular area to a specific node with some time constraints. We assume that each room in the building is equipped with acoustic sensors or motion sensors and the goal is to collect information in areas where noise or motion is detected. The mobile robots have camera devices that can capture a fullsight (i.e., 360-degree view) snapshot of a target area. The raw image may be directly sent to other nodes if the network supports it, or it can be preprocessed, e.g., by applying some form of compression or abstraction, and feature extraction (possibly at a different more powerful node), and then communicated to other nodes.

For the specific example, we assume that the primary goal is injected into the network by the user at a fixed root node around which initially the robots are randomly clustered. Goals and facts are opportunistically shared whenever connectivity exists. Each robot can compute its local solution based on its local knowledge. The solution assigns an approximate target region as a subgoal to each robot, which then starts to move in order to locally realize the goal. The distributed reasoning continues so that the local solutions are continuously recomputed and movements are adjusted correspondingly. In addition to the randomness due to network and environment, randomization techniques, e.g., random selection of clauses and goals, and random waiting, are used to desynchronize the robots. The movements are constrained by the floor plan, which could be opportunistically updated and shared, but is simply assumed to be given as part of the logical theory, which is available at all nodes, in our simple example. The network connectivity model can also be exploited to suppress position changes that would lead to disconnections. Still, temporary disruptions are possible due to system perturbations, failures, and uncertainty caused by delayed/incomplete knowledge.

3.1 Simulation Setup

Figure 2 shows knowledge sharing between three cyber-nodes and their devices. Each node is equipped with a knowledge manager, i.e., an implementation of the distributed knowledge-sharing model, a reasoner, i.e., an implementation of our logical framework, and attached devices that can be regarded as subnodes exhibiting a declarative knowledge-based interface. The devices are using the distributed knowledge-sharing model but are implemented using conventional code (simulation code in our case), i.e., outside the logical framework. Figure 2 contains two mobile robots, each with a positioning device and a camera device. It also contains a fixed sensor node, which can have attached noise or motion sensors or both. In our simulation, we experimented with one to five robots and used a root node that is similar to a robot but assumed to be at a fixed location and serves as a user access point to the cyber-physical system.

By definition, cyber-facts/goals have a form $p_c(t, e_1, ...e_n)$. However, in our example we leave the creation time t implicit in the case of cyber-goals, because it is not used in the theory.

To illustrate a simple operation, assume now that a user injects a goal such as $TakeSnapshot(t_T, t_T + \Delta t_{sd}, a, I)$ with $t_T = 0.0$ and $\Delta t_{sd} = 20.0$ into the root node. t_T and Δt_{sd} indicate the earliest time and the deadline to take the snapshot. The reasoner cannot solve this goal by means of the logical theory, but the local camera device may find that it can handle $TakeSnapshot(0.0, 20.0, a, I)$ by taking an image of the area a in the time interval 0.0, ..., 20.0. As a result it generates a fact $Snapshot(10.0, a, i)$, indicating that image i was taken at time 10.0 in area a, which is added to the local knowledge base and can in turn lead to further reasoning in the logical theory.

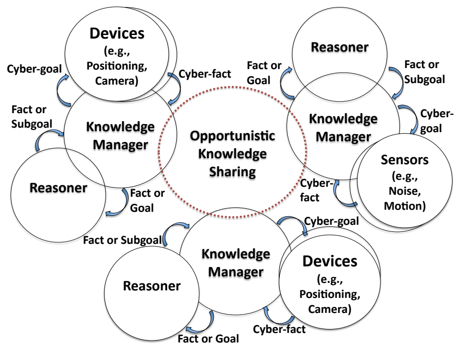

Fig. 2. Distributed Knowledge Sharing between two Robots and a Sensor in an Instrumented Cyber-Space

Knowledge (i.e., facts and goals) is opportunistically disseminated whenever connectivity exists among robots. In our experiments, we use an abstract topological *mobility model* instead of a model with actual coordinates. We assume that rooms in our scenario correspond to regions exhibiting similar connectivity, and the *network model* is defined so that links are up between robots when they reside in the same or adjacent rooms. For our experiments, we use a disruption-tolerant networking simulator [16] that abstracts from the underlying networking stack. It uses a simple graph-based dynamic network model, where each link has a state, e.g., up or down, and is characterized by its abstract features such as bandwidth, latency, and error rate. The floor plan restricts the mobility and will be reflected by a collection of adjacency facts as part of the theory.

3.2 Declarative Problem Formulation

Figure 3 shows the logical theory that is used to declaratively represent our sample application. The predicates are summarized in Table 1. Recall that a user wants the system to take an image whenever some trigger condition occurs, process it, and deliver it at the root node. The trigger conditions are specified as forward clauses $F1$ and $F2$, which can be applied at any time when the condi-

Forward Clauses:

$F1 : Noise(T, A) \Rightarrow Trigger(T, A).$
$F2 : Motion(T, A) \Rightarrow Trigger(T, A).$
$F3 : Adjacent(A, B) \Rightarrow Adjacent(B, A).$

Backward Clauses:

$B1 : Interest(T_I, I, R) \Leftarrow Result(T_I, T_T, 0, I), Deliver(T_I, T_T, 1, I, R).$
$B2 : Deliver(T_I, T_T, N_D, I, R) \Leftarrow Delivered(T_I, T_T, N_D, I, R).$
$B3 : Deliver(T_I, T_T, N_D, I, R) \Leftarrow$
$\quad Position(T_P, R, A), Position(T'_P, R', A'), R' \neq R,$
$\quad MoveTo(T_I, T_T, N_D, 0, \infty, R', A), Deliver(T_I, T_T, N_D, I, R).$
$B4 : Result(T_I, T_T, N_D, I') \Leftarrow CompImage(T_I, T_T, N_D, I), I' = Extract(I).$
$B5 : CompImage(T_I, T_T, N_D, I') \Leftarrow RawImage(T_I, T_T, N_D, I), I' = Compress(I).$
$B6 : RawImage(T_I, T_T, N_D, I) \Leftarrow Trigger(T_T, A), T_I \leq T_T,$
$\quad MoveTo(T_I, T_T, N_D, 0, T_T + \Delta t_{sd}, R, A),$
$\quad TakeSnapshot(T_I, T_T, N_D, T_T + \Delta t_{sd}, A, I).$
$B7 : TakeSnapshot(T_I, T_T, N_D, D, A, I) \Leftarrow$
$\quad Snapshot(T_I, T_T, N_D, T_S, A, I), T_T \leq T_S, T_S \leq D.$
$B8 : MoveTo(T_I, T_T, N_D, W', D, R, B) \Leftarrow Position(T_P, R, B), T_P \leq D.$
$B9 : MoveTo(T_I, T_T, N_D, W', D, R, B) \Leftarrow Adjacent(A, B), W' > -b_w, W = W' - 1,$
$\quad MoveTo(T_I, T_T, N_D, W, D, R, A), Move(T_I, T_T, N_D, W', D, R, A, B).$

Replacement Ordering: (f denotes a fact and g a goal and x denotes either)

$O1 : f : Position(t_P, r, \ldots) \prec f : Position(t'_P, r, \ldots)$ if $t_P < t'_P$.
$O2 : x : X(t_I, \ldots) \prec g : Interest(t'_I, \ldots)$ if $t_I < t'_I$.
$O3 : x : X(t_I, t_T, n_D, \ldots) \prec f : Result(t_I, t_T, n_D, \ldots)$ if $x : X \neq f : Result$.
$O4 : x : X(t_I, t_D, n_D, \ldots) \prec f : Deliver(t_I, t_D, n_D, \ldots)$ if $x : X \neq f : Deliver$.

Variables: T: time, D: snapshot deadline, A and B: area, R: robot,
I: image or derived information, N: identifier, W: weight
Constants: Δt_{sd}: relative snapshot deadline (max. delay from trigger event),
b_w: bound for weight (diameter of the floor plan)

Fig. 3. Logical Theory for Distributed Surveilance by a Team of Mobile Robots.

tions are met. Like all knowledge, facts are disseminated in the network, hence compensating for heterogeneity in the node capabilities and other limitations, e.g., due to resources and sensor failures.

Different from forward clauses, backward clauses are evaluated only when a user or the reasoner injects a goal (or a new subgoal) that unifies with the conclusion of the clause. For example, the backward clause $B1$ is triggered by an *Interest* goal and generates a corresponding fact when successfully applied. In the case of $B2$ and $B3$, the reasoner attempts to check if the required image is delivered to the root node. If a corresponding *Delivered* fact is available, then the *Deliver* goal is satisfied by applying $B2$. Otherwise, $B3$ needs to be used to check the current position of the root node (via the *Position* fact) and to guide the robot toward the area where the root node is positioned (via the *MoveTo* subgoal). $B4$-$B6$ specify how we construct an image I. When one of trigger conditions, $F1$ or $F2$, is met, a robot needs to be located at the specific area

Atom	Type	Realization	Meaning
$Adjacent(a, b)$	Fact	Theory	areas a and b are adjacent (represents floorplan)
$Noise(t, a)$	Cyber-Fact	Sensor	noise is detected in the area a at time t
$Motion(t, a)$	Cyber-Fact	Sensor	motion is detected in the area a at time t
$Trigger(t, a)$	Fact	Theory	triggering condition is met in the area a at time t
$Position(t, r, a)$	Cyber-Fact	Positioning	robot r is positioned in the area a at time t
$Interest(t_I, I, r)$	Cyber-Goal	Theory	user at root node r is interested in information I
$Result(t_I, t_T, n_D, I)$	Goal	Theory	an feature extraction I needs to be computed
$CompImage(t_I, t_T, n_D, I)$	Goal	Theory	an abstract image I needs to be computed
$RawImage(t_I, t_T, n_D, I)$	Goal	Theory	an image I needs to be generated
$MoveTo(t_I, t_T, n_D, w, t, R, b)$	Goal	Theory	robot R needs to move to the area b until time t
$Move(t_I, t_T, n_D, w, t, R, a, b)$	Cyber-Goal	Positioning	robot R needs to move from area a to area b until time t
$TakeSnapshot(t_I, t_T, n_D, t, a, I)$	Goal	Theory	a snapshot I needs to be taken in area a between time $t_T, ..., t$
$TakeSnapshot(t_I, t_T, n_D, t, a, I)$	Cyber-Goal	Camera	a snapshot I needs to be taken in area a between time $t_T, ..., t$
$Snapshot(t_I, t_T, n_D, t_s, a, i)$	Cyber-Fact	Camera	a snapshot i is taken in the area a at time t_s
$Deliver(t_I, t_T, n_D, i, r)$	Cyber-Goal	Root Node	request information i to be delivered to user at root node r
$Delivered(t_I, t_T, n_D, i, r)$	Cyber-Fact	Root Node	information i has been delivered to user at root node r

Table 1. Interpretation of Cyber-Predicates and Theory Predicates

(the $MoveTo$ predicate should be satisfied for this purpose) to take a snapshot. Only after these two subgoals, $Trigger$ and $MoveTo$ in $B6$, are satisfied, the $TakeSnapshot$ goal can be realized by the device, which generates $Snapshot$ as a new fact. Now, a fact $RawImage$ can be generated as specified by $B6$ and $B7$.

The clauses $B4$ and $B5$ show the processing of a captured image, namely, compression and feature extraction. For example, in the condition of $B5$ the $Compress$ function takes an image I and assigns the compressed image to I'. Raw images are implemented as built-in objects with attributes such as time and area information, and image processing functions such as *Compress* and *Extract* do not alter that information.

The *Adjacent* predicate is used to represent the topological floor plan, which in turn determines the network connectivity among robots. $Adjacent(a, b)$ means areas a and b are adjacent rooms. The forward clause $F3$ captures the assumption that adjacency, and hence connectivity, is symmetric. The clause $B9$ also uses the *Adjacent* predicate to plan the robot movement, since a *Move* goal can be realized by the robot's positioning device only if the room is adjacent.

The *MoveTo* goal will generate a new *Position* fact when it is realized. For example, $MoveTo(T_I, T_T, N_D, 0, T_T + \Delta t_{sd}, R, A)$ is used in the condition as a subgoal of $B6$, and $Position(0.0, r, a)$ is a fact that the local device of robot r provides as its initial position. The clause $B9$ is of particular interest, since it initiates the position change of a robot. As we see from $B8$, if the current position of a robot is equal to its destination, then the *MoveTo* predicate is already satisfied. Otherwise, a robot plans to move toward the destination as specified in $B9$. We use W and the bound b_w to limit the depth of the *MoveTo* subgoal generation. The value W is also used to represent weighted goals to enable locally weight-adaptive goal selection, a first step toward combining distributed reasoning and optimization. When a positioning device has a local choice among several possible *MoveTo* goals, it favors a *MoveTo* goal with a higher weight W since higher weight indicates a goal closer to the final destination, 0 being the maximum. Generally, other factors can influence the local goal selection of cyber-physical devices, e.g., in this case, since R is unbound, robots closer to an area with sufficient resources may be more likely to select a corresponding goal if it is easier for them to realize.

In addition to forward and backward clauses, the replacement ordering is specified in Figure 3. Under $O1$ we specify that an old *Position* fact of a robot is replaced by a new *Position* fact based on their timestamps. In $O2$, an old *Interest* goal becomes obsolete in view of a new *Interest* goal based on the time t_I associated with the *Interest* goal. Under $O3$ and $O4$, we specify orderings so that a fact can cancel goals and facts (except itself) that have led to the generation of such a fact. To avoid canceling goals and facts that are not related to a specific fact, we assume that t_I, t_T, n_D can serve as unique identifiers. Specifically, t_I and t_T are intended to distinguish different interest goals and trigger conditions, respectively. We define t_I and t_T as the time when an interest goal is injected ($B1$) and when a trigger condition occurs ($B6$), respectively. The number n_D, which can be either 0 or 1, distinguishes between *Result* and *Deliver* stages in $B1$, and their corresponding subgoals and related facts.

3.3 Sketch of a Distributed Execution

Figure 4 shows duality of two kinds of knowledge: facts and goals. Facts can be derived from observations or by applying forward clauses. Goals can be injected by a user or refined by applying backward clauses. Forward reasoning derives facts from known facts. Backward reasoning refines a goal to a number of subgoals. Facts and goals are both disseminated in the network as shown in Figure

2. A goal can be matched with a fact anywhere in the network as illustrated with dotted lines in Figure 4. For example, the goal $Trigger(T, A)$ can be matched with the fact $Trigger(0.0, a)$ as illustrated in Figure 5.

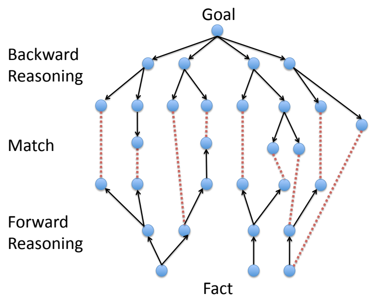

Fig. 4. Matching between Goals and Facts

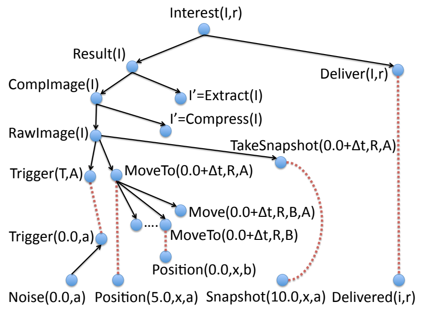

Fig. 5. Example of Distributed Execution

Figure 5 shows a more detailed view of a possible execution of the theory from Figure 3. For brevity, we have omitted some identifier arguments (e.g., T_I,T_T,N_D) in the figure and in the following description unless they are needed. At the top of Figure 5, the user injects a cyber-goal $Interest(I, r)$ at the root node r. At the bottom, a $Noise$ cyber-fact (representing an observation) leads to a fact $Trigger(0.0, a)$ by forward reasoning. In our framework, conditions are processed from left to right as we explained in Section 2. For example, the clause $B1$ in Figure 3 applied to the goal $Interest(I, r)$ attempts to solve $Result(I)$ before $Deliver(I, r)$, and $Result(I)$ fails at first, since a fact $Result(i)$ does not exist yet. The local reasoner feeds $Result(I)$ as a new subgoal into the local knowledge base. The knowledge base contains two goals at this point, $Interest(I, r)$ and $Result(I)$. Assume that $Result(I)$ is randomly selected from the local knowledge base. Subsequently, $CompImage(I)$ and then $RawImage(I)$ are fed into the knowledge base as new subgoals.

Clause $B6$ for $RawImage(I)$ has three subgoals involving $Trigger$, $MoveTo$, and $TakeSnapshot$. The leftmost subgoal can be finally matched with a fact $Trigger(0.0, a)$ that is derived from forward reasoning as depicted at the bottom of Figure 5. Next, the $MoveTo$ goal is further refined by applying $B8$ or $B9$. In case the root node r is positioned in the area a, it is able to detect its position and generate $Position(0.0, r, a)$ as a fact. Otherwise, backward clause $B9$ is used to plan for movement toward the desired area. The cyber-goal $Move(w, t + \Delta t, R, a, b)$ is realized by the local positioning device of a robot x when its current position is a and current time is before the deadline t. As a result, a new cyber-fact such as $Position(5.0, x, a)$ will be generated if the robot x indeed manages to move to a at the time 5.0 assuming a deadline $t + \Delta t = 20.0$. In a similar manner, the local camera device of robot x could take a snapshot of the area and have $Snapshot(10.0, a, i)$ generated to realize $TakeSnapshot(t, a, I)$, which eventually leads to the satisfaction of the $RawImage(I)$ goal.

The goals $CompImage(I)$ and $Result(I)$ can in turn be solved by the robot x, since it has a $RawImage(i)$ available as a fact in its local knowledge base. Alternatively, thanks to the fuly distributed nature of the reasoning process, *Compress* or *Extract* can be solved at other nodes (e.g., depending on resource availability) including the root node r, because $RawImage(i), CompImage(i)$, and $Result(i)$ are facts and disseminated through the network. The backward clause $B3$ is used to steer a robot toward the root node r unless backward clause $B2$ can be applied because the delivery has already been accomplished by means of other nodes or the robot can directly satisfy the delivery to the user (or to a higher-layer application). In the end, $Interest(I, r)$ is selected and satisfied by generating a fact $Delivered(i, r)$ at the root node r.

3.4 Discussion and Variations

The distributed nature of the framework improves the robustness of the system under intended or unintended perturbations. For instance, the system continues to operate correctly, i.e., within the scope of the logical theory, even after a human disables or replaces a robot or moves it to a different location. Complete failure of robots or some of their devices is covered as well. If the system is not already heterogeneous from the beginning, partial failures or resource limitations (e.g., battery charge of a robot) will eventually lead to a heterogeneous system which is why any assumption of homogeneity is avoided.

We have also experimented with several variations of the running example. For instance, the robot movement can be constrained by adding a condition $Movable(T, R, A)$ in $B9$ to check if a robot R can move away from an area A at time T. Now coverage, connectivity, or energy constraints can be formulated such as $Energy(T, R, E), E \geq e \Rightarrow Movable(T, R, A)$ to make sure that only robots with a sufficient amount of energy participate in position changes.

Our simple example assumes that the floor plan is given in advance. However, generalizing techniques such as SLAM (simultaneous localization and mapping) [3], which are well established in robotics, models could continuously adapt to the facts derived from observations, specifically the floor plan and the connectivity model. In the case of the floor plan, observations could be simply added as facts and SLAM could benefit from consequences generated based on a background theory. In the case of connectivity, the model parameters could be adapted to match the facts generated by neighbor discovery or even (distributed) signal strength measurements if available. SLAM could be generalized to a distributed mapping of signal strength or more generally the wireless spectrum for the purpose of network optimization [10]. An interesting direction would be to explore how such algorithms can be developed in a declarative framework.

4 Related Work

The idea of applying declarative techniques in communication and networking is not new. A large body of work exists in the areas of specification, analysis, and synthesis of networking policies and protocols, e.g., in the context of security, routing, or dynamic spectrum access.

Declarative querying of sensor networks has been studied through several approaches, for instance in [18], which composes services on the fly and in a goal-driven fashion using a concept of semantic streams. Declarative techniques to specify destinations have been used in disruption-tolerant networking [2]. A variant of Datalog has been applied to the declarative specification of peer-to-peer protocols in the P2 system [12]. Based on this work, [4] develops a very interesting approach to declarative sensor networks that can transmit generated facts to specific neighbors and can also utilize knowledge about neighbors to specify, e.g., routing algorithms. The promising idea of providing an abstraction that views a system as a single asset (an ensemble) rather then programming its individual components has been explored in several papers. Most interesting, the approach adapted in Meld [1] extends the ideas from declarative sensor networks to modular robots, i.e., ensembles of robots with inter-robot communication limited to immediate neighbors. As an example, the movement of a composite robot emerges as a result of the coordinated interaction between its homogeneous robot modules. An earlier functional approach to programming sensor networks is the Regiment macro-programming system [13], which uses a stream-based data-flow language. Most of the work focuses not on the theoretical foundations, but on the efficient compilation into a conventional programming language, which is one possible approach for practical deployment. Another approach, which we sometimes refer to as *embedded formal methods*, is the use of an effient reasoning engine in embedded systems such as software-defined radios [6] or routers [16] as explored in the context of disruption-tolerant networking.

In this paper, we have presented first steps toward combining forward and backward reasoning in a fully distributed fashion with knowledge that is transparently shared. A fixed or known neighborhood is not assumed in our more abstract approach, and the use and dissemination of both facts and goals aims at general cyber-physical systems with distributed actuation, and hence leads us beyond sensor networks, in particular to dynamic sensor/actuator networks that are, unlike ensembles, inherently heterogeneous.

5 Conclusions and Future Work

We have presented a distributed computational and logical foundation for declarative control of NCPS. Our approach is based on distributed knowledge sharing and supports a broad spectrum of operations between autonomy and cooperation with minimum assumptions on network connectivity. Specifically, we developed a distributed inference system based on Horn clause logic with built-ins and cyber-predicates that can capture the interaction with the physical world. In the underlying distributed computing model, facts and goals are represented as knowledge that can be shared opportunistically and guide the distributed reasoning process. The duality between facts and goals extends to the proof system, which treats forward and backward reasoning on an equal footing. Essential properties of our logical framework, such as soundness, completeness, and termination conditions, have been established under very general conditions, which will be the subject of a future paper.

A key feature of our distributed logical framework is its dynamic and interactive nature, meaning that facts represent observations, and goals can lead to changes in the environment that will manifest themselves as new facts flowing into the system. Whether an original local goal can be solved is often secondary, because the combined effect of a set of local goals on the cyber-physical system and its nondeterministic dynamics can lead to solutions of higher-level goals even without requiring solutions of all lower levels. To cope with system perturbations and unexpected failures, it is essential that local goals are not eliminated as soon as they are satisfied, because their main purpose, namely steering the system toward the satisfaction of a high-level property, may not have been achieved yet. Only after certain high-level goals are reached, auxiliary goals can be eliminated so that they do not have to comptete with new goals that steer the system toward the next high-level objective. Real-world systems are more complex, because such processes take place concurrently at multiple levels of abstraction and at different time scales. Our combination of a logic with an underlying partial order facilitates a new declarative specification style, which is particularly suitable for open distributed systems that can interact with a cooperative or uncooperative environment.

For experimentation, we have implemented a prototype of our logical framework that heavily relies on randomization techniques to achieve decoupling and implement locally fair nondeterminism that is used in our proof system to cover a broad range of possible implementations with different reasoning strategies. Our experiments with the simulation of an abstract networked multirobot system are a snapshot of ongoing work, but illustrate the application and the features of our framework in a simplified but nontrivial setting, and indicate possible directions for future work.

For realistic experiments, we plan to make use of existing robotic abstraction layers, such as the open-source Player/Stage/Gazebo Project [15]. Experiments in a real-world multirobot testbed similar to that of SRI's Centibots [14] and Commbots [8] projects are another possible direction. Beyond multirobot teams, we see opportunities to apply (extensions of) our framework to other unmanned autonomous vehicles, networks of pico-satellites [17], instrumented smart spaces [11], ad hoc social networking in a cyber-physical (instrumented) world, and next-generation adaptive networks and cognitive radios [9], where the devices exhibit a high degree of flexibility and the network can be morphed to adapt to user objectives and policies. Clearly, a lot of work remains ahead, because many of these applications require the combination of distributed reasoning with distributed optimization, and the use of weighted goals in our example is only a very first step in this direction.

References

1. Michael P. Ashley-Rollman, Seth Copen Goldstein, Peter Lee, Todd C. Mowry, and Padmanabhan Pillai. Meld: A declarative approach to programming ensembles. In *Proc. of the IEEE International Conference on Intelligent Robots and Systems (IROS '07)*, October 2007.
2. P. Basu, R. Krishnan, and D. W. Brown. Persistent delivery with deferred binding to descriptively named destinations. In *Proc. of IEEE Military Communications Conference*, 2008.

3. Howie Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization. *IEEE Trans. on Robotics and Automation*, 17(1):125–137, April 2001.
4. David Chu, Lucian Popa, Arsalan Tavakoli, Joseph M. Hellerstein, Philip Levis, Scott Shenker, and Ion Stoica. The design and implementation of a declarative sensor network system. In *SenSys '07: Proc. of the 5th International Conference on Embedded Networked Sensor Systems*, pages 175–188, 2007.
5. Falko Dressler. *Self-Organization in Sensor and Actor Networks.* Wiley, 2008.
6. Daniel Elenius, Grit Denker, and Mark-Oliver Stehr. A semantic web reasoner for rules, equations and constraints. In *RR '08: Proc. of the 2nd International Conference on Web Reasoning and Rule Systems*, pages 135–149, Berlin, Heidelberg, 2008. Springer-Verlag.
7. Stephen Farrell and V. Cahill. *Delay- and Disruption-Tolerant Networking.* Artech House, Inc., Norwood, MA, USA, 2006.
8. Brian P. Gerkey, Roger Mailler, and Benoit Morisset. Commbots: Distributed control of mobile communication relays. In *Proc. of the AAAI Workshop on Auction Mechanisms for Robot Coordination (AuctionBots)*, pages 51–57, 2006.
9. Gregory D. Troxel et al. Enabling open-source cognitively-controlled collaboration among software-defined radio nodes. *Comput. Netw.*, 52(4):898–911, 2008.
10. Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Nonlinear constraint network optimization for efficient map learning. *Trans. Intell. Transport. Sys.*, 10(3):428–439, 2009.
11. M. Kim, D. Massaguer, N. Dutt, S. Mehrotra, S. Ren, M.-O. Stehr, C. Talcott, and N. Venkatasubramanian. A semantic framework for reconfiguration of instrumented cyber physical spaces. In *Workshop on Event-based Semantics, CPS Week*, 2008.
12. Boon Thau Loo, Tyson Condie, Minos Garofalakis, David E. Gay, Joseph M. Hellerstein, Petros Maniatis, Raghu Ramakrishnan, Timothy Roscoe, and Ion Stoica. Declarative networking. *Commun. ACM*, 52(11):87–95, 2009.
13. Ryan Newton, Greg Morrisett, and Matt Welsh. The regiment macroprogramming system. In *IPSN '07: Proc. of the 6th International Conference on Information Processing in Sensor Networks*, pages 489–498, New York, NY, USA, 2007. ACM.
14. Charles L. Ortiz, Régis Vincent, and Benoit Morisset. Task inference and distributed task management in the Centibots robotic system. In *AAMAS '05: Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 860–867, 2005.
15. Radu Bogdan Rusu, Alexis Maldonado, Michael Beetz, and Brian Gerkey. Extending Player/Stage/Gazebo towards cognitive robots acting in ubiquitous sensor-equipped environments. In *ICRA '07: Proc. of the IEEE International Conference on Robotics and Automation Workshop for Network Robot Systems*, 2007.
16. Mark-Oliver Stehr and Carolyn Talcott. Planning and learning algorithms for routing in disruption-tolerant networks. In *Proc. of IEEE Military Communications Conference*, 2008.
17. S. Toorian, K. Diaz, and S. Lee. The CubeSet approach to space access. In *Aerospace Conference, IEEE*, 2008.
18. Kamin Whitehouse, Feng Zhao, and Jie Liu. Semantic streams: A framework for composable semantic interpretation of sensor data. In *Proc. of the European Workshop on Wireless Sensor Networks*, pages 5–20. EWSN, 2006.