

Reasoning about Uncertain Contexts in Pervasive Computing Environments

Context-aware systems can't always identify the current context precisely, so they need support for handling uncertainty. A prototype pervasive computing infrastructure, Gaia, allows applications and services to reason about uncertainty using mechanisms such as probabilistic logic, fuzzy logic, and Bayesian networks.

Mark Weiser envisioned computing environments that are pervaded with so many computing devices and sensors that they seem to disappear into the background, allowing humans to focus on daily tasks rather than on underlying technologies.¹ To enable this vision, we must transform today's "dumb," context-insensitive, and isolated machines into intelligent, programmable, and context-aware clusters of machinery. However, unobtrusiveness and context awareness involve capturing and making sense of imprecise and sometimes conflicting data and uncertain physical worlds.

Different types of entities (software objects) in the environment must be able to reason about uncertainty. These include entities that sense uncertain

contexts, entities that infer other uncertain contexts from these basic, sensed contexts, and applications that adapt how they behave on the basis of uncertain contexts. Having a common model of uncertainty that is used by all entities in the environment makes it easier for developers to build new services and applications in such environments and to reuse various ways of handling uncertainty.

We developed an uncertainty model based on a predicate representation of contexts and associated confidence values. This model forms the basis for reasoning about uncertainty using various mechanisms such as probabilistic logic, fuzzy logic,² and Bayesian networks. Each of these mechanisms is useful in handling uncertainty in different situations. We incorporated these mechanisms in Gaia,³ a distributed middleware system that enables Active Spaces—physical spaces enhanced with ubiquitous computing devices—to form an interactive, programmable computing and communication system. This article describes our model for uncertain contexts and how Gaia handles uncertainty and various reasoning mechanisms.

One of our goals was to make it easy for developers to integrate the use of uncertainty in their programs. Gaia's context infrastructure provides services and libraries that help entities acquire and reason about uncertain contextual information. We define the structure and properties of context predicates in ontologies—this makes it easier to develop programs that reason about context.

Model for dealing with uncertainty

Our model represents any piece of information whose truth value is potentially uncertain as a predicate.

Anand Ranganathan, Jalal Al-Muhtadi, and Roy H. Campbell
University of Illinois at Urbana-Champaign

The context predicate

We represent contexts as predicates, following the convention that the predicate's name is the type of context being described (such as location, temperature, or time).^{4,5} This gives us a simple, uniform representation for different kinds of contexts. Many predicates are defined to have arguments in a subject-object format (**ContextType**(**<Subject>**,**<Object>**) or in a subject-verb-object format (**ContextType**(**<Subject>**,**<Verb>**,**<Object>**). We can also use relational operators such as "=" and "<" as arguments. Example context predicates include

```
location (jeff, in, room 3105)
activity (room 3102, meeting)
light (room 3220, dim)
office (chetan, room 3216)
```

Some contexts (such as **office**) are certain, whereas others (such as **location** and **activity**) might be uncertain. The type of context constrains the values that the predicate arguments can take. For example, if the type of context is **location**, the first argument must be a person or object, the second argument must be a preposition or a verb such as **entering**, **leaving**, or **in**, and the third argument must be a location. We type-check the predicates whenever they are used in rules or other reasoning mechanisms.

Our context model gives a common representation for context that all entities in our environment use; the model itself doesn't describe what operations can be performed on contexts. Instead, it provides a common base on which various reasoning mechanisms can be specified to handle context. Using predicates to represent context information is helpful because we can plug them directly into rules and other reasoning and learning mechanisms for handling uncertainty. Moreover, it lets us describe context in a uniform way that is independent of programming language, operating system, or middleware.

Ontologies

We specify the predicates' structures and semantics in an ontology that defines various context types as well as the arguments that the predicates must have.⁶ Each context type corresponds to a class in the ontology. The ontology is written in DAML+OIL,⁷ which is fast becoming the de facto language of the Semantic Web.⁸

We use the ontology to check predicates' validity. It also makes including

Reasoning using probabilistic and fuzzy logic

Our use of probabilistic logic is based on earlier work.⁹ This logic lets us make statements such as "the probability of *E* is less than 1/3" and "the probability of *E* is at least twice the probability of *F*," where *E* and *F* are arbitrary events. The logic has a complete axiomatization, and the complexity of deciding satisfiability in it is no worse than that of propositional logic.

Using predicates lets us describe context in a uniform way that is independent of programming language, operating system, or middleware.

context predicates in rules easier, since we know the predicates' structure and the kinds of values different arguments can take. Using ontologies also lets different ubiquitous computing environments interoperate, since we can define translations between the terms used in these environments' ontologies. Ontological descriptions also help reduce the possibility of one kind of uncertainty: how different entities will interpret the same piece of context information.

Confidence values

We model uncertainty in our environment by attaching a confidence value between 0 and 1 to predicates. This value measures the probability (in the case of probabilistic approaches) or the membership value (in the case of fuzzy logic²) of the event corresponding to the context predicate being true. For example, **prob(location(carol, in, room 3233)) = 0.5** means that the probability that Carol is in Room 3233 is 0.5. Gaia uses these confidence values in various ways.

Probabilistic logic lets us write rules that reason about events' probabilities in terms of the probabilities of other related events. For writing rules in probabilistic logic and performing reasoning, we use XSB (<http://xsb.sourceforge.net>), a kind of Prolog that uses tabling and indexing to improve performance. Besides standard Prolog, XSB also allows programming in HiLog.¹⁰ Because HiLog has a higher-order syntax, it allows predicates to appear as arguments of other predicates. This allows unification on the predicate symbols themselves as well as on their arguments. However, the semantics of HiLog is first-order and has a sound and complete proof procedure. We need such higher-order logic syntax to write rules about the probabilities of context predicates. An example rule (written in XSB) is

```
prob(X, Y, union, P) :-
  prob(X, Q), prob(Y, R), disjoint(X, Y), (P is Q + R).
```

This rule essentially says that $\Pr(X \cup Y) = \Pr(X) + \Pr(Y)$ if *X* and *Y* are disjoint events—that

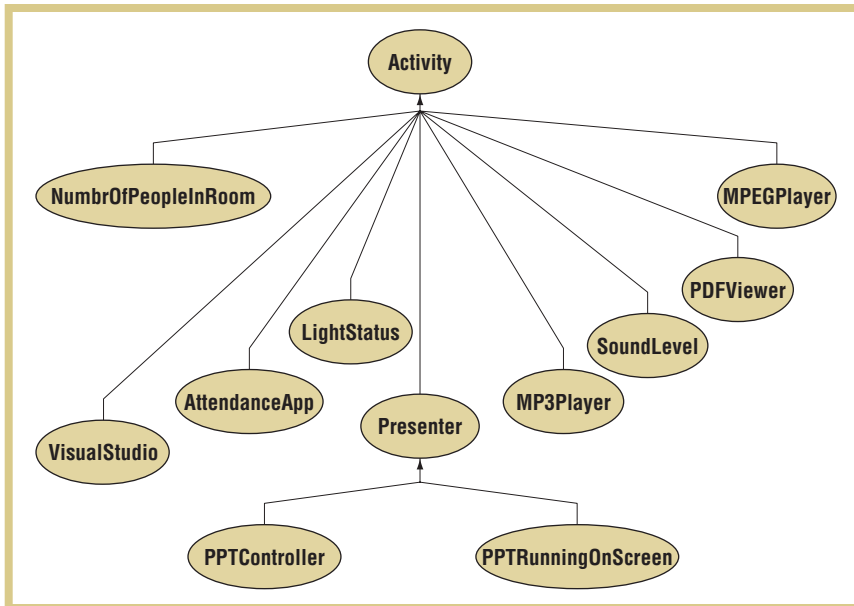


Figure 1. A Bayesian network for inferring activity.

is, they never occur together. X and Y can be context predicates. For example, X could be `location(Bob, in, Room 2401)` and Y could be `location(Bob, in, Room 3234)`. X and Y are disjoint events, since Bob can't be in two different locations at the same time.

Fuzzy logic is somewhat similar to probabilistic logic. In fuzzy logic, confidence values represent degrees of membership rather than probability. Fuzzy logic is useful in capturing and representing imprecise notions such as “tall,” “trustworthy,” and “confidence” and reasoning about them. We base this model on the fuzzy predicate logic developed by Petr Hajek¹¹ and adapt it to pervasive computing environments. We can thus write rules involving various context predicates and then reason on the basis of these rules.

Bayesian networks

Bayesian networks are *directed acyclic graphs*, where the nodes are random variables representing various events and the arcs between nodes represent causal relationships. In our model, each value that a variable can take corresponds to a certain context predicate. To specify a Bayesian network's probability distribution, you give the prior probabilities of all root nodes (nodes with no predecessors) and the conditional probabilities of

all nonroot nodes (given all possible combinations of their direct predecessors).

Figure 1 shows an example of a Bayesian network that infers the activity taking place in a room. Take the Activity node at the root of the network, for instance. The random variable *activity* can take values such as “meeting,” “presentation,” “idle,” and so on, with different probabilities. Assuming that we've trained the network for Room 2401, these values correspond to the context predicates `activity(room 2401, meeting)`, `activity(room 2401, presentation)`, `activity(room 2401, idle)`, and so on. These context predicates have different probability values based on what the network deduces.

Bayesian networks are a powerful way of handling uncertainty, especially when there are causal relationships between various events. They are useful for performing probabilistic sensor fusion and higher-level context derivation. In our environment, root nodes in the Bayesian network represent the information to be deduced, while the leaves are sensed information. The intermediate nodes are important subgoals that are helpful to the deduction process.

Context infrastructure

In Gaia, our prototype pervasive computing infrastructure, different entities

are aware of their context and adapt their behavior to it.⁴ The infrastructure provides support for gathering context information from sensors and delivering appropriate context information to entities. It also lets entities infer higher-level contexts from low-level sensed contexts. Different kinds of entities are involved in Gaia's context infrastructure (see Figure 2). All parts of the infrastructure have support for handling uncertain contexts:

- *Context Providers* are sensors or other data sources of context information. They allow other agents (or Context Consumers) to query them for context information. Some Context Providers also have an event channel in which they regularly send context events. Thus, other agents can either query a Provider or listen on the event channel to get context information. Context Providers can associate a probability measure with the context information they provide, if the information is uncertain. They get information on the semantics of the context they provide (that is, whether it's certain or not) and the context predicates' structure from the ontologies.
- *Context Synthesizers* get sensed contexts from various Context Providers, deduce higher-level or abstract contexts from these simple sensed contexts, and then provide these deduced contexts to other agents. For example, we have a Context Synthesizer that infers the activity occurring in our smart room based on information such as the number of people in the room and the applications that are running. Context Synthesizers also associate a probability measure with the context information they provide, if the information is uncertain.
- *Context Consumers* are entities (context-aware applications) that get different types of contexts from Context

Providers or Context Synthesizers. They reason about the current context and adapt the way they behave according to it. They can use various mechanisms for reasoning about context, such as rules written in different types of logic and learning mechanisms such as Bayesian learning. These reasoning mechanisms can take into account any uncertainties in context information.

- *Context Provider Lookup Service* enables Context Providers to advertise what they offer and agents to find appropriate Context Providers.
- *Context History Service* lets agents query for past contexts, which are logged in a database.
- *Ontology Server* maintains the ontologies that describe different types of contextual information.

Each Active Space has one Context Provider Lookup Service, one Context History Service, and one Ontology Server.

Structure of a reasoning entity

Figure 3 illustrates the structure of an entity (application, device, or service) that uses reasoning to deal with uncertain contexts. Each entity has a knowledge base associated with it, in which relevant facts and rules are asserted. The KB contains facts about the current context in the form of context predicates as well as information from other services, such as authentication and access control services. When a Context Provider or other service detects a change in the current context, it sends an event on its event channel. The KB listens to various event channels and updates itself when a change happens.

The common model for context helps frame uniform, easy-to-use APIs for the interactions between entities. Context

Providers, the KB, the Reasoning Engine, and the application/service use the same predicate representation of context. Applications and services reason about facts in the KB using a particular reasoning mechanism (for example, Bayesian inference or fuzzy logic). They query the reasoning engine through a fixed API to get information they need to make decisions or to get inferred facts. The use of a fixed interface between the application or service and the reasoning engine lets us plug in different reasoning engines for reasoning about uncertain contexts. Thus, adding new reasoning engines (say, based on neural networks) in the future will be easy.

One of the goals of our infrastructure was to let developers concentrate on developing rules or networks for reasoning and not be burdened with the low-level details involved in getting entities to operate in the environment. Developers who want to use reasoning in their entities specify the kinds of facts the KB must contain (that is, which context predicates are required by the reasoning engine to perform the

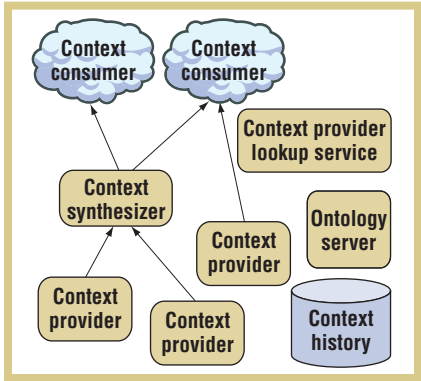


Figure 2. Gaia context infrastructure.

inferencing). The Context Infrastructure automatically takes care of updating the KB with the current context information by making it listen to the appropriate event channels. The developer also specifies the reasoning mechanism in the form of rules or a Bayesian network.

Support for probabilistic and fuzzy logic

Some entities in Gaia use probabilistic or fuzzy logic in the form of rules written in Prolog or HiLog. The infrastructure provides easy mechanisms for developers to specify inference rules for these entities. Our Ontology Explorer browser lets developers find definitions

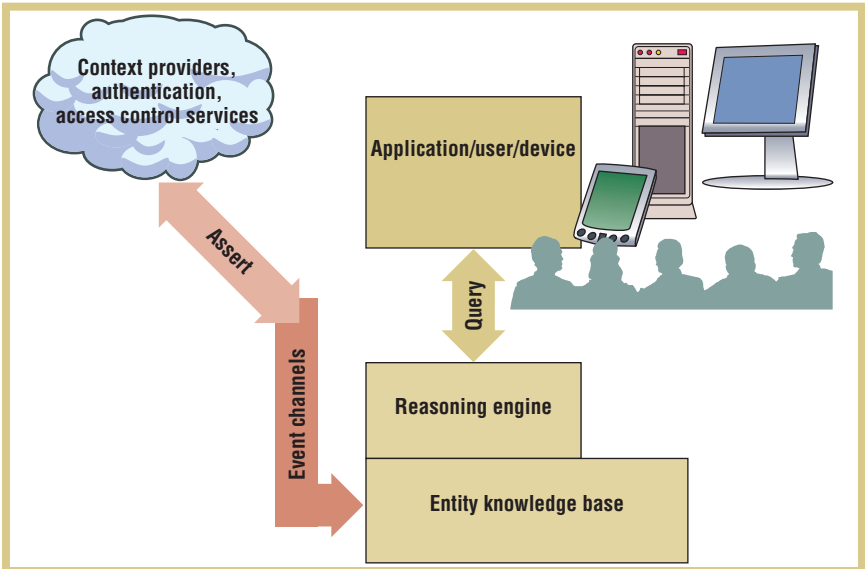


Figure 3. Structure of an entity that uses reasoning.

of various terms and contextual information as well as relationships between terms; they can also browse the ontologies to get the terminology used in the environment. We have also developed a GUI that lets developers construct rules in probabilistic and fuzzy logic. This GUI lets developers construct context predicates (based on their definitions in the ontologies) and combine them in different rules.

Support for Bayesian networks

Using Microsoft's Belief Network (MSBN) software, Gaia developers can create Bayesian networks representing causal relationships between events¹² and then enter prior probability distri-

butions on the nodes. The developer associates each network node with a specific context predicate. Constructing a Bayesian network involves identifying causal dependencies between different events, which might not be an easy task. In our environment, developers browse the ontology of context information to know what contexts are available and how they might relate to one another.

the network's root nodes. MSBN uses clique-tree propagation methods to calculate probabilities exactly. This inference method, however, could take exponential time and wouldn't work for very large networks. So far, though, the kinds of problems we've used Bayesian inferencing for involve fairly small networks (less than 50 nodes), and we haven't encountered any performance problems. However, for larger networks, we would have to employ other approximate inference strategies.

Examples

Gaia handles uncertainty in three broad areas: sensing context information, inferring context information, and

tainty of whether the badge is actually in the room. Because a badge detector's range can fall outside the room it's monitoring, the detector might sense people as they move outside (for example, through a nearby corridor). Thus, the probability that the badge is actually in the room is the area of the room within the badge detector's circle of detection divided by the area of the circle of detection. If there are other badge detectors nearby, then one or more of them might sense the person's badge, and we can calculate the probability that the badge is actually in the room using a sensor-fusion procedure. Similar probability measures can be worked out if 802.11-based access points are used to sense people's locations.

Another factor that causes uncertainty in using badges for location tracking is the fact that the person associated with the badge might not be carrying the badge. The person might have left the badge on a desk, or someone might have stolen it. Quantifying this uncertainty and associating it with a probability value is more difficult. Such behavior varies from person to person and is difficult to generalize to all people.

To partially address this problem, in our system, we assign a probability value to the event of a person actually having the badge in hand. This probability value is not entirely ad hoc; it's an intuitive sense of how reliable the location-sensing device is. For example, the value of this probability is less for a badge than it is for a smart ring because, intuitively, a ring is less likely to be misplaced or stolen. However, such a solution isn't entirely satisfactory, and we are looking at other ways of dealing with such immeasurable factors.

More generally, any location Context Provider (associated with a badge, ring, fingerprint recognizer, or similar device) uses probabilistic logic to evaluate the

The kinds of problems we've used Bayesian inferencing for involve fairly small networks, and we haven't encountered any performance problems.

using uncertain context information by applications. (See the "Related Work" sidebar for information on others' research on uncertainty.)

Two examples of Context Providers that provide uncertain contexts are *RFID badges* for location detection and various authentication devices.

Uncertainty in sensing context

RFID badges. Many location-sensing systems cannot resolve an object's location accurately, but only do so within some tolerance. In our environment, we detect people's locations using RF-based badges. Badge detectors are placed in different parts of the building.

We use badges to try to determine if a certain person is in a certain room. Two factors contribute to uncertainty in a person's location. One factor is the uncer-

using uncertain context information by applications. (See the "Related Work" sidebar for information on others' research on uncertainty.)

Two examples of Context Providers that provide uncertain contexts are *RFID badges* for location detection and various authentication devices.

RFID badges. Many location-sensing systems cannot resolve an object's location accurately, but only do so within some tolerance. In our environment, we detect people's locations using RF-based badges. Badge detectors are placed in different parts of the building.

We use badges to try to determine if a certain person is in a certain room. Two factors contribute to uncertainty in a person's location. One factor is the uncer-

Related Work

There has been work in addressing the problem of uncertainty in context information. However, none of the existing work addresses the problem in a general way. Anind Dey and colleagues suggest that ambiguous information can be resolved by a mediation process involving the user.¹ However, considering the potentially large quantities of context information involved in pervasive computing environments and the rapid rate at which context can change, this approach places an unreasonable burden on the user. Hui Lei and colleagues describe a context service that allows context information to be associated with quality metrics, such as freshness and confidence,² but their model of context lacks formality. Paul Castro and colleagues use Bayesian networks for sensor fusion,³ but their work considers only location information. Albrecht Schmidt and colleagues associate each of their context values with a certainty measure that captures the likelihood that the value accurately reflects reality.⁴ Philip Gray and Daniel Salber include information quality as a type of meta-information in their context model, and describe six quality attributes: coverage, resolution, accuracy, repeatability, frequency, and timeliness.⁵ However, they all do not specify ways of handling the uncertainty or reasoning about it. The model described by Karen Henriksen and colleagues⁶ supports quality by allowing associations between objects to be annotated with a number of quality

parameters, which capture the dimensions of quality considered relevant to that association. However, the model doesn't give any way of using these quality metrics in real-life situations.

REFERENCES

1. A. Dey, J. Manko., and G. Abowd, *Distributed Mediation of Imperfectly Sensed Context in Aware Environments*, tech. report GIT-GVU-00-14, Georgia Inst. of Technology, 2000.
2. H. Lei et al., "The Design and Applications of a Context Service," *ACM Mobile Computing and Comm. Rev.*, vol. 6, no. 4, Oct. 2002.
3. P. Castro et al., "A Probabilistic Room Location Service for Wireless Networked Environments," *UbiComp 2001 Conf.*, LNCS 2201, Springer-Verlag, 2001, pp. 18–34.
4. A. Schmidt et al., "Advanced Interaction in Context," *Proc. 1st Int'l Symp. Handheld and Ubiquitous Computing (HUC 99)*, LNCS 1707, Springer-Verlag, 1999, pp. 89–101.
5. P. Gray and D. Salber, "Modeling and Using Sensed Context in the Design of Interactive Applications," *8th IFIP Conf. Eng. Human-Computer Interaction*, LNCS 2254, Springer-Verlag, 2001, pp. 317–336.
6. K. Henriksen, J. Indulska, and A. Rakotonirainy, "Modeling Context Information in Pervasive Computing Systems," *Pervasive 2002*, LNCS 2414, Springer-Verlag, 2002, pp. 167–180.

probability of a person being in a certain room. The main inference rule is

$$\begin{aligned} \text{Prob}(\text{location}(X, \text{in}, Y)) = \\ \text{Prob}(\text{device associated with person is in } Y) \\ \times \text{Prob}(\text{device is actually with person } X) \end{aligned}$$

There are other issues that come into play as well in determining whether the badge is actually in the room, such as people's behavior and the typical amount of time people spend in the room. Our current algorithm serves only as a first level of approximation in determining location on the basis of probabilistic logic; more complex algorithms could consider other factors. For example, inferring that a user has forgotten his badge is possible using other information such as his log-in into a machine that is situated away from his badge. Also, the time elapsed since the last detection of a badge is a useful indicator

of the certainty that the user (and his badge) is still in the room. We plan on adding rules that incorporate these conditions as well.

Authentication devices. Various biometric authentication devices such as fingerprint readers, face and voice recognition software, and so on give a measure of confidence while recognizing users. They also use probabilistic logic to evaluate the degree of confidence in authentication. These devices typically use pattern-matching algorithms (such as neural networks) to compare the sample fingerprint or face with a database of stored fingerprints or faces. The extent of match gives a measure of the device's confidence that the person has been correctly identified. Like active badges, authentication devices also have rules to evaluate the probability that people are indeed who they claim to be. Authentication

devices then associate this probability with the authentication context predicate they provide.

Inferred context

Gaia has various components that try to infer higher-level context from basic, sensed contexts. If the basic, sensed contexts are themselves uncertain, then the inferred contexts will also be uncertain. We need to somehow combine the probability or confidence measures from different sensors to get a measure of uncertainty for the inferred combined context. We can use probabilistic logic or Bayesian approaches to do this.

Authentication. Users have a variety of devices in pervasive computing environments to identify themselves to the system, including wearable devices, voice and face recognition, badges containing identification information, fingerprint

identification, and retinal scans. The authentication process should enable users to authenticate themselves to the system using various means. At the same time, authentication information from different devices is associated with different levels of confidence, because some authentication mechanisms (for example, biometrics features and DNA samples) are much stronger than others (such as passwords and badges). Also, depending on the context, the level of confidence associated with some mechanisms might change. For example, if a system suspects that someone's password has been compromised, it might reduce the level of confidence associated with passwords.

Intuitively, it's also reasonable to assume that the more devices the user employs to authenticate himself, the

```
confidenceLevel(mechanism1, 0.6)
confidenceLevel(mechanism2, 0.8)
possessRole(bob, grad student)
possessRole(bob, teaching assistant)
```

The authentication service uses a simple rule to combine confidence values. Let's assume a principal tries to authenticate herself using n different authentication methods, and all these attempts were successful. If the confidence values associated with these authentication methods are V_1, V_2, \dots, V_n , then the net confidence V_{net} associated with this authentication is given by the formula (derived from probability theory)

$$V_{\text{net}} = 1 - (1 - V_1)(1 - V_2) \dots (1 - V_n).$$

Access control policies use the confidence values of authentication and the

assigns probabilities to the different possible activities in the room.

We trained the network on data we collected from the smart room. The training process helped us assign probability distributions to the different nodes. We can now use the network to deduce activity in the room on the basis of sensed contexts. At any point of time, the context infrastructure obtains the state of various leaf nodes. The Bayesian inferencing algorithm then calculates the conditional probability distribution of the Activity node given the states of the leaf nodes.

We evaluated the accuracy of our Bayesian network in deducing the correct activity in the room. In our test run, we took 210 observations (one every half hour, 30 per day for a week). In each observation, the network assigned probabilities to different activities (such as a meeting or a presentation) on the basis of the room's context (such as the light status or presence of people). We found that the network assigned the highest probability to the actual activity in the room in nearly 84 percent of the observations. One possible reason why the network performed well is that we used the smart room in only a limited number of ways that followed fairly learnable and distinct patterns, so the network could easily learn the activity patterns.

Applications' use of uncertain context information

Applications in Gaia use the probabilities of various contexts to influence their decisions and behaviors. The applications get context predicates and their associated probabilities from Context Providers and Context Synthesizers, and they reason about the uncertain context information using fuzzy logic or Bayesian inferencing mechanisms. If they don't want to deal directly with the probability distributions of different context predicates, they can assume that the

Deck, some filler text some filler
insert a three to five line, filler text filler
more filler text has been inserted here

greater the system's confidence in that user's identity. The authentication service performs a kind of sensor fusion. It collects authentication information from the different devices and determines the net confidence of authentication in case the user uses more than one device to authenticate himself. This net confidence, expressed as a confidence value, is then used in various access control decisions. Once authentication is complete, the authentication service asserts new information about the authenticated principals and the confidence values associated with them, as well as additional information.¹³ For example,

```
authenticatedUsing(bob, mechanism1)
authenticatedUsing(bob, mechanism2)
```

role names associated with a principal, as we illustrate later.

Room activity example. Figure 1 shows a subset of the Bayesian network we use to infer activity in our smart room. The network consists of various nodes representing basic context information (whether a PDFViewer is running, the light and sound levels, the number of people). Some nodes represent the activity of a single user (such as the Presenter node, which indicates if a person is giving a presentation in the room). Finally, an Activity node describes the activity in the room as a "Presentation," "Demo," "Meeting," "Seminar," "Idle," and so on. On the basis of evidence collected from various Context Providers, the network

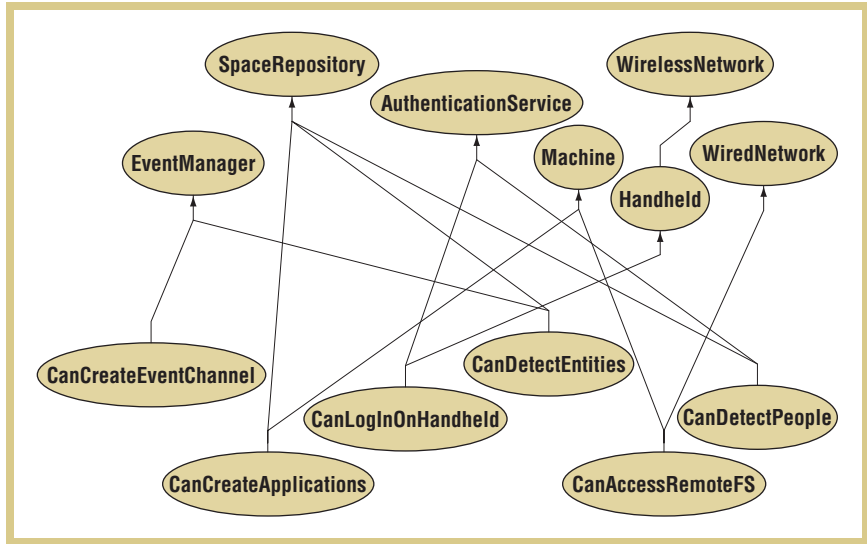
Figure 4. Bayesian network for troubleshooting in Gaia.

predicate with the maximum probability is true and the rest are false.

Access control. Access control decisions in Gaia use fuzzy reasoning. Access control policies for various applications and services are based on linear inequalities on the confidence values of context information. Gaia grants access to a resource to a particular principal if the principal has been authenticated to a satisfactory level and if the context has been sensed with a certain degree of accuracy. For example, the following is an access control rule written in Prolog:

```
canAccess(P, display) :-
  confidenceLevel(authenticated(P), C), C > 0.7,
  Prob(activity(2401,cs 101 presentation), Y),
  Y > 0.8,
  possessRole(P, presenter)
```

This example states that a principal P may access the display if P has been authenticated with a confidence value of at least 0.7, the activity in the room is a presentation with probability of at least 0.8, and P is the presenter for this presentation activity. The `authenticated` predicate is asserted on the basis of information from the Authentication Service, and the `activity` predicate is obtained from the Activity Context Provider (which itself uses a Bayesian network to get the probability associated with an activity). The minimum probabilities are set on the basis of how critical the resource is. The Access Control service sets the role information on the basis of what role the user has chosen or what role the user's schedule or the system administrator have assigned to the user. Using a logic-based language gives us a powerful, expressive, yet simplistic language for writing security policies. Developers use our infrastructure to write access control rules for their applications and services and then to deploy them in the environment.



Troubleshooting. We also use a Bayesian network for troubleshooting the environment. Given the presence of various symptoms, the network infers the probabilities of various Gaia services, applications, devices, or other resources having failed. For example, if the smart room doesn't detect people's presence in the room, this implies that the Space Repository (which maintains a database describing the room's current state) or the Authentication Service could have failed with certain conditional probabilities.

Figure 4 shows a subset of this network. The network consists of nodes such as EventManager and AuthenticationService that represent the state of different Gaia entities. Other nodes represent certain kinds of failures (or symptoms of failures). For example, the CanCreateEventChannel node represents whether or not creating an event channel is possible. The network helps to determine which entities in Gaia might have failed given certain observed symptoms. We trained the network over a period of time using data logged from the environment. We now use it to determine which services or applications to troubleshoot or restart on the basis of observing certain symptoms. Again, we built and deployed this network using the mechanisms our infrastructure provides for creating and reasoning with

Bayesian networks. We are looking at enhancing the network by using temporal information.

To cope with uncertainty, our environment uses a combination of learning based on Bayesian networks and explicit rules written in probabilistic logic. Both methods are useful in different scenarios.

Bayesian networks are useful for learning the probability distributions of events and enable reasoning about causal relationships between observations and the system state. They, however, must be trained before they can be used (we trained our activity and troubleshooting networks over the period of a week), but because they are flexible and can be retrained easily, they can adapt to changing circumstances.

Probabilistic logic is useful when we have precise knowledge of events' probabilities; fuzzy logic is useful when we want to represent imprecise notions. Both probabilistic and fuzzy logic are useful in scenarios where getting data to train a Bayesian network is difficult. This is especially true in the area of security. In our prototypical pervasive computing environment, getting training data for authentication and access control operations was difficult, mainly

because getting data implied attacking the system in some manner and measuring how often such attacks were successful. Our context model provides a common base for the different reasoning mechanisms, besides allowing new reasoning mechanisms to be easily incorporated in the future. Our infrastructure also provides support for developers to incorporate the use of reasoning in the services and applications they develop. In particular, many devel-

opers have used Gaia's GUI to develop and deploy access control rules based on fuzzy logic for their entities. The support our infrastructure provides for building, training, and deploying Bayesian networks has also allowed us to rapidly incorporate Bayesian reasoning in many of our entities.

Reasoning about uncertainty has helped make components in our pervasive computing environment more robust and more capable of adapting to changing dynamics. We envision several enhancements to our infrastructure for coping with and reasoning about uncertainty. We are trying to see if incorporating reasoning about uncertainty could make other applications more robust. An important aspect of uncertainty is the freshness of context information. We hope to quantify freshness and develop axioms or rules to deal with it.

Another enhancement is the use of plausibility measures¹⁴ in situations where we can't associate a specific number like a probability but where we could say that one event is more or less likely than another. While a probability measure maps an event to a number between 0 and 1, a plausibility measure just associates an event with an element in a partially ordered set. This would enable us to express the fact that a fingerprint sensor might more accurately determine a person's location than a badge detector could. ■

the AUTHORS



Anand Ranganathan is a doctoral candidate and research assistant working on the Gaia project in the University of Illinois at Urbana-Champaign's Department of Computer Science. His research interests include context-aware computing, mobile computing, the Semantic Web, and automated reasoning and learning. He received his B.Tech in computer science from the Indian Institute of Technology in Madras. Contact him at 1330 SC, 201 N. Goodwin, Urbana, IL 61801; ranganat@uiuc.edu.



Jalal Al-Muhtadi is a doctoral candidate and research assistant working on the Gaia project at the University of Illinois at Urbana-Champaign, where he researches middleware and infrastructure security and privacy issues. He received his MS in computer science from the University of Illinois at Urbana-Champaign. Contact him at 1330 SC, 201 N. Goodwin, Urbana, IL 61801; almuhtad@cs.uiuc.edu.



Roy H. Campbell is a professor of computer science at the University of Illinois at Urbana-Champaign. His research interests include operating systems, distributed multimedia, network security, and ubiquitous computing. He received his PhD in computing from the University of Newcastle upon Tyne. Contact him at 3122 SC, 201 N. Goodwin, Urbana, IL 61801; rhc@uiuc.edu.

Computing, vol. 1, no. 4, 2002, pp. 74–83.

4. A. Ranganathan and R.H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," *ACM/IFIP/USENIX Int'l Middleware Conf.*, LNCS 2672, Springer-Verlag, 2003; <http://choices.cs.uiuc.edu/~ranganat/Pubs/MiddlewareForContext-FinalVersion.pdf>.
5. A. Ranganathan et al., "ConChat: A Context-Aware Chat Program," *IEEE Pervasive Computing*, vol. 1, no. 3, July–Sept. 2002, pp. 51–57.
6. A. Dey, J. Manko., and G. Abowd, *Distributed Mediation of Imperfectly Sensed Context in Aware Environments*, tech. report GIT-GVU-00-14, Georgia Inst. of Technology, 2000.
7. I. Horrocks, "DAML+OIL: A Description Logic for the Semantic Web," *IEEE Bull. Tech. Committee on Data Eng.*, 2002, vol. 25, no. 1, Mar. 2002, pp. 4–9.
8. P. Castro et al., "A Probabilistic Room Location Service for Wireless Networked Environments," *UbiComp 2001 Conf.*, LNCS 2201, Springer-Verlag, 2001, pp. 18–34.
9. J. Halpern, R. Fagin, and N. Megiddo, "A Logic for Reasoning about Probabilities," *Information and Computation*, vol. 87, nos. 1–2, 1990, pp. 78–128.
10. W. Chen, M. Kifer, and D.S. Warren, "HILOG: A Foundation for Higher-Order Logic Programming," *J. Logic Programming*, vol. 15, no. 3, 1993, pp. 187–230.
11. P. Hajek, "Fuzzy Predicate Calculus and Fuzzy Rules," *Fuzzy If-Then Rules in Computational Intelligence*, D.A. Ruan and E.E. Kerre, eds., Kluwer Academic Publishers, 2000, pp. 27–35.
12. C.M. Kadie, D. Hovel, and E. Horvitz, *MSBNx: A Component-Centric Toolkit for Modeling and Inference with Bayesian Networks*, tech. report MSR-TR-2001-67, Microsoft Research, 2001.
13. J. Al-Muhtadi et al., "Cerberus: A Context-Aware Security Scheme for Smart Spaces," *IEEE Int'l Conf. Pervasive Computing and Communications (PerCom 2003)*, IEEE CS Press, 2003.
14. J. Halpern and N. Friedman, "Plausibility Measures: A User's Guide," *Proc. 11th Conf. Uncertainty in AI*, Morgan Kaufmann, 1995, pp. 175–184.

REFERENCES

1. M. Weiser, "The Computer for the Twenty-First Century," *Scientific Am.*, vol. 265, no. 3, Sept. 1991, pp. 94–104.
2. L. Zadeh, "Fuzzy Sets as Basis for a Theory of Possibility," *Fuzzy Sets and Systems*, vol. 1, 1978, pp. 3–28.
3. M. Roman et al., "A Middleware Infrastructure for Active Spaces," *IEEE Pervasive*