Context Reasoning 2

Melika Abolhassani
March 26, 2013
Context Reasoning and Ontology

- What is Reasoning?

- "Machine Understanding"

- Find facts that are implicit given explicitly stated facts

- Find what you know, but you don't know you know it – yet.

- Example
  - A is father of B, B is father of C, then A is ancestor of C.
  - D is mother of B, then D is female

- Context Reasoning: Finding higher level context which is more meaningful from the low level context.

- Ontology: The shared understanding and relation of some domains. Often conceived as a set of entities, relations, functions, axioms and instances
Ontology Based Context Modeling and Reasoning using OWL

Authors: Xiao Hang Wang, Da Qing Zhang, Tao Gu, Hung Keng Pung
Institute for Infocom Research, Singapore

- Previously various informal context data models existed:
  - Context Toolkit: Attribute-value Tuples
  - CoolTown: Web based data model each object has a corresponding Web description
  - Gaia: First-order predicates written in DAML+OIL

Problem with current formal context models: no formal knowledge sharing or quantitative evaluation of feasibility of context reasoning in pervasive computing environments
Goal of the Paper

An ontology-based formal context model is introduced and quantitatively evaluated.

The model addresses critical issues like:

- Formal context representation
- Knowledge sharing
- Logic based context reasoning

The quantitative evaluation is for context reasoning in a pervasive computing environment.
CONON: The CONtext ONtology

- It is for modeling context in pervasive computing environments

- The objective is a model which includes modeling a set of upper-level entities and provides flexible extensibility to add specific details

- Application and services grouped as a collection of sub-domains for different environments (home, office, vehicle)

- The main concepts of the context are common but different in details.

- Context model has 2 parts: upper ontology + specific ontology
Upper and Specific Ontology

Upper Ontology

- Context in each domain shares common concepts
- Encourages the reuse of general concepts
- Provides flexible interface for defining application-specific knowledge

Specific ontology

- collection of ontology set
- defines the details of general concepts and their features in each sub-domain
Figure 1. Partial Definition of CONON upper ontology
Specific Ontology for home domain

Figure 2. Partial definition of a specific ontology for home domain
Context Reasoning

Example: when the user is sleeping in the bedroom or taking a shower in the bathroom, incoming calls are forwarded to voice mail box or when the user is watching TV the volume of ringing is higher, when he is having dinner the phone is silenced.

The use of context reasoning has two folds

- Checking the consistency of context
- Deducing high-level implicit context from low-level explicit context

Two categories of context reasoning

- Ontology reasoning
- User-defined reasoning
### Ontology Reasoning

<table>
<thead>
<tr>
<th></th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>subClassOf</strong></td>
<td>(?a rdfs:subClassOf ?b) ∧ (?b rdfs:subClassOf ?c) → (?a rdfs:subClassOf ?c)</td>
</tr>
<tr>
<td><strong>subProperty-Of</strong></td>
<td>(?a rdfs:subPropertyOf ?b) ∧ (?b rdfs:subPropertyOf ?c) → (?a rdfs:subPropertyOf ?c)</td>
</tr>
</tbody>
</table>
### Example of Ontology Reasoning

#### Table 2. Reasoning about location using ontology

<table>
<thead>
<tr>
<th>DL Reasoning Rules</th>
<th>Implicit Context</th>
<th>Explicit Context</th>
</tr>
</thead>
</table>
  <locatedIn rdf:resource="#Home"/>  
</Person>  
  <Building rdf:ID="Home">  
  <contains rdf:resource="#Bedroom"/>  
  <contains rdf:resource="#Wang"/>  
</Building>  
  <Room rdf:ID="Bedroom">  
  <contains rdf:resource="#Wang"/>  
</Room> | <owl:ObjectProperty rdf:ID="locatedIn">  
  <rdf:type owl:TransitiveProperty"/>  
  <owl:inverseOf rdf:resource="#contains"/>  
</owl:ObjectProperty>  
  <Person rdf:ID="Wang">  
  <locatedIn rdf:resource="#Bedroom"/>  
</Person>  
  <Room rdf:ID="Bedroom">  
  <locatedIn rdf:resource="#Home"/>  
</Room> |
## User-defined Context Reasoning

<table>
<thead>
<tr>
<th>Situation</th>
<th>Reasoning Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleeping</td>
<td>(?u locatedIn Bedroom) $\land$ (Bedroom lightLevel LOW) $\land$ (Bedroom drapeStatus CLOSED) $\Rightarrow$ (?u situation SLEEPING)</td>
</tr>
<tr>
<td>Showering</td>
<td>(?u locatedIn Bathroom) $\land$ (WaterHeater locatedIn Bathroom) $\land$ (Bathroom doorStatus CLOSED) $\land$ (WaterHeater status ON) $\Rightarrow$ (?u situation SHOWERING)</td>
</tr>
<tr>
<td>Cooking</td>
<td>(?u locatedIn Kitchen) $\land$ (ElectricOven locatedIn Kitchen) $\land$ (ElectricOven status ON) $\Rightarrow$ (?u situation COOKING)</td>
</tr>
<tr>
<td>Watching-TV</td>
<td>(?u locatedIn LivingRoom) $\land$ (TVSet locatedIn LivingRoom) $\land$ (TVSet status ON) $\Rightarrow$ (?u situation WATCHINGTV)</td>
</tr>
<tr>
<td>Having-Dinner</td>
<td>(?u locatedIn DiningRoom) $\land$ (?v locatedIn DiningRoom) $\land$ (?u owl:differentFrom ?v) $\Rightarrow$ (?u situation HAVINGDINNER)</td>
</tr>
</tbody>
</table>
Conclusion

Prototype implementation and Experiment shows that CONON is Feasible for non-time-critical applications

Question: what we need to care for time-critical applications such as navigating systems?

Three major factors

- Size of context information
- Complexity of reasoning rules
- CPU speed

We need to control the scale of context dataset and the complexity of rule set

Perform static, complex reasoning tasks (e.g., description logic reasoning for checking inconsistency) in an off-line manner.

De-coupling context processing and context usage is needed in order to achieve satisfactory performance
Introduction:

- Context-awareness involves dealing with imprecise and conflicting data and uncertainty.
- Different types of entities (software objects) in the environment must be able to reason about uncertainty.
Goal of the paper

A prototype pervasive computing infrastructure, Gaia, allows applications to reason about uncertainty.

This paper shows:

- How Gaia (middleware) handles uncertainty and various reasoning mechanisms
- It makes it easy for developers to integrate the use of uncertainty in their programs
Any piece of information whose truth value is potentially uncertain is modeled as a predicate.

Contexts are represented as predicates. Predicate's name is type of context (location- temperature -
time)
- `ContextType(<Subject>, <Verb>, <Object>)`
- `ContextType(<Subject>, <Object>)`

example:
- location(jeff, in, room 3105)
- activity(room 3102, meeting)
- light(room 3220, dim)
- office(chetan, room 3216)
Model for dealing with uncertainty

We type-check the predicates whenever they are used in rules or other reasoning mechanisms.

Benefit of this model:
- Predicates plug directly into rules and learning mechanisms to handle uncertainty.
- Uniform way that is independent of programming language, operating system or middleware
- The context model has a common representation for context for all devices in environment.
- The model gives a common base for mechanisms to handle context
Model for dealing with uncertainty

Ontologies:
- Written in DAML+OIL which is becoming the language of the Semantic Web
- We use the ontology to check predicates’ validity (both structure and argument)
- Defines various context types as well as the arguments that predicates must have

Confidence values
- a value between 0 and 1
- probability for probabilistic approaches
- membership value if fuzzy logic
- These are the probabilities of the events
- For example, \( \text{prob(location(carol, in, room 3233))} = 0.5 \)
Reasoning using probabilistic and fuzzy logic

**Probabilistic logic:**
- We can say probability of E is less than F (E and F are events)
- Writing rules in probabilistic logic we can use XSB (kind of a prolog which allows programming in Hilog)

Example of rule in XSB:

\[
\text{prob}(X, Y, \text{union}, P) :\text{- prob}(X, Q), \text{prob}(Y, R), \text{disjoint}(X, Y), (P \text{ is } Q + R).
\]

X could be `location(Bob, in, Room 2401)` and Y could be `location(Bob, in, Room 3234)`

Meaning of the rule: \( \Pr(X \cup Y) = \Pr(X) + \Pr(Y) \) if X and Y are disjoint events

**Fuzzy logic:**
- Variables have a truth value (ranging from 0 to 1) rather than 0 or 1 as in binary logic
- Used confidence values which represent degrees of membership rather than probability
- Fuzzy logic captures imprecise notions such as adjectives (For example how cold it is)
Reasoning Using Bayesian Networks

Bayesian networks are directed acyclic graphs

- nodes are random variables representing various events
- arcs are causal relationships
- Values are certain context predicates

Example:
- activity(room 2401, meeting),
- activity(room 2401, presentation),
An example of Bayesian Network

<table>
<thead>
<tr>
<th>A</th>
<th>P(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>0.6</td>
</tr>
<tr>
<td>true</td>
<td>0.4</td>
</tr>
</tbody>
</table>

| A | B | P(B|A) |
|---|---|-------|
| false | false | 0.01 |
| false | true  | 0.99 |
| true  | false | 0.7  |
| true  | true  | 0.3  |

| B | C | P(C|B) |
|---|---|-------|
| false | false | 0.4 |
| false | true  | 0.6 |
| true  | false | 0.9 |
| true  | true  | 0.1 |

| B | D | P(D|B) |
|---|---|-------|
| false | false | 0.02 |
| false | true  | 0.98 |
| true  | false | 0.05 |
| true  | true  | 0.95 |
Infrastructure components

- **Context Providers**
  - Sensors or other data sources of context information
  - Being allowed to query for context information by other agents
  - Providing context information with probability measure if information is uncertain

- **Context Synthesizers**
  - Getting sensed contexts from various context providers
  - Deducing higher-level or abstract contexts from simple sensed contexts
  - Providing deduced contexts to other agents
  - Providing context information with probability measure if information is uncertain

- **Context Consumers**
  - Entities (context-aware applications)
  - Getting different types of contexts from Context Providers or Context Synthesizers
  - Reasoning about the current context and behaving according to it

- **Context Provider Lookup Service**
  - Enables Context Providers to advertise what they offer and agents to find appropriate Context Provider

- **Context History Service**
  - Lets agents query for past contexts, which are logged in a database.
How Gaia handles uncertainty?

Gaia handles uncertainty in three broad areas:

• Sensing context information
• Inferring context information
• Enabling applications to use uncertain context information.
Example: first uncertainty in Gaia: Sensing context

*Radio Frequency Identification badges* for location detection (RFID badges)

There are badge detectors in rooms

Why uncertainties?

- Is the badge actually in the room? (probability area of room / area of detector)

![Diagram](image)

- Is the person carrying the badge?

![Diagram](image)

Can become innovative:

- login of the person to another badge for example can help
- the time elapsed since the last detection
Example: first uncertainty in Gaia: Sensing context

- Authentication devices: fingerprint readers, face and voice recognition software, and so on give a measure of confidence while recognizing users.
- Uses probabilistic logic to evaluate the degree of confidence in authentication.
- Typically use pattern-matching algorithms to compare the sample fingerprint or face with a database of stored ones.
- The extent of the match shows how good it is.
Example: second uncertainty in Gaia: Inferred context

Gaia has components that try to infer higher-level context from basic.

We need to somehow combine the probability or confidence measures from different sensors to get a measure of uncertainty for the inferred (probabilistic logic or Bayesian)

1- Authentication: wearable devices—passwords, voice and face recognition—fingerprints, ...

Authentication information from different devices is associated with different levels of confidence: DNA vs passwords

The more devices to use for authentication the better

Probabilistic logic

\[ V_{\text{net}} = 1 - (1 - V_1)(1 - V_2) \ldots (1 - V_n). \]
Example: second uncertainty in Gaia: Inferred context

Room activity example: Bayesian Network


How to assign probability distributions to the nodes? training methods

The authors evaluated the accuracy of the model in this example
210 observations and highest probability to actual activity in 84% of cases
Example: 3rd uncertainty in Gaia: Applications’ use of uncertain context information

- Access Control: Fuzzy reasoning: can assume that the predicates with maximum value is true and everything else is false

Access control policies based on linear inequalities on the confidence values of context information.

Gaia grants access if the principal has been authenticated to a satisfactory level and if the context has been sensed with a certain degree of accuracy.

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 $0.8$
- $P$ is the presenter for this presentation activity.

$$\text{canAccess}(P, \text{display}) : -$$
$$\text{confidenceLevel}($$
$$\text{authenticated}(P), C), C > 0.7,$$
$$\text{Prob}($$
$$\text{activity}(2401, \text{cs 101 presentation}), Y),$$
$$Y > 0.8,$$
$$\text{possessRole}(P, \text{presenter})$$
Conclusion

- Bayesian networks useful for learning the probability distributions of events and enable reasoning about causal relationships. They, however, must be trained.
- Probabilistic logic when we have precise knowledge of events’ probabilities;
- Fuzzy logic is useful when we want to represent imprecise notions.

- This infrastructure provides support for developers to incorporate the use of reasoning in the services and applications
- Many developers have used Gaia’s rule-creation GUI to develop and deploy access control rules based on fuzzy logic
- Reasoning about uncertainty has lead to capability of adapting to changing dynamics

- Possible enhancements: 1- Plausibility measure: e.g. fingerprint sensor is better than badge detector
- 2-freshness of context information: how to deal with that?
A Survey of Semantics-Based Approaches for Context Reasoning in Ambient Intelligence

- A key issue in the study of Ambient Intelligence is context reasoning.
- The main challenges of context reasoning derive from the imperfect context, information, and the dynamic and heterogeneous nature of the ambient environments.

Goal of the paper:

- introduce semantics-based approaches for reasoning about context.
- how each approach addresses the requirements of ambient environments, identify their limitations, and propose possible future research directions.
Introduction

Knowledge Management in Ambient Intelligence should enable:

- (a) Reasoning with the highly dynamic and ambiguous context data;
- (b) Managing the potentially huge piece of context data, in a real-time fashion, considering the restricted computational capabilities of some mobile devices;
- (c) Collective intelligence, by supporting information sharing, and distributed reasoning between the entities of the ambient environment.

Outline:

- Ontological Reasoning
- Rule based Reasoning
- Distributed Reasoning
- Case base Reasoning
Ontological Reasoning

Advantages:

- Integrate well with the ontology model,
- Description Logic
- Low computational complexity → deal well with rapid changing context

Disadvantages:

- Limited reasoning capabilities
- Cannot deal with missing or ambiguous information, which is a common case in ambient environments,
- Not able to provide support for decision making.
- Therefore they cannot serve as a standalone solution for the needs of ambient context-aware applications.
Description Logic

- Concepts (unary predicates/formulae with one free variable) E.g., Person, Father, Mother
- Roles (binary predicates/formulae with two free variables) E.g., hasChild, hasHusband
- Individual names (constants) E.g., Alice, Bob, Cindy
- Subsumption (relations between concepts) E.g. Female $\subseteq$ Person
- Operators (for forming concepts and roles) And($\Pi$), Or($U$), Not ($\neg$)
  
  Universal qualifier ($\forall$), Existent qualifier ($\exists$) Number restriction $\leq, \geq, = \ldots$

A Description Logic (DL) models concepts, roles and individuals, and their relationships.

Example: hasParent(Bob, Alice) $\rightarrow$ hasChild(Alice, Bob)

hasBrother(Bob, David), hasBrother(David, Mack) $\rightarrow$ hasBrother(Bob, Mack)
Rule Based Reasoning

- Examples: SOCAM and Gaia (partially)
- Uses first order logic, logic programming, defeasible logic (richer reasoning models by more expressive rule languages)
- Rule languages provide a formal model for context reasoning.
- They are easy to understand and widely used.
- There are many systems that integrate them with the ontology model.
- Common deficiency: They cannot handle the highly changeable, ambiguous and imperfect context information.
- They have to build additional reasoning mechanisms to deal with conflicts, uncertainty and ambiguities.
Distributed Reasoning

- There might coexist many different entities that collect, process, and change the context information.
- They face context from different viewpoints based on their perceptive capabilities, their experiences and their goals.
- They may have different reasoning, storage and computing capabilities; they may ”speak” different languages; they may even have different levels of sociality.
Case Base Reasoning

- Example: AmbieSense
- on-line part that resides on the user’s mobile device
- off-line part that resides on the user’s backbone system
- When new information arrives the agent tries to retrieve a known context or case and classify the current situation based on the retrieved one
- The case is stored in the case base
- The device is filled so some of the reasoning is done by backbone servers.
Thank You!