Context Matching for Ambient Intelligence Applications

Andrei Olaru

University Politehnica of Bucharest

26.09.2013
Context Matching for Ambient Intelligence Applications

overview
**Ambient Intelligence** – or AmI – is a ubiquitous electronic environment that supports people in their daily tasks, in a proactive, but ”invisible” and non-intrusive manner. [Ducatel et al., 2001]

- We can view an AmI environment as a system of “information conveyers” [Weiser, 1993]
- **Software agents** are an appropriate implementation for AmI systems [Ramos et al., 2008]

The ideal features of AmI are also its greatest challenges:

- Uniformity / unification
- Scalability
- Availability / reliability

**Our approach:** Build a multi-agent system for the context-aware exchange of information in an AmI environment – the **AmlciTy initiative**. [Olaru et al., 2013]
Aml Layers (based on [El Fallah Seghrouchni, 2008])
**Context** is any information that can be used to characterize the situation of entities (i.e. a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. [Dey, 2001]

- Example of context-aware scenario:

  *If I am passing near my bank, during working hours, but I am not currently walking together with someone, I want to be reminded to go to the bank.*
Context is any information that can be used to characterize the situation of entities (i.e. a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. [Dey, 2001]

- Example of context-aware scenario:

*If I am passing near my bank location, during working hours time, but I am not currently walking together with someone social, I want to be reminded to go to the bank.*
Context is any information that can be used to characterize the situation of entities (i.e. a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. [Dey, 2001]

· Example of context-aware scenario:

or
Context is any information that can be used to characterize the situation of entities (i.e. a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. [Dey, 2001]

· Example of context-aware scenario:

Having received an email, I want the Aml system to detect if it is a call for papers and to notify me if I haven’t sent a paper.
Context is any information that can be used to characterize the situation of entities (i.e. a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. [Dey, 2001]

- Example of context-aware scenario:

  Having received an email, I want the Aml system to detect if it is a call for papers association and to notify me if I haven’t sent a paper association.
We define **context matching** as matching **context patterns** against the current **context graph**. The context graph represents relations between concepts; context patterns are graphs featuring generic nodes.

[Olaru et al., 2011]
· Context matching is used for
  ▶ knowledge integration → incoming information matches the agent’s patterns;
  ▶ situation recognition → the pattern matches part of the context graph;
  ▶ problem detection → only part of the pattern matches the context;
  ▶ sharing information → other agent’s patterns match the context graph.

· Local matching helps scalability and privacy-awareness.

← perceiving
← reactivity
← pro-activity / anticipation
← cooperation
← reasoning and detection are performed locally.

Local matching helps scalability and privacy-awareness.
Problem statement: devise an algorithm that makes context matching (underpinned by graph matching) a valid approach for the implementation of a context-awareness mechanism in agents that reside on devices of various sizes.

- that is, an algorithm that is tractable for cases specific to our problem:
  - graphs have mostly labeled edges;
  - there may be a reasonable amount of generic nodes in graph patterns;
  - the size of the context graph and context patterns will be adequate to the capabilities of the device;
Existing graph matching algorithms date from the 70’s to present times [Cordella et al., 2004]

- **exact vs. inexact matching**;
- **traditional algorithms match unlabeled, undirected graphs → modifications are needed**;
- **studied algorithms**:
  - McGregor – exploring the entire state space; [McGregor, 1982]
  - Koch – searching maximal cliques in the modular product of the edges; [Koch, 2001]
  - Larossa – modeling the matching problem as CSP. [Larrosa and Valiente, 2002]
- **an adaptation of various algorithms has been implemented and comparison has been performed**. [Dobrescu and Olaru, 2013]
Context and Patterns  

Context Matching

**Example**

**Model**

**Context graph**

\[ CG_A = (V, E) \]

\[ V \subseteq Concepts \]

\[ E = \{ \text{edge(from, to, value)} \mid \text{from, to} \in V, \text{value} \in \text{Relations} \} \]

**Context pattern**

\[ G^P_s = (V^P_s, E^P_s) \]

\[ V^P_s \subseteq Concepts \cup \{?\} \]

\[ E^P_s = \{ \text{edge(from, to, value)} \mid \text{from, to} \in V^P_s, \text{value} \in \text{Relations} \cup \{\lambda\} \} \]
Context Matching for Ambient Intelligence Applications

Introduction  Related Work  Formal Model  Algorithm  Evaluation  Visualization  Conclusion

Context and Patterns

Context Matching

Example

Model

Match: \( M_{A-si}(G'_A, G^P_m, G^P_x, f_v, k) \)

\[ G'_A \subseteq CG_A, G^P_m = (V^P_m, E^P_m) \subseteq G^P_s \] – matched subgraph, pattern solved part

\[ G^P_x = (V^P_x, E^P_x) \subseteq G^P_s \] – pattern unsolved part

\[ G^P_m \cup G^P_x = G^P_s, V^P_m \cap V^P_x = E^P_m \cap E^P_x = \emptyset \] – no solved & unsolved intersection

\[ f_v : V^P_s \rightarrow V' \] – vertex correspondence (bijective) – with:

- \( \forall v^P \in V^P_m, v^P = ? \) or \( v^P = f(v^P) \)

- \( \forall \text{edge}(v^P_i, v^P_j, \text{value}) \in E^P_m, \text{edge}(f(v^P_i), f(v^P_j), \text{value}) \in E' \)
- Start from all valid matches of one edge in the pattern with one edge in the graph;
- For each initial match, detect which other matches are valid merger candidates;
- Iterate over matches and create new matches, by merging them to their merger candidates;
· a match is represented as $M(G', G_m^P, f_v, f_e, fr, MC, MO, k)$

$G' \subseteq CG_A$ – matched subgraph of $CG_A$

$G_m^P \subseteq G_s^P$ – matched part of the pattern

$f_v : V_m^P \rightarrow V'$ – node function (bijective)

$f_e : E_m^P \rightarrow E'$ – edge function (bijective)

$fr \subseteq V_m^P$ – frontier

$MC$ – merger candidates

$MO$ – ’outer’ merger candidates

$k = |E_s^P| - |E_m^P|$ – missing edges
**Context Matching for Ambient Intelligence Applications**

**Introduction**

**Related Work**

**Formal Model**

**Algorithm**

**Evaluation**

**Visualization**

**Conclusion**

---

**Principle**

**Description (1)**

**Complexity Analysis**

**Matching Algorithm**

\[ \text{Match}(G, G^P) \]
**Context Matching for Ambient Intelligence Applications**

<table>
<thead>
<tr>
<th>Introduction</th>
<th>Related Work</th>
<th>Formal Model</th>
<th>Algorithm</th>
<th>Evaluation</th>
<th>Visualization</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principle</td>
<td><strong>Description (1)</strong></td>
<td>Complexity Analysis</td>
<td>Matching Algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Match**\((G, G^P)\)

**AddInitialMatches**

1.

2.

\[ \text{AddInitialMatches} \]

\[ \text{Match}(G, G^P) \]
\[ \text{Match}(G, G^P) \]

**AddInitialMatches**

- Immediate candidates
Context Matching for Ambient Intelligence Applications

**Introduction**

**Related Work**

**Formal Model**

**Algorithm**

**Evaluation**

**Visualization**

**Conclusion**

**Principle**

**Description (1)**

**Complexity Analysis**

**Matching Algorithm**

\[ \text{Match}(G, G^P) \]

**AddInitialMatches**

- Immediate candidates
- Outer candidates
\textbf{Context Matching for Ambient Intelligence Applications}

<table>
<thead>
<tr>
<th>Introduction</th>
<th>Related Work</th>
<th>Formal Model</th>
<th>Algorithm</th>
<th>Evaluation</th>
<th>Visualization</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principle</td>
<td>Description (1)</td>
<td>Complexity Analysis</td>
<td>Matching Algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{Match}(G, G^P) \]

\textbf{AddInitialMatches}

- Immediate candidates
- Outer candidates

for each $M', M''$
**Match**($G, G^P$)

**AddInitialMatches**

Immediate candidates

Outer candidates

for each $M', M''$

**Merge**($M', M''$)
Context Matching for Ambient Intelligence Applications

Introduction Related Work Formal Model | Algorithm Evaluation Visualization Conclusion

Principle Description (2) Complexity Analysis | Matching Algorithm

**AddInitialMatches**

for each \((e^P_{kp}, e^P_{kg})\) \(\in E^P \times E\)
if \(e^P_{kp}\) and \(e^P_{kg}\) match
create new initial single-edge match \(M\)
with \(E_m^P = \{e^P_{kp}\}\) and \(E' = \{e^P_{kg}\}\)
search \(MatchQueue\)
for all match candidates for \(M\)
add \(M\) to \(MatchQueue\)

**Match** \((G, G^P)\)

for each \(M' \in MatchQueue\), for each \(M'' \in M'.MC\)
remove \(M''\) from \(M'.MC\) and \(M'\) from \(M''.MC\)

\[
V = V' \cup V''; E = E' \cup E'' \\
V_m^P = V_m^P' \cup V_m^P''; E_m^P = E_m^P' \cup E_m^P'' \\
f_v = f_v' \cup f_v''; f_e = f_e' \cup f_e'' \\
fr = \{v^P \in fr' \cup fr'' \mid \exists e^P \text{ adj } v^P, e^P \notin E_m^P\} \\
MC = (MC' \cap MC'') \cup (MC' \cap MO'') \cup (MC'' \cap MO') \\
MO = MO' \cap MO''
\]

**GrowMatches**

**Merge** \((M', M'')\)

\[
MC = (MC' \cap MC'') \cup (MC' \cap MO'') \cup (MC'' \cap MO') \\
MO = MO' \cap MO''
\]
While the classic problem of matching undirected, unlabeled graphs is NP-complete, for the problem at hand the algorithm behaves significantly better.

\textit{AddInitialMatches} Creates a maximum of $m \times m^P$ matches, with many less if edges are labeled.

In our example ($m = 11, m^P = 8$) there are 19 initial matches.

Each initial match is tested against the other matches for compatibility.

Complexity: $O(m \times m^P) + O(initialMatches^2)$

\textit{Merge}(M’, M'’)

Adds all edges and nodes to the new match

Merges immediate and outer merger candidates

Complexity: $O(|E^P_M| + |E^P_M'|)$

\textit{GrowMatches}

Iterates over the match queue and merges matches with their candidates.

Complexity:

$O(initialMatches \cdot \log(initialMatches) \cdot \text{average } |E^P_m|)$
### Algorithm Evaluation

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Akkoyunlu</th>
<th>Bron-Kerbosch</th>
<th>Balas-Yu</th>
<th>Durand-Pasari</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expanded edges:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small example</td>
<td>124</td>
<td>120</td>
<td>135</td>
<td>119</td>
<td>34</td>
</tr>
<tr>
<td>Initial example</td>
<td>5431</td>
<td>5440</td>
<td>6423</td>
<td>5219</td>
<td>2459</td>
</tr>
<tr>
<td>No labeled edges</td>
<td>7054</td>
<td>9454</td>
<td>15843</td>
<td>9060</td>
<td>7581</td>
</tr>
<tr>
<td>No labels</td>
<td>326044</td>
<td>371943</td>
<td>578401</td>
<td>367725</td>
<td>108902</td>
</tr>
<tr>
<td>Large example</td>
<td>20470</td>
<td>19989</td>
<td>22170</td>
<td>18322</td>
<td>11834</td>
</tr>
</tbody>
</table>

Comments on algorithms not in the table:
- McGregor expands less edges, but many more nodes;
- Larossa expands significantly less edges, but can only provide full pattern matches.
Text-based representation for directed graphs that is easy to read by humans.

- it relies on building a tree of paths, starting with the longest path.

merging match [-] (k=6): AICnf (→CFP) →300311 : ?#3 (→article→?#2) -CFP→?#4
and match [3:2] (k=7): AICnf→conftime : ?#3→deadline→?#2
new match: match [-] (k=5): AICnf (→CFP) (→300311) →conftime : ?#5 (→article→?#4) (→CFP→?#6) -deadline→?#2

Console (ASCII)
Graphical representation for directed graphs

- relies on the textual representation to build paths with a low number of links between paths
- lays nodes out on concentric $120^\circ$ arcs

manual

automatic layout
We have developed an efficient algorithm for the partial matching of context patterns against context graphs.

It relies on creating all valid single-edge matches and then growing matches by merging.

The algorithm has been implemented and it has been compared with other, traditional, graph matching algorithms.

Future work:

- Further comparison with other algorithms using automatic graph generation and testing tools.
- Integration context matching as reasoning and decision engine in an agent-based platform for Ambient Intelligence.
Thank You!

Any Questions?
The enumeration of maximal cliques of large graphs.

Finding a maximum clique in an arbitrary graph.

Algorithm 457: finding all cliques of an undirected graph.
Communications of the ACM, 16(9):575–577.

A (sub) graph isomorphism algorithm for matching large graphs.

Understanding and using context.

Graph matching for context recognition.

Scenarios for ambient intelligence in 2010.

An efficient algorithm for similarity analysis of molecules.
Intelligence ambiante, les defis scientifiques.
presentation, Colloque Intelligence Ambiante, Forum Atena.

Enumerating all connected maximal common subgraphs in two graphs.

Constraint satisfaction algorithms for graph pattern matching.
Mathematical structures in computer science, 12(4):403–422.

Backtrack search algorithms and the maximal common subgraph problem.

Graphs and patterns for context-awareness.

A context-aware multi-agent system as a middleware for ambient intelligence.

Ambient intelligence - the next step for artificial intelligence.
IEEE Intelligent Systems, 23(2):15–18.
Thank You!

Any Questions?