Context Matching for Ambient Intelligence Applications

Andrei Olaru

University Politehnica of Bucharest

26.09.2013

Context Matching for Ambient Intelligence Applications

overview

Introduction

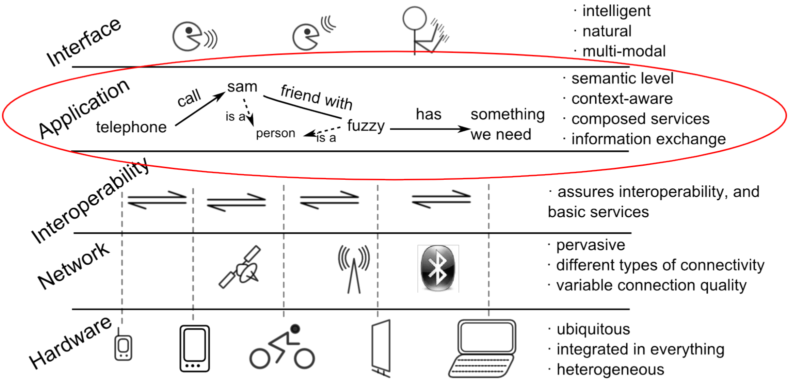
Ambient Intelligence – or AmI – is a ubiquitous electronic environment that supports people in their daily tasks, in a proactive, but "invisible" and non-intrusive manner. [Ducatel et al., 2001]

- 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 AmIciTy initiative. [Olaru et al., 2013]



AmI Layers (based on [El Fallah Seghrouchni, 2008])

Introduction

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.

Introduction

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

Introduction

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 AmI system to detect if it is a call for papers and to notify me if I haven't sent a paper .

Introduction

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]

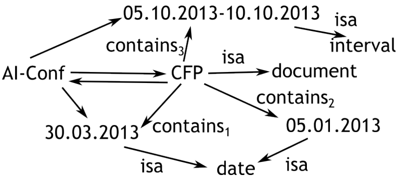
· Example of context-aware scenario:

Having received an email, I want the AmI system to detect if $\underbrace{\textit{it is a call for papers}}_{\textit{association}}$ *and to notify me* $\underbrace{\textit{if I haven't sent a paper}}_{\textit{association}}$.

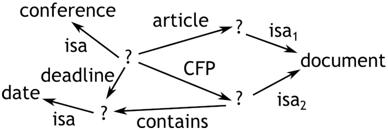
Introduction

We define context matching as matching *context patterns* against the current *context graph*.
[Olaru et al., 2011]

← · the context graph represents relations between concepts;
· context patterns are graphs featuring generic nodes;



context graph



context pattern

Introduction

· Context matching is used for

- knowledge integration → incoming information matches the agent's patterns; ← perceiving
- situation recognition → the pattern matches part of the context graph; ← reactivity
- problem detection → only part of the pattern matches the context; ← pro-activity / anticipation
- sharing information → other agent's patterns match the context graph. ← cooperation

· Local matching helps scalability and privacy-awareness. ← reasoning and detection are performed locally.

Introduction

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 a various sizes.

· that is, an algorithm that is tractable for cases specific to our problem:

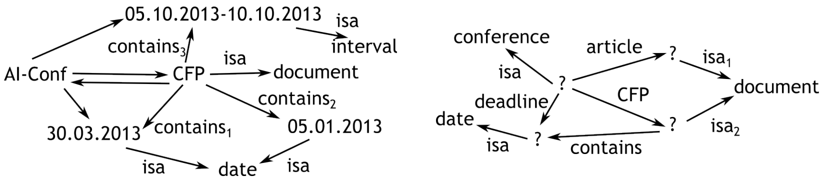
- graphs have mostly labeled edges;
- there may be a reasonable amount of generic nodes in graph patterns;
- the size of the context graph and context patterns will be adequate to the capabilities of the device;

Related Work

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]
  - Bron-Kerbosch, Durand-Pasari, Akkoyunlu and Balas-Yu – searching maximal cliques in the associations graph; [Bron and Kerbosch, 1973, Akkoyunlu, 1973, Balas and Yu, 1986, Durand et al., 1999]
  - 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]

Formal Model



Example

Model

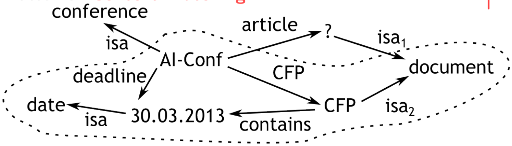
<u>Context graph</u>

$CG_A = (V, E)$
$V \subseteq Concepts$
$E = \{edge(from, to, value) \mid$
$from, to \in V,$
$value \in Relations\}$

<u>Context pattern</u>

$G_s^P = (V_s^P, E_s^P)$
$V_s^P \subseteq Concepts \cup \{?\}$
$E_s^P = \{edge(from, to, value) \mid$
$from, to \in V_s^P,$
$value \in Relations \cup \{\lambda\}\}$

Formal Model



Example

Model

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

$G'_A \subseteq CG_A, G^P_m = (V^P_m, E^P_m) \subseteq G^P_s$ – 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

$f_v : V^P_s \to V'$ – vertex correspondence (bijective) – with:

- $\forall v^P \in V^P_m, v^P = ?$ or $v^P = f(v^P)$
- $\forall edge(v^P_i, v^P_j, value) \in E^P_m, edge(f(v^P_i), f(v^P_j), value) \in E'$

Computer Science & Engineering Department
AI-MAS Group
Andrei Olaru
SYNASC 2013
Timisoara, Romania 26.09.2013

Matching Algorithm

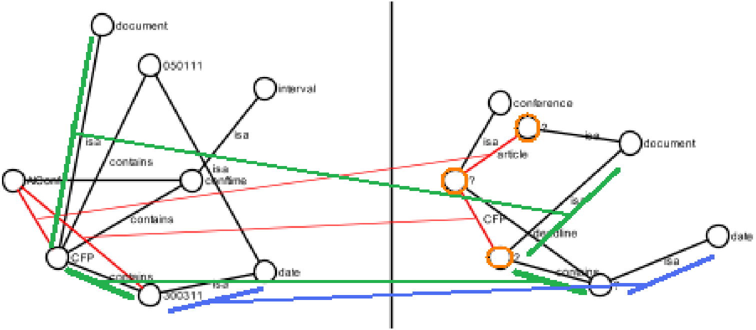
- Start from all valid matches of one edge in the pattern with one edge in the graph;
- For each initial match, detect which other matches are valid merger candidates;
- Iterate over matches and create new matches, by merging them to their merger candidates;

Matching Algorithm

- a match is represented as $M(G', G_m^P, f_v, f_e, fr, MC, MO, k)$

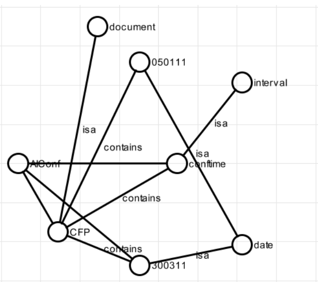
$G' \subseteq CG_A$ – matched subgraph of CG_A
$G_m^P \subseteq G_s^P$ – matched part of the pattern
$f_v : V_m^P \rightarrow V'$ – node function (bijective)
$f_e : E_m^P \rightarrow E'$ – edge function (bijective)

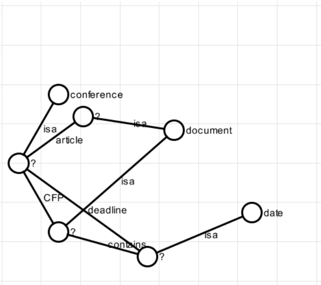
$fr \subseteq V_m^P$ – frontier
MC – merger candidates
MO – 'outer' merger candidates
$k = |E_s^P|\text{-}|E_m^P|$ – missing edges



Matching Algorithm

$Match(G, G^P)$

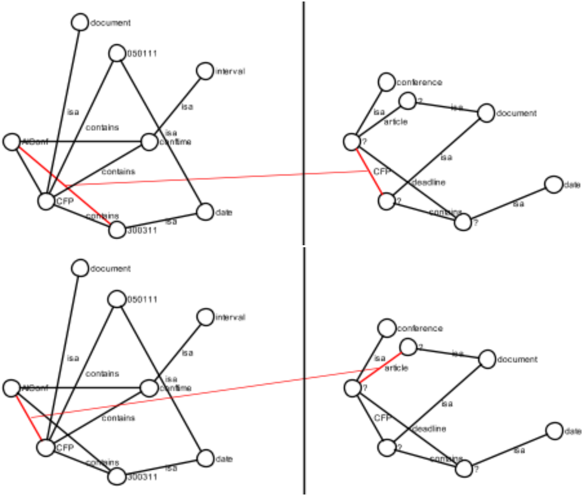




Matching Algorithm

$Match(G, G^P)$

$AddInitialMatches$



1.

2.

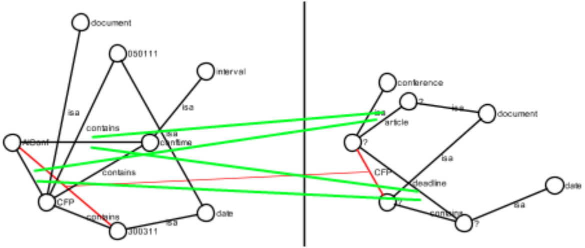
. . .

Matching Algorithm

Match(G, G^P)

AddInitialMatches

Immediate candidates



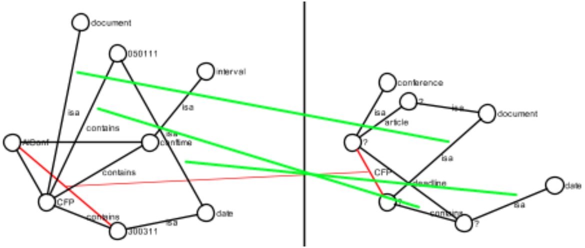
Matching Algorithm

$Match(G, G^P)$

AddInitialMatches

Immediate candidates

Outer candidates



Matching Algorithm

$Match(G, G^P)$

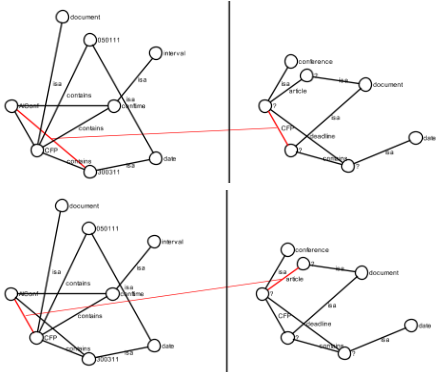
AddInitialMatches

Immediate candidates

Outer candidates

for each M', M''

+



Matching Algorithm

Match(G, G^P)

AddInitialMatches

Immediate candidates

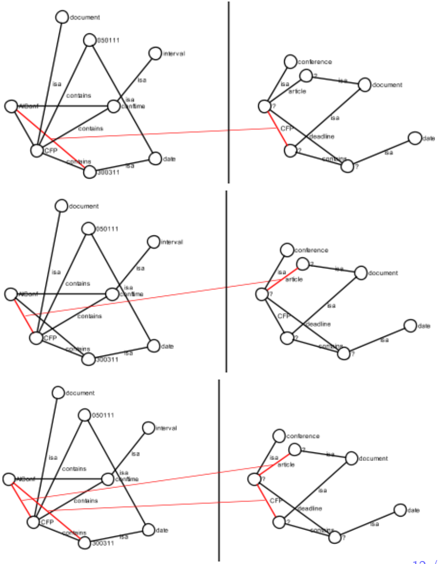
Outer candidates

for each M', M''

Merge(M', M'')

+

→



Matching Algorithm

AddInitialMatches

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

$Match(G, G^P)$

GrowMatches

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

Merge (M', M'')

$$V = V' \cup V''; E = E' \cup E''$$
$$V^P_m = V^{P'}_m \cup V^{P''}_m; E^P_m = E^{P'}_m \cup E^{P''}_m$$
$$f_v = f'_v \cup f''_v; f_e = f'_e \cup f''_e$$
$$fr = \{v^P \in fr' \cup fr'' \mid \exists e^P \text{ adj } v^P, e^P \notin E^P_m\}$$
$$MC = (MC' \cap MC'') \cup (MC' \cap MO'') \cup (MC'' \cap MO')$$
$$MO = MO' \cap MO''$$

Matching Algorithm

· While the classic problem of matching undirected, unlabeled graphs is NP-complete, for the problem at hand the algorithm behaves significantly better.

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: $\mathcal{O}(m \times m^P) + \mathcal{O}(initialMatches^2)$
$Merge(M', M'')$	Adds all edges and nodes to the new match Merges immediate and outer merger candidates Complexity: $\mathcal{O}(\lvert E_m^{P'}\rvert + \lvert E_m^{P''}\rvert)$
GrowMatches	Iterates over the match queue and merges matches with their candidates. Complexity: $\mathcal{O}(initialMatches \cdot log(initialMatches) \cdot average\,\lvert E_m^P\rvert)$

Algorithm Evaluation

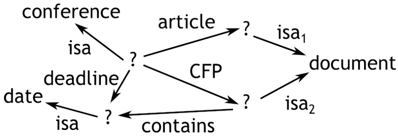
Algorithm	Akkoyunlu	Bron-Kerbosch	Balas-Yu	Durand-Pasari	Our algorithm
Expanded edges:					
Small example	124	120	135	119	34
Initial example	5431	5440	6423	5219	2459
No labeled edges	7054	9454	15843	9060	7581
No labels	326044	371943	578401	367725	108902
Large example	20470	19989	22170	18322	11834

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.

Visualization Tools

- 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.



manual

?#1
$(\xrightarrow{CFP}$?#4
$(\xrightarrow{contains}$?#3 $\xrightarrow{isa}$ date)
$\xrightarrow{isa}$ document)
$(\xrightarrow{deadline}$ *?#3)
$(\xrightarrow{article}$?#2 $\xrightarrow{isa}$ *document)
$\xrightarrow{isa}$ conference

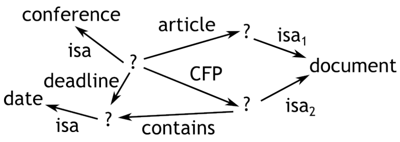
textual

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

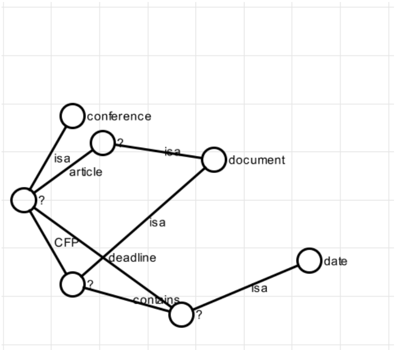
Console (ASCII)

Visualization Tools

- 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° arcs



manual



automatic layout

Conclusion and Future Work

- 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?

Akkoyunlu, E. (1973).
The enumeration of maximal cliques of large graphs.
SIAM Journal on Computing, 2(1):1–6.

Balas, E. and Yu, C. S. (1986).
Finding a maximum clique in an arbitrary graph.
SIAM Journal on Computing, 15(4):1054–1068.

Bron, C. and Kerbosch, J. (1973).
Algorithm 457: finding all cliques of an undirected graph.
Communications of the ACM, 16(9):575–577.

Cordella, L., Foggia, P., Sansone, C., and Vento, M. (2004).
A (sub) graph isomorphism algorithm for matching large graphs.
Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26(10):1367–1372.

Dey, A. (2001).
Understanding and using context.
Personal and ubiquitous computing, 5(1):4–7.

Dobrescu, A. and Olaru, A. (2013).
Graph matching for context recognition.
In Dumitrache, I., Florea, A. M., and Pop, F., editors, Proceedings of CSCS 19, the 19th International Conference on Control Systems and Computer Science, 29-13 May 2013, Bucharest, Romania, pages 479–486. IEEE CPS.

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., and Burgelman, J. (2001).
Scenarios for ambient intelligence in 2010.
Technical report, Office for Official Publications of the European Communities.

Durand, P. J., Pasari, R., Baker, J. W., and Tsai, C.-c. (1999).
An efficient algorithm for similarity analysis of molecules.
Internet Journal of Chemistry, 2(17):1–16.

El Fallah Seghrouchni, A. (2008).
Intelligence ambiante, les defis scientifiques.
presentation, Colloque Intelligence Ambiante, Forum Atena.

Koch, I. (2001).
Enumerating all connected maximal common subgraphs in two graphs.
Theoretical Computer Science, 250(1):1–30.

Larrosa, J. and Valiente, G. (2002).
Constraint satisfaction algorithms for graph pattern matching.
Mathematical structures in computer science, 12(4):403–422.

McGregor, J. J. (1982).
Backtrack search algorithms and the maximal common subgraph problem.
Software: Practice and Experience, 12(1):23–34.

Olaru, A., Florea, A. M., and El Fallah Seghrouchni, A. (2011).
Graphs and patterns for context-awareness.
In Novais, P., Preuveneers, D., and Corchado, J., editors, Ambient Intelligence - Software and Applications, 2nd International Symposium on Ambient Intelligence (ISAmI 2011), University of Salamanca (Spain) 6-8th April, 2011, volume 92 of Advances in Intelligent and Soft Computing, pages 165–172. Springer Berlin / Heidelberg.

Olaru, A., Florea, A. M., and El Fallah Seghrouchni, A. (2013).
A context-aware multi-agent system as a middleware for ambient intelligence.
Mobile Networks and Applications, 18(3):429–443.

Ramos, C., Augusto, J. C., and Shapiro, D. (2008).
Ambient intelligence - the next step for artificial intelligence.
IEEE Intelligent Systems, 23(2):15–18.

Weiser, M. (1993).
Some computer science issues in ubiquitous computing.
Communications - ACM, pages 74–87.

Thank You!

Any Questions?