Context Emergence Using Graph Theory
Defining and Modeling Context for Industrial Assets Using Graphs

Bharath Rao and Arila Atanassova-Barnes
GE Software,2623 San Ramon Blvd, San Ramon, CA, USA
rao@ge.com, arila.barnes@ge.com

Keywords: Context, Graph theory, IoT, Industrial Internet, Modeling.

Abstract: Traditionally, context in software is modeled as a global variable, static class, or similar mechanisms that are initialized when an application is loaded and updated periodically through the lifetime of the application as the end user interacts with instructions. This approach is limited to customizations, personalization and initialization based on previously captured, mostly static information. It is a replay of a previous state. In this paper we propose a new approach to defining and modeling context for software applications using graphs. Context is fundamentally interaction-based and comes into play when one entity interacts with another to achieve a goal in a given environment constrained by time and location. By capturing the interactions between entities in a graph, context becomes emergent rather than declarative and can be learned from user interactions. The context is discovered by first-degree graph traversal of interacting entities. The discovered context is used to achieve context sensitive goals in environments with a large number of interconnected entities such as the Internet of Things (IoT).

1 INTRODUCTION

To support decisions based on goals and actions in industrial scenarios, it is critical to detect and evolve context of operational environments in such a way that the system remains adaptive and dynamic and can react to changing goals in real time (Loren, 2015).

For the purpose of this paper we start with the following definition of context: 'Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.' (Abowd et al., 1999)

Entities can be modeled as vertices in a graph and relationships can be modeled as edges. This enables us to represent any set of relationships as a graph. Using graph relationships and paths, the use of simple graph operations enables us to determine context and leverage it to achieve goals. We present a novel approach that allows context to emerge organically based on the entity relationship graph rather than being explicitly maintained.

2 GRAPH APPROACH TO CONTEXT

2.1 Information as a Graph

All known information at any given time may be represented as a graph. The graph itself is made up of a set of vertices and edges.

$$G = (V, E)\,. \tag{1}$$

In an open world assumption, all vertices are presumed to exist and a graph can simply be defined as a set of edges bounded by two vertices. Such a graph can be described as a directed, labeled multi-graph. Such multi-graphs may be modeled as a set of triples (W3C, 2014), each triple t is a 3-tuple containing the subject s, predicate p and object o representing an arc in a directed graph. The subject is the vertex at the origin of the arc, the object is the destination and the predicate is the label on the arc.

$$t = (s, p, o)\,, \tag{2}$$
$$\mathcal{G} = \{t\}\,. \tag{3}$$

Rao B. and Atanassova-Barnes A.
Context Emergence Using Graph Theory - Defining and Modeling Context for Industrial Assets Using Graphs.
DOI: 10.5220/0005887302130217
In *Proceedings of the Fifth International Symposium on Business Modeling and Software Design (BMSD 2015)*, pages 213-217
ISBN: 978-989-758-111-3

Maintaining the relationship graph then simply becomes a matter of adding a triple when a new relationship is formed and removing a triple when a relationship is dissolved and archived.

2.2 Context as a Function of Graph

For context modeling we present a light asset-centric ontological approach (RDF) and leverage graph theory for both the storage of the asset related data and the inference of new contexts dependent on goals.

As edges are added and removed over time, a characteristic pattern develops around an entity as an emergence (Lewes, 1874). This emergent context may be used to avoid manually maintaining a context.

More context does not necessarily improve the accuracy of the inference in a considerable manner. (Guan et al., 2007) Rather than creating an exhaustive context model, our approach is to let it grow organically.

Definition 1: *The context of any entity is the set of its first-degree connections*.

$$C_\varepsilon = \{t \mid s = \varepsilon \vee o = \varepsilon\} . \quad (4)$$

The entity ε may be connected to other entities or may simply point to a value. This includes both inbound and outbound connections in case of directed graphs.

Definition 2: *The common context between any two entities is the intersection of their individual contexts, i.e., their first-degree connections.*

$$C_{AB} = C_A \cap C_B . \quad (5)$$

Definition 3: *Emergent Context of entities is the union of their contexts that naturally arises as old connections dissolve and new connections form.*

$$C_E = C_A \cup C_B . \quad (6)$$

Definition 4: *Goal based context is a subset of the emergent context that is relevant for the aforesaid goal.*

$$C_G \subseteq C_E . \quad (7)$$

2.3 Graph Path Notation

We introduce the following notation for describing sets of graph paths from $\mathcal{G}$, which may be effectively used as queries.

A path P is a non-empty sub-graph of $\mathcal{G}$ such that the edges e_i connect one vertex to the next and every vertex x_i is distinct.

$$P = \{(x_0\, e_0 x_1), (x_0 e_1 x_1), \ldots, (x_{k-1} e_k x_k)\},$$
$$e_i = {}^{>}e_i \vee {}^{<}e_i , \quad (8)$$

Where ${}^{>}e_i$ and ${}^{<}e_i$ represent outgoing and incoming arcs from vertex x_i respectively.

Let the path expression ${}^{=}s_i$ be the subset of all paths in $\mathcal{G}$ that start with vertex s_i.

$${}^{=}s = \{P \mid x_0 = s\} . \quad (9)$$

Let the path expression ${}^{>}p$ be the subset of all paths in $\mathcal{G}$ that start from any subject having a predicate p (i.e., all paths beginning from any vertices and an outgoing edge with label p) to any vertices.

$${}^{>}p = \{P \mid (x_0\ {}^{>}e_0\, x_1) = (x_0\, p\, x_1)\} . \quad (10)$$

Let the path expression ${}^{<}p$ be the subset of all paths in $\mathcal{G}$ that start from any object having a predicate p (i.e., all paths beginning from any vertices and an incoming edge with label p) to any vertices.

$${}^{<}p = \{P \mid (x_0\ {}^{<}e_0\, x_1) = (x_0\, p\, x_1)\} . \quad (11)$$

Let the path expression ${}^{\circ}p$ be the subset of all paths in $\mathcal{G}$ that have p as its first predicate.

$${}^{\circ}p = {}^{>}p \cup {}^{<}p . \quad (12)$$

The path elements in (9) to (12) can be chained to specify a subset of paths that follow any specific pattern. For example, the path expression ${}^{=}s\,{}^{>}p$ is the subset of all paths from (9) that begin with subject s and predicate p (i.e., all paths beginning from a vertex s and an outgoing edge with label p) to any vertices.

$${}^{=}s\,{}^{>}p = \{P \mid x_0 = s \wedge {}^{>}e_0 = p\} . \quad (13)$$

The expression ${}^{=}s\,{}^{>}p\,{}^{>}q$ is the subset of all paths from (13) that have a successor predicate q.

$${}^{=}s\,{}^{>}p\,{}^{>}q = \{P \mid x_0 = s \wedge {}^{>}e_0 = p \wedge {}^{>}e_1 = q\} .$$

The expression ${}^{=}s\,{}^{>}p\,{}^{=}o$ represents the set of all paths beginning with the subject s and having an outgoing predicate p and an incoming object o.

$${}^{=}s\,{}^{>}p\,{}^{=}o = \{P \mid x_0 = s \wedge {}^{>}e_0 = p \wedge x_1 = o\} .$$

Let the path expression $\pi_0\{\pi_1, \pi_2\}\pi_3$ represent a branched path which splits after π_0 into as many branches as the number terms within the braces where π_i represents a chain of path expressions defined in

(9) to (12). This notation may be nested, enabling us to specify arbitrary branching.

$$\pi_0\{\pi_1,\pi_2\}\pi_3 = \pi_0\{\pi_1\pi_3,\pi_2\pi_3\} = \{\pi_0\pi_1\pi_3,\pi_0\pi_2\pi_3\},$$
$$\pi_0\{\pi_1,\pi_2\}\pi_3 = \pi_0\pi_1\pi_3 \wedge \pi_0\pi_2\pi_3 . \quad (14)$$

For example,

$$^{=}s\,\{^{>}p,\,^{>}q\} = {}^{=}s\,^{>}p \wedge {}^{=}s\,^{>}q .$$
$$^{=}s\,^{\circ}\{p,q\} = {}^{=}s\,^{\circ}p \wedge {}^{=}s\,^{\circ}q .$$

2.4 Contextual Goals

Defining how the system behaves in a given context is key to context sensitive computing. This context-specific behaviour can be modeled as a set of goals. Goals are composed of constraints and actions. In a general graph, especially one as large as the IoT, there could be billions of entities and some of the more complex entities such as power plants may have thousands of goals per entity. In an environment where multiple entities interact, prioritizing goals to evaluate can be a complex problem. An entity needs to quickly prioritize goals to evaluate. Mapping the common context from Definition 2 to a set of goals would ensure that goals in the common context are given priority.

In our example below, goals relating to `California Plant 3` and `Repair Process` are prioritized and can be obtained by a simple lookup.

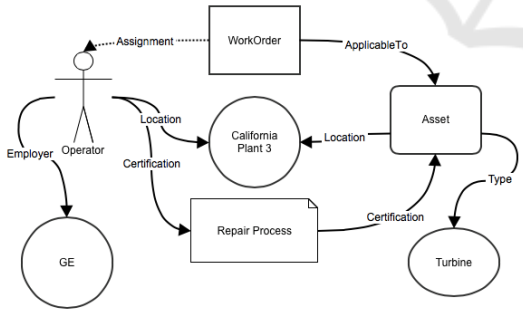

Figure 1: Portion of graph of entities and their known relationships.

Figure 1 represents a small portion of the graph that's relevant to a specific goal, the assignment of a work-order to a qualified operator. The Assignment link from WorkOrder to the operator does not yet exist; adding it is the goal of this exercise.

From our definitions, we have:

Operator's context, from (4):

```
Location: California Plant 3
Certification: Repair Process
Employer: GE
Type: Operator
```

Asset's context, from (4):

```
Location: California Plant 3
Certification: Repair Process
applicableTo: WorkOrder
Type: Turbine
```

Common context for the Operator and Asset Interaction, from (5):

```
California Plant 3
Repair Process
```

2.5 Contextual Constraints

Constraints define when a goal is applicable and are specified as a set of paths. If all constraint paths specified by the goal exist, then the goal can be evaluated.

Let φ be the path expression that represents a contextual constraint as in Section 2.3

$$\varphi = \pi_0\pi_1 .. \pi_k . \quad (15)$$

φ is said to exist when it contains at least one path satisfying φ. For a branched path as in (14), all branches should exist.

Example Contextual Constraint 1: Only an operator present in the same location should service Asset.

$$Asset^{>}location\ ^{<}location\ ^{>}type\ ^{=}operator$$

Decomposing the path expression above by its constituents, the initial term $Asset^{>}location$ represent the location of Asset, the succeeding term $^{<}location$ represents all entities that are in the same location as Asset and the final term $^{>}type\ ^{=}operator$ limits the above to our contextual goal of all operators in the same location as Asset.

Example Contextual Constraint 2: Only an Operator who is a GE employee and is certified in the relevant repair process should service Turbines.

$$Turbine\ ^{<}type\ ^{\circ}certification\ ^{\circ}\{type\ ^{=}operator,\ employer\ ^{=}GE\}$$

This path expression uses the direction agnostic operator from (12) and branching from (14)

2.6 Contextual Actions

Actions are simply graph changes such as new arcs added to the graph. These added arcs grow the context and are the elemental constituents of the emerging context.

A contextual action Γ specifies the exact paths to subject and object and supplies the predicate used to insert or remove an arc into the context graph.

Let $\prod$ represent the union of all contextual constraints φ_0 to φ_k

$$\prod = \varphi_0 \cup \varphi_1 \cup \ldots \cup \varphi_k \ . \quad (16)$$

Let $\Gamma_{\prod}$ be the contextual action where p_Γ represents the predicate to be added and φ_s and $\varphi_o$represent the path to the subject and object within $\prod$

$$\Gamma_{\prod} = (\varphi_s, p_\Gamma, \varphi_o) \ . \quad (17)$$

Example Contextual Action: Assign WorkOrder to Operator.

$$\Gamma_{\prod} = (Asset^{<}applicableTo, Assignment \; Asset^{>}location^{<}location) \ . \quad (18)$$

In the above:
Subject path: $Asset^{<}applicableTo$
Predicate: $Assignment$
Object path: $Asset^{>}location^{<}location$

From figure 1, its clear that both constraint paths exist in $\mathcal{G}$ and therefore the action can be performed. The new arc can be inserted at the location specified in (18). If multiple matches are found, then the system may simply pick one. If none are found, the operation may not be performed or be deferred to a human.

3 APPLICATIONS

We believe that the technique above is generally applicable to any software-defined context.

We have shown an example of operator assignment based on discovered context. This can be extended to the context discovery of any interacting entities whose relationships and attributes are modeled as a graph.

Context discovery and modeling in large graphs such as the Internet of things (IoT) poses special challenges (Pereira et al., 2013). The IoT is a very large graph, possibly in the tens or hundreds of billions of entities and quickly finding the context of any entity or between any two entities is critical to scalability.

In addition, many IoT entities are likely to lose connectivity to the Internet but need to interact with other entities that they are connected to on a local or ad-hoc network.

Our approach enables us to emerge context in a decentralized manner by focusing on the context of any interacting entities and deal with the challenge of huge graphs by reducing the context to a small subgraph as described in section 2.

Entities can maintain a graph model of their attributes and relationships and let the context emerge naturally over their lifetime. This enables them to act independently without the need for a central repository that aggregates and serves as the authority for context of billions of entities.

4 RELATED WORK

Goals in the IoT and the Industrial Internet space are related to events. Detecting events in real time is another major challenge for context-aware frameworks in the IoT paradigm (Pereira et. al, 2013). The graph of entities and their relations is continuously and dynamically updated as the users of the system interact with the graph triggered by sensor events or time-based events.

Nguyen et. al. address this challenge using a context-aware framework that uses a form of context graph. They introduce the notion of a graph made of context nodes and action nodes: 'the basic idea of contextual graph relies on the fact that past contexts can be remembered and adapted to solve the current context. The context is managed to organize in the graph type. In the contextual graph, rather than creating a solution from scratch, the contexts similar to the current context are retrieved from memory. The best match is selected and adapted to fit the current context based on the differences and similarities between the two contexts.' (Nguyen et. al., 2008.)

Their approach is also focused on the paths going through a node to discover context. We improve upon Nguyen et al. by further constraining the model by introducing the definitions of context goal, constraints, and common context.

5 CONCLUSIONS AND FUTURE WORK

No single system yet exists that could fully satisfy the criteria and challenges for the context-aware requirements for any large system of interconnected entities, especially one on the scale of IoT. The inherent complexity requires more dynamic approach that emerges from the system interactions as the system evolves rather than being pre-calculated. We believe that our graph approach to emergent context gets us closer to the larger goal of a complete contextual system for the Internet of Things.

Our approach can be generalized by extending Definition 1 to context from one to the nth degree, enabling a much larger graph and more indirect constraints and goals.

We could also extend the approach to interaction of multiple entities by extending Definition 2 from two to n entities.

Although Definition 3 implies an archived historical record of every link ever formed and dissolved, we only use the current state of the graph in this paper. The ability to use historical arcs could open new possibilities such as affinity discovery.

GE Software is currently investigating methods connected to this paper's contributions on a broad class of complex industrial and IoT applications such as analytics on large asset graphs.

REFERENCES

Loren, J., 2015. Big Data Meets Big Context. https://www.gesoftware.com/blog/big-data-meets-big-context

Abowd, G, Dey, A., Brown, P., Davis, N., Smith, M., and Steggles, P.,1999. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing HUC'99,* London, UK.

W3C Working Committee, RDF 1.1 Concepts and Abstract Syntax 2014, http://www.w3.org/TR/rdf11-concepts/

Lewes, G. H.,1874. *Problems of Life and Mind (First Series)*, https://ia700500.us.archive.org/2/items/problemsoflifemi01leweiala/problemsoflifemi01leweiala_bw.pdf

Guan, D, Yuan, W., Lee, S., and Lee, Y.-K. 2007. Context selection and reasoning in ubiquitous computing. In *Intelligent Pervasive Computing, 2007 IPC,* The 2007 International Conference on, oct 2007, pp. 184-187.

Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.,2013. Context Computing for the Internet of Things. In *IEEE Communication Surveys & Tutorials Journal*.

Nguyen, T., Lopes,Wontaek, L., Nguyen, H., Choi, D., 2008. Context Awareness Framework based on Contextual Graph. In *WOCN'08, 5th International Conference on Wireless and Optical Communications Network*. East Java Indonesia.