

Context Inconsistency Management Using Partial Constraint Checking

Samuel Esposito

Alexander Jurjens

Abstract—Thanks to the pervasive computing paradigm more and more computer systems in utility buildings and industry are context-aware. They use a representation of the world they operate in to reduce the human-computer interaction necessary for their operation. Unfortunately reasoning based on contexts is not without flaws and context inconsistencies are the main reason for context-aware applications' incongruous behavior. Context consistency management is not adequately studied in existing literature and approaches for detecting and resolving context conflicts are not suited for pervasive computing [1].

In this paper we present two complementary approaches for improving the mitigation of context inconsistencies. First we present partial constraint checking for timely identifying context inconsistencies at runtime. An extra constraint layer is added to the traditional ontology based context model and conflicts can be detected by locally checking partial constraints in the ontology. This dramatically improves performance compared to iterative evaluation of an entire ontology [1]. Secondly we discuss the extension of the traditional ontology model with context lifecycles to more accurately represent the environment of context-aware applications. This information can then be used to estimate the relative reliability of contexts in a conflict set and discard the contexts with lowest reliability [2]. Apart from resolving context conflicts it is also possible to represent inconsistencies into the context model itself. In this paper we explore the possibilities of incorporating inconsistencies into ontologies using fuzzy OWL and discuss the consequences of this approach on standard reasoning methods [3].

Index Terms—Pervasive Computing, Ontology Model, Context Lifecycle, Inconsistency Resolution.

1 INTRODUCTION

In pervasive computing, contexts are used to make applications perform a certain action. Contexts can be seen as pieces of environmental information, which the applications depend on. Sometimes contexts are not consistent due to crossreads and misreads. The correctness of the contexts can be checked by evaluating the consistency constraints associated to them. There are multiple ways to check for consistency constraints. Constraint checking techniques are split up in two classes: Non-incremental checking and incremental checking. Incremental checking consists also of two classes: Entire Constraint Checking and Partial Constraint Checking. Once inconsistent contexts have been detected, an algorithm can be used to select the correct contexts. The algorithms use specific datastructures, like ontology models, in order to resolve the inconsistencies. However, certain conflict resolving algorithms are too slow for use in practice. This paper describes a way to use an ontology model in combination with the Partial Constraint Checking algorithm in order to detect context inconsistencies fast enough, such that it can be used in practice.

2 RELATED WORK

Pervasive or ubiquitous computing is a fast-developing discipline that has been receiving increasing attention from both researchers and software developers [1]. In the past decade, many context-aware systems have been developed, ranging from smart room environments to warehouse and supply chain management systems. Lots of effort has been put into building middleware infrastructures that handle vast amounts of sensory data and extract the context information relevant for pervasive applications. Examples of such systems are CoBrA and CORTEX [2]. Various modeling approaches have been proposed for capturing context information, of which the ontology based context model appears to be most promising for most pervasive applications [2].

Context management for consistency however has not been adequately studied in the existing literature. None of the studies on context-awareness discusses a way for detecting context inconsistencies for reliable pervasive computing [1, 2]. Even though there has been related research going on in other disciplines as artificial intelligence and software engineering, it doesn't provide adequate support for context inconsistency detection in ubiquitous computing. In addition the strategies proposed in literature for resolving context conflicts are not suited for pervasive computing. Some are based on assumptions that may not apply to general pervasive environments. Other require human participation for conflict resolution, which is usually

expensive and slow for pervasive computing [1]. Finally no research has been done on the potential of fuzzy ontologies for representing context inconsistencies in the environment instead of trying to resolve them [3].

In this article we aim at putting a milestone for context management by presenting an efficient inconsistency detection algorithm based on a constraint language extending the traditional ontology based context model [1]. In addition we put forward a conflict resolution algorithm which is based on a context reliability heuristic [2]. Finally the use of fuzzy ontologies representing context inconsistencies as a promising alternative to conflict resolution is explored.

3 PARTIAL CONSTRAINT CHECKING

Constraint checking techniques have been extensively studied in software engineering. Existing constraint checking algorithms focus on checking software artifacts that do not change rapidly or frequently [1]. Context-aware applications require more efficient algorithms because they use a huge set of contexts, which can change very rapidly and frequently (in the range of milliseconds). An inefficient software solution for this does not only require massive computing power, but interestingly also induces a higher inconsistency detection miss rate: because of the computing delay conflicting contexts slip through the context buffer before they are detected by the software [1]. One example of an inefficient approach is nonincremental checking: whenever there is a change in the set of software artifacts these artifacts are each checked against the entire set of consistency constraints to find out all detectable inconsistencies. An improvement to this would be incremental checking: only a subset of all constraints are checked, namely those that are affected by the specific change in the artifact set. But the real merit of Xu *et al.* was replacing the traditional entire constraint checking approach by partial constraint checking based on a consistency computation tree [1]. Their idea is that constraints can be represented as trees with nodes for logical operations and leaves for specific properties of contexts or context sets. Whenever a context is added to or deleted from a context set in time, the branch corresponding to this context can respectively be added to or deleted from the tree (see Fig. 1). Because intermediate values are retained in the tree nodes after calculation they can be reused whenever the tree changes. More specifically, when a branch is added, only the values for the new branch itself and for the nodes from the branch top to the tree root need to be calculated. When a branch is deleted,

