A Design Calculus for Structured Data

Jia Lin ∗1, Christian Horn1 and Fiona Lawless1

1Department of Computing and Mathematics, Dundalk Institute of Technology

1 INTRODUCTION

Software Configuration Management (SCM) systems are used to store and track the changes made to intermediate versions of general text documents, which arise during the development of a software application. Such tools provide facilities to identify, patch and/or merge the differences between these versions.

Traditional SCM systems rely on existing diff tools which are limited in their scope. Additional diff tools have been developed for other common data structures, such as XML files and even spreadsheet files. However, regardless of the type of the object being updated, such changes take the form of a deletion and/or an insertion. Therefore, a generic version control tool could be developed to identify the difference between objects of arbitrary data type.

To achieve this, we have created a theoretical framework to formalise the elements of version control tools based on a type system. It allows us to model arbitrary data structures and to develop a version control tool for any structured data.

2 RELATED WORK

Some previous work has been done to detect difference between general structured data. Most of this work converts any data structure into a tree, then changes the problem to one of identifying difference between two trees. Lempsink et al., in their paper “Type-safe diff for families of data types” [Lempsink et al., 2009], implemented a generic diff and patch using a functional programming language in which they defined a Diff type as a set of edit operations. The investigation started by applying the diff and patch algorithms to lists and trees respectively, a generic diff algorithm based on converting any data structure to a rose tree was then developed.

3 OUR APPROACH

Instead of converting structured data to a tree, we create a theoretical framework which involves a Design Calculus built on a type system. This Design Calculus is a mathematical model which is used to formalise version control facilities.

The investigation of our approach starts with the construction of a type system \( \mathcal{T} \) containing of primitive and structured types. Primitive types are the foundation type, structured types are built recursively on primitive types and/or structured types:

- Product types \( (T_1 \times \ldots \times T_n) \)
- Sets \( (\text{Set}(T)) \)
- Multisets \( (\text{MSet}(T)) \)
- Lists \( (\text{List}(T)) \)
- Mappings \( (S \Rightarrow T) \)

Here \( S, T, T_1, \ldots, T_n \in \mathcal{T} \) can be any type in \( \mathcal{T} \). This approach allows us to handle more data structures recursively. Except for the primitive types in this type system, the difference between structured objects of the same type is determined from the difference of their underlying components.

Operators which are used to identify the difference between objects of each type are defined in our Design Calculus model. We have implemented a directed difference operator \( \delta_T \), used to identify the directed difference between objects of the same type, and a similarity measure operator \( \mu_T \), which determines the similarity measure between them.

Moreover, this Design Calculus is computer language independent. It is a mathematical model which can be implemented on any computer language platform that supports generic types. For our implementation of this model, we use Java generics to develop each type in the type system \( \mathcal{T} \) and all operators in each type.

In order to evaluate the Design Calculus, two applications have been developed by applying this Design Calculus to JSON data [ECMA-404, 2013] and the graph data structure.

4 CONCLUSIONS

The results of applications indicate that it is possible to use a universal approach to develop a version control tool for arbitrary structured data. By applying the Design Calculus to objects modelled on the type system, the directed difference and the similarity measure are obtained. Work continues to complete the framework and to apply this Design Calculus to more complex structured data.

References