Advanced Engineering Informatics 33 (2017) 397–409

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Full length article

A string-wise CRDT algorithm for smart and large-scale collaborative editing systems ☆

Xiao Lv [a,c], Fazhi He [a,b,*], Weiwei Cai [a], Yuan Cheng [a]

[a] State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China
[b] School of Computer Science and Technology, Wuhan University, Wuhan 430072, China
[c] Department of Computer Engineering, Naval University of Engineering, Wuhan 430033, China

ARTICLE INFO

Article history:
Received 29 June 2016
Received in revised form 9 October 2016
Accepted 27 October 2016
Available online 24 November 2016

Keywords:
Smart and large-scale collaborative editing
String-wise operation
Operational transformation
Commutative replicated data type

ABSTRACT

With the development of big data and cloud computing, real-time collaborative editing systems have to face new challenges. How to support string-wise operations for smart and large-scale collaborations is one of the key issues in next generation of collaborative editing systems, which is both the core topic of collaborative computing area and the fundamental research of many collaborative systems in science and engineering. However, string-wise operations have troubled the existing collaborative editing algorithms, including Operational Transformation (OT) and Commutative Replicated Data Type (CRDT), for many years. This paper proposes a novel and efficient CRDT algorithm that integrates string-wise operations for smart and massive-scale collaborations. Firstly, the proposed algorithm ensures the convergence and maintains operation intentions of collaborative users under an integrated string-wise framework. Secondly, formal proofs are provided to prove both the correctness of the proposed algorithm and the intentions preserving of string-wise operations. Thirdly, the time complexity of the proposed algorithm has been analyzed in theory to be lower than that of the state of the art OT algorithm and CRDT algorithm. Fourthly, experiment evaluations show that the proposed algorithm outperforms the state of the art OT algorithm and CRDT algorithm.

1. Introduction

Collaborative editing systems (CESs) allow multiple geographically dispersed users to view and edit the shared document over computer networks, which have been a core topic of continuous research in Computer Supported Cooperative Work (CSCW). Over the past 25 years, an increasing number of collaborative editing algorithms have been researched, developed and applied for collaborative systems in science and engineering, e.g. Google Wave/Docs,[1] 2D spreadsheets [1], 2D images [2,3], 3D digital media design systems [4–6], 2D/3D Computer-Aided Design [7–10] and so on.

More recently, with the development of big data and cloud computing [11–17], CESs increasingly tend to smart and large-scale collaborations, which have to face new technical challenges.

Smart and large-scale CESs support tens or hundreds of collaborators to share and exchange the intention, idea, knowledge and wisdom of people in a large-scale collaborative scenario [18–20]. As stated in [21], the CESs should "make a more intelligent and semantically meaningful usage of the network resources". Only if the "atomic operation of collaborative editing" is advanced from "character-wise operation" to "string-wise operation", will the knowledge-based collaboration be effectively supported. In other words, the collaborative editing operations are always "knowl edge-grained" such as blocks or paragraphs in smart and massive-scale collaborations. Therefore, the string-wise CESs become the foot-stone of smart collaborations.

In addition, for a large-scale collaboration [22], in which a large amount of users edit the shared document simultaneously, the shared document will be updated frequently. This situation will generally lead to the decrease of collaborative computing performance [23,24]. How to enhance the computing performance is another challenge for the success of smart and massive-scale collaborative editing systems. The "string-wise operation" has a potential advantage over the "character-wise operation" for high efficient collaborations in large-scale collaborative applications.

In a short, string-wise CESs have been the research focus for smart and large-scale collaborative applications in the time of big data and cloud computing.

☆ Fully documented templates are available in the elsarticle package on CTAN.

* Corresponding author at: State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China.
E-mail address: fzhe@whu.edu.cn (F. He).

[1] http://docs.google.com.

http://dx.doi.org/10.1016/j.aei.2016.10.005
1474-0346/

As a great extension of CSCWD2016 paper [25], this paper proposes a string-wise CRDT algorithm for smart and large-scale collaborative editing systems to solve the new challenges. Major contributions of this paper are listed below.

(1) The proposed algorithm can preserve operation intentions of collaborative users and maintain the consistency of the shared document.
(2) The proposed algorithm has been formally proved its correctness and the intention preserving of string-wise operations as long as it can satisfy two conditions operation commutativity (OC) and precedence transitivity (PT).
(3) The time complexity of the proposed algorithm is analyzed in theory to be lower than that of the state of the art OT algorithm and CRDT algorithm.
(4) The experiment evaluations show that the proposed algorithm has better computing performance than that of the state of the art OT algorithm and CRDT algorithm.

2. Background and related work

A fully replicated architecture is adopted in CESs in order to achieve high responsiveness, which brings a great challenge for the consistency maintenance [26–29]. OT algorithms are particularly suitable for consistency maintenance and have been proposed for nearly three decades [1–5,7,8,26,27,30–39]. A plethora of OT algorithms have been increasingly developed for collaborative applications in sciences and engineering, such as Jupiter [30], Nice [31], IBM OpenCoWeb,[2] CoWord.[3] The main idea of OT algorithms is that local editing operations are executed as soon as they are issued and then propagated to remote sites. Remote operations need to be transformed with concurrent operations before their executions in order to repair divergence. The major advantage of OT algorithms is the high responsiveness of local operations. Multiple users may freely and simultaneously generate and edit local operations. Despite the good local responsiveness, how to support string-wise operations has troubled the existing OT algorithms.

- Since the first OT algorithm developed by Ellis and Gibbs [26], most published OT algorithms only support character-based operations due to the inherent sophistication of OT, which are not suitable for smart and large-scale collaborations. GOT is the first work which describes how to support string-based operations [27], but no published work shows how to achieve string-based operations [36]. ABTS supports string-based primitive operations and handles overlapping and splitting of operations, but the time complexity is $O(|H|^2)$ [36]. Based on ABTS, ABTSO has improved the time complexity to $O(|H|)$ by keeping history operations according to the operation effects relation [35]. To the best of our knowledge, ABTSO has the best computing performance in a representative class of OT algorithms in publications. Despite the good computing performance, there is a space for improvement. In addition, ABTSO cannot support string-wise deletions.

In recent years, another class of collaborative editing algorithms called CRDT have been proposed and gradually become the hot research in collaborative computing and distributed computing [23,40–46]. The main idea of CRDT algorithms is to design commutative concurrent operations. Hence, transformations are not required anymore and concurrent operations can be executed in any order. By assigning unique identifiers for all objects of operations, CRDT algorithms can place all objects into abstract data structure in a total order. Therefore, CRDT algorithms can preserve operation intentions of collaborative users and guarantee eventual consistency. CRDT algorithms have been proved to outperform traditional algorithms by a factor between 25 and 1000 [44,45]. However, CRDT algorithms are quite young, how to support string-wise operations has been a challenge issue.

- In the typical CRDT algorithms, except the literature [23,45], most existing CRDT algorithms only support character-based operations. The literature [45] is based on WOOT [40], this work uses a WOOT-like way to sort concurrent strings in the same position, the time complexity of integrating remote insertions is $O(k^2), k$ is the number of the concurrent insertions. With the increase of the number of concurrent insertions, it costs much higher computing time. The literature [23] is based on LOGOOT [41,42], strings are assigned to unique compressed identifiers with letters of the alphabet, which can reduce the memory consumption. However, the literature [23] only support unbreakable line-based operations, which cannot handle splitting of lines. In addition, similar to LOGOOT, how to make sure causality is not given.

3. Proposed algorithm

3.1. The integrated string-wise framework

A real-time collaborative system consists of a large number of collaborative sites. Every site maintains a two-layer data structure including *View* and *Model*. *Model* is composed of a hash table called *HT* and a double-linked list named L_{model}. L_{model} links all visible and invisible nodes in a total order. Every node represents a string including a paragraph or a block. *HT* stores all original and splitting nodes. *View* is composed of a double-linked list named L_{view}, which can provide the interaction interface for collaborative users. L_{view} links all the visible nodes of L_{model}.

The whole framework is shown in Fig. 1. The control procedure is as follows. A user at each site can concurrently generate local operations and receive remote operations from other sites. The integrated procedures of both local operations and remote operations include two steps. Firstly, local and remote operations need to find the target node in *HT* with unique identifiers. Secondly, the correct operation position needs to be found in L_{model} before their executions. The procedure of synchronization between *View* and *Model* needs to make the effects of integrated updates appear in *View*.

3.2. Basic operations and splitting functions

The basic primitive operations include string-wise insertions and string-wise deletions. A user may run the following operations as follows.

(1) LocalInsert(*ID tar_key*, *int pos*, *string str*, *ID key*). (2) RemoteInsert(*int pos*, *ID tar_key*, *string str*, *ID key*). (3) LocalDelete(*ID tar_key*, *int pos*, *int del_len*, *ID key*). (4) RemoteDelete(*int pos*, *int del_len*, *ID key_list*, *ID key*). The parameter *tar_key* is used for finding the target node in *HT*. The parameter *pos* is an integer index, which is used for finding the operation position in L_{model}. The parameter *str* is the inserted string, which is specially used for an insertion. The parameter *key* is the identifier of the inserted(deleted) string. The parameters *del_len* is the length of the deleted string, which is specially used for a deletion. The parameter *key_list* is used for reserving multiple *IDs* of deleted nodes.

The target node may be split by current operations. The splitting cases are as follows. (1) The target node is split into two sub-nodes by insertions or deletions, which is shown in

[2] https://github.com/opencoweb/coweb#readme.
[3] http://www.codoxware.com.